# BAB 2
# LANDASAN TEORI

## 2.1 Investasi

Investasi merupakan terminologi yang erat hubungannya dengan bidang keuangan dan ekonomi. Konsep ini berkaitan dengan akumulasi suatu bentuk aset dengan harapan mendapatkan keuntungan di masa depan. Sebelum dikenalnya investasi, banyak orang yang hanya menyimpan uang mereka dalam bentuk tabungan. Namun, seiring dengan perkembangan zaman, orang-orang mulai beralih dari metode kuno tersebut dan mulai membeli saham, obligasi, emas, reksadana, dan jenis investasi lainnya yang dianggap memberikan potensi keuntungan yang menjanjikan di masa depan. Sebelum melakukan investasi pada suatu instrumen investasi, tentunya seorang investor harus memahami dan mempelajari segala aspek yang terkait dengan investasi tersebut [12].

Pada umumnya, lahan investasi terdiri dari tiga kategori, yaitu investasi keuangan, investasi komoditi perhiasan (seperti emas, intan, lukisan, dan lain-lain), dan investasi pada sektor riil. Investasi keuangan merupakan investasi yang memiliki objek berupa uang, valuta asing, serta surat-surat berharga yang diterbitkan oleh industri perbankan, seperti sertifikat deposito, *commercial paper*, SBPU (Surat Berharga Pasar Uang), dan sejenisnya. Investasi komoditi perhiasan, di sisi lain, meliputi investasi pada barang-barang perhiasan. Sedangkan investasi sektor riil mengacu pada investasi yang diwujudkan dalam pendirian pabrik atau pembukaan perkebunan pertambangan, dan sejenisnya. Oleh karena itu, dalam memilih jenis investasi yang tepat, investor harus mempertimbangkan kategori investasi yang ingin dilakukannya [3].

## 2.2 Emas

Emas adalah komoditas utama di pasar ekonomi dan moneter. India dan Cina adalah importir utama di dunia dan mengonsumsi 60% dari emas global. Setiap hari, nilai emas meningkat dan tidak bisa dikontrol. Saat ini, orang cenderung berinvestasi di emas karena keuntungan besar di masa depan. Harga emas sangat terkait dengan komoditas lainnya. Kenaikan harga minyak akan berdampak positif pada harga emas dan sebaliknya. Ketika saham mengalami kenaikan, harga emas turun. Hal ini karena ketika terjadi booming di pasar saham,

para investor cenderung menginvestasikan uang mereka di saham [10].

Emas sering menjadi pilihan investasi karena dianggap sebagai aset yang stabil dan aman. Emas memiliki nilai intrinsik yang tinggi karena langka dan sulit ditambang, serta memiliki keunikan dalam bentuknya yang tidak dapat dihasilkan oleh manusia. Selain itu, nilai emas juga cenderung naik seiring dengan melemahnya nilai mata uang [13]. Selama beberapa dekade terakhir, emas juga dianggap sebagai bentuk perlindungan terhadap inflasi dan ketidakpastian ekonomi global [14]. Oleh karena itu, emas sering dijadikan sebagai salah satu pilihan investasi dalam portofolio investor untuk mengurangi risiko dan meningkatkan keamanan investasi [15].

Pergerakan harga emas di pasar Indonesia memegang peranan penting dalam menentukan arah investasi, khususnya bagi investor yang ingin memanfaatkan peluang investasi dengan menempatkan dana pada instrumen investasi yang berisiko rendah [16]. Oleh karena itu, memprediksi pergerakan harga emas sangatlah penting untuk membantu investor dalam membuat keputusan investasi yang tepat pada waktu yang tepat [11]. Selain itu, prediksi yang akurat dapat membantu investor dalam meminimalkan risiko kerugian yang mungkin terjadi akibat fluktuasi harga emas yang cepat dan tidak dapat diprediksi secara pasti. Selain itu, memprediksi pergerakan harga emas juga dapat membantu pemerintah dalam merencanakan kebijakan ekonomi yang tepat dalam rangka mencapai tujuan stabilitas ekonomi dan pertumbuhan yang berkelanjutan [17].

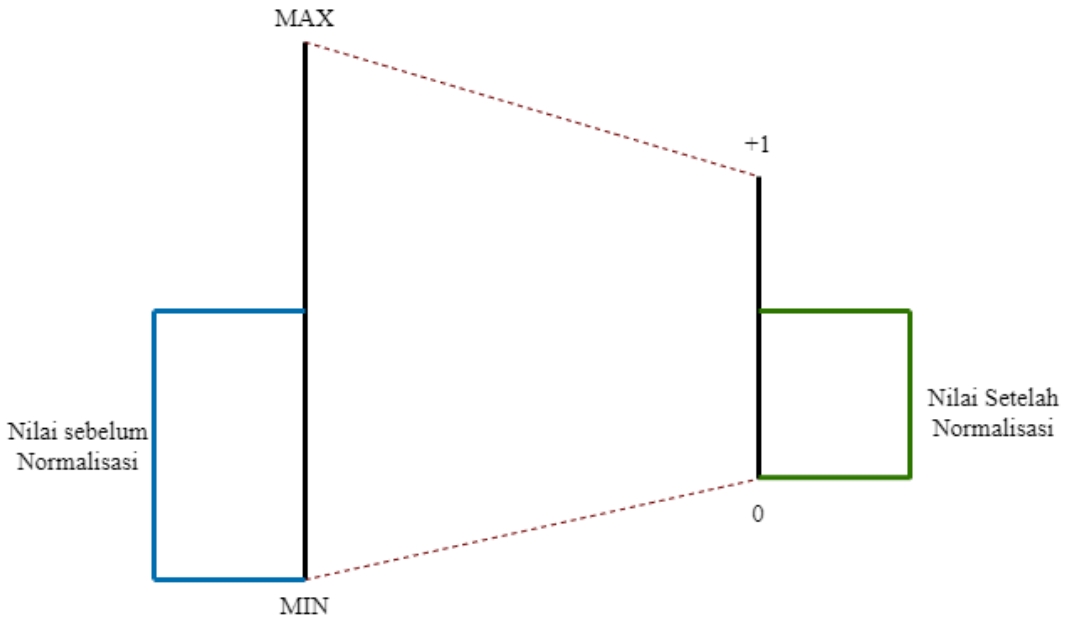
## 2.3 Normalisasi Data

Normalisasi adalah proses mengubah rentang data ke dalam rentang tertentu, seperti antara 0 dan 1 atau antara -1 dan +1. Hal ini diperlukan ketika terdapat perbedaan yang signifikan antara rentang nilai fitur-fitur yang berbeda. Metode penskalaan ini berguna ketika tidak terdapat nilai yang ekstrem dalam kumpulan data [18].

Terdapat beberapa teknik yang dapat digunakan untuk normalisasi dataset, termasuk *Min-Max normalization, z-score normalization, decimal scaling,* dan *standardized moment* [19]. Dalam penelitian ini, teknik normalisasi yang digunakan adalah *min-max normalization*.

*Min-max normalization* adalah teknik yang digunakan untuk melakukan transformasi linier pada data asli. Tujuan dari normalisasi ini adalah untuk mengubah rentang nilai data menjadi rentang yang baru, biasanya antara 0 dan 1

[20]. Gambar 2.1 merupakan visualisasi normalisasi data dengan rentang 0 dan 1.




Gambar 2.1. Normalisasi data rentang 0 dan 1
sumber : [18]

Persamaan untuk menghitung MinMax Normalization dapat dilihat pada Persamaan 2.1.

$$\text{Normalisasi}(x) = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{2.1}$$

## 2.4 Denormalisasi Data

Denormalisasi adalah proses mengubah data hasil prediksi yang berada dalam rentang interval 0 dan 1 menjadi nilai yang sesuai dengan nilai aktualnya. Tujuan denormalisasi adalah untuk membuat hasil prediksi lebih mudah dibaca dan dimengerti, serta memungkinkan perbandingan antara hasil prediksi dan nilai aktual untuk mengetahui tingkat kesalahan prediksi [21]. Persamaan yang digunakan untuk denormalisasi dapat ditemukan dalam Persamaan 2.2 [18]

$$\text{Denormalisasi}(x_{\text{norm}}) = [x_{\text{norm}} \cdot (\max(X) - \min(X))] + \min(X) \tag{2.2}$$

Di mana $x_{\text{norm}}$ adalah nilai yang telah dinormalisasi, $\max(X)$ adalah nilai maksimum dalam dataset $X$, dan $\min(X)$ adalah nilai minimum dalam dataset $X$.

## 2.5 Artificial Intelligence, Machine Learning dan Deep Learning

Artificial intelligence merupakan sebuah bidang dalam teknologi informasi yang mencakup konsep-konsep yang berkaitan dengan komputasi, penciptaan perangkat lunak, dan transmisi data. AI bertujuan untuk menciptakan mesin-mesin yang dapat berpikir dan bekerja seperti manusia, dengan kemampuan untuk mempelajari, mengambil keputusan, dan menyelesaikan tugas-tugas yang kompleks [22].

Artificial Intelligence (AI) merupakan landasan dari banyak konsep dalam bidang ilmu komputer dan teknologi. Konsep-konsep ini mencakup machine learning, deep learning, robotika, computer vision, internet, sistem rekomendasi, dan natural language processing. Konsep-konsep ini secara luas diterapkan dalam bidang ilmu dan teknologi dengan menggunakan program komputer. AI memiliki efisiensi yang tinggi karena sebagian besar tidak bergantung pada bantuan manusia. Machine learning, sebagai salah satu konsep yang berasal dari AI, bergantung pada data dan pola yang digunakan oleh sistem untuk mengambil keputusan. Begitu pula dengan robotika, konsep lain yang berasal dari AI, yang melibatkan desain dan konstruksi mesin cerdas untuk melakukan pekerjaan yang biasanya dilakukan oleh manusia. Saat ini, insinyur menggunakan robotika dan AI untuk menciptakan kendaraan mandiri di masa depan yang akan menerima perintah dari manusia dan mencegah terjadinya kecelakaan [22].

Machine learning adalah salah satu sub-bidang dari Artificial Intelligence yang populer digunakan untuk menyelesaikan berbagai permasalahan. Definisi machine learning dapat dinyatakan sebagai penerapan algoritma matematika dan teknologi komputer yang melalui tahap latihan dan pengujian, dan mampu memproses data untuk menghasilkan prediksi pada masa depan [23]. Proses pembelajaran tersebut terdiri dari dua tahap yaitu latihan (training) dan pengujian (testing) [24], yang bertujuan untuk memperoleh kecerdasan dalam menjawab berbagai masalah. Terdapat beberapa jenis machine learning, yaitu *supervised le arning*, *unsupervised learning*, dan *reinforcement learning* [25].

Deep learning adalah cabang dari Machine Learning dan Artificial Intelligence yang didasarkan pada konsep jaringan saraf buatan. Ini adalah teknologi yang mampu belajar dari data dan telah menjadi topik hangat dalam konteks komputasi. Deep learning menggunakan beberapa lapisan untuk mewakili abstraksi data guna membangun model komputasi. Ini secara luas diterapkan dalam berbagai bidang aplikasi seperti perawatan kesehatan, pengenalan visual, analisis

teks, keamanan cyber, dan banyak lagi. Namun, membangun model deep learning yang sesuai adalah tugas yang menantang, karena sifat dinamis dan variasi dalam masalah dunia nyata dan data [26].

Dalam konteks saat ini, terdapat banyak kebingungan mengenai Artificial Intelligence (Artificial Intelligence), Machine Learning (Machine Learning), dan Pembelajaran Mendalam (Deep Learning). Artificial Intelligence adalah konsep yang memungkinkan sistem komputer untuk melaksanakan tugas-tugas yang pada umumnya memerlukan kecerdasan manusia, seperti pengenalan visual, pengenalan suara, pengambilan keputusan, dan terjemahan antar bahasa. Dalam hal ini, Deep Learning merupakan bagian dari Machine Learning, dan Machine Learning sendiri merupakan bagian dari Artificial Intelligence yang merupakan istilah umum untuk program komputer yang beroperasi secara cerdas [27].

Secara sederhana, semua Machine Learning termasuk dalam Artificial Intelligence, tetapi tidak semua Artificial Intelligence adalah Machine Learning, dan seterusnya. Machine Learning merupakan perkembangan penting dalam bidang ilmu komputer, analisis data, rekayasa perangkat lunak, dan Artificial Intelligence. Machine Learning merupakan bidang penelitian yang berkembang dengan berbagai metode dan aplikasi menarik. Beberapa bidang penelitian yang menarik meliputi interpretabilitas algoritma, ketahanan, privasi, keadilan, inferensi kausalitas, interaksi manusia-mesin, dan keamanan. Tujuan utama dari Machine Learning bukanlah menghasilkan ”tebakan sempurna”, karena Machine Learning bekerja dalam domain di mana hal tersebut tidak mungkin tercapai. Tujuan utamanya adalah menghasilkan tebakan yang cukup akurat untuk memberikan manfaat [27].

Deep Learning merupakan jenis Machine Learning tertentu yang memiliki kemampuan dan fleksibilitas luar biasa dalam mengenali dunia melalui representasi konsep yang tersusun dalam hirarki bertingkat. Setiap konsep dihubungkan dengan konsep yang lebih sederhana, dan representasi yang lebih abstrak dihitung berdasarkan representasi yang kurang abstrak. Dalam skripsi ini, akan disajikan gambaran secara komprehensif tentang Artificial Intelligence, Machine Learning, dan teknik Deep Learning, serta dibandingkan dengan teknik-teknik lainnya [27].

## 2.6 Artificial Neural Networks

Artificial Neural Networks (ANNs) merupakan salah satu teknik machine learning yang cukup populer dalam bidang pengolahan data dan Artificial

Intelligence. ANN merupakan suatu sistem pemrosesan informasi yang terdiri dari banyak unit pemrosesan sederhana yang saling terhubung dan bekerja secara paralel. Dalam ANN, informasi diteruskan dari input layer menuju hidden layer dan kemudian ke output layer melalui serangkaian koneksi atau bobot yang telah ditentukan. Proses ini terinspirasi dari cara kerja jaringan saraf biologis manusia, di mana neuron menerima input, memproses informasi tersebut, dan mengirimkan output ke neuron lainnya [28].

ANN memiliki kemampuan untuk mempelajari pola-pola yang ada dalam data, sehingga dapat digunakan untuk melakukan klasifikasi, prediksi, dan optimasi. Namun, ANN juga memiliki beberapa kelemahan seperti kecenderungan untuk overfitting dan waktu komputasi yang lama pada model yang kompleks. Untuk mengatasi masalah tersebut, banyak penelitian telah dilakukan untuk meningkatkan kinerja ANN, seperti menggunakan algoritma pelatihan yang lebih efisien dan penggunaan model jaringan yang lebih kompleks seperti *Convolutional Neural Networks* (CNNs) dan *Recurrent Neural Networks* (RNNs) [28].

ANN adalah model komputasi yang dikembangkan berdasarkan jaringan saraf biologis. ANN terdiri dari neuron buatan yang saling terhubung. Kemampuan pemrosesan jaringan disimpan dalam kekuatan koneksi antar-neuron yang disebut bobot (*Weights*). Bobot-bobot ini diperoleh melalui pembelajaran atau adaptasi dari sejumlah pola pelatihan. Biasanya, ANN mengadaptasi strukturnya berdasarkan informasi yang diterimanya. Ada sejumlah langkah sistematis yang disebut aturan pembelajaran yang perlu diikuti saat mengembangkan ANN. Selain itu, proses pembelajaran membutuhkan data pembelajaran untuk menemukan titik operasi terbaik dari ANN. ANN dapat digunakan untuk mempelajari fungsi aproksimasi dari beberapa data yang diamati [29].

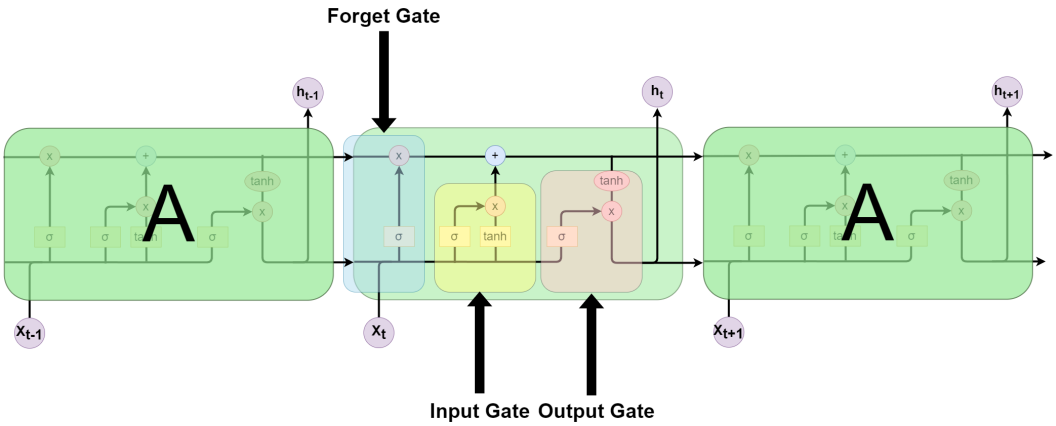
Dalam konteks Jaringan Saraf Tiruan (Artificial Neural Networks/ANN), Parameter tambahan yang ditambahkan ke setiap neuron untuk mempengaruhi output neuron tersebut disebut bias ($b$) [30]. Bias digunakan untuk mengontrol sejauh mana aktivasi neuron terhadap input yang diberikan, yang membantu dalam pembentukan pola pembelajaran dan meningkatkan kemampuan jaringan saraf untuk menangkap relasi yang kompleks dalam data [31]. Dengan adanya bias, ANN dapat mengatasi masalah linearitas dan meningkatkan kapasitas adaptasi serta kemampuan generalisasi jaringan saraf. Dengan demikian, bias memiliki peran penting dalam meningkatkan performa dan kinerja jaringan saraf dalam pemodelan dan pengambilan keputusan.

## 2.7 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) adalah salah satu jenis model jaringan saraf tiruan yang digunakan untuk memproses urutan data yang panjang, seperti urutan kata dalam kalimat atau urutan waktu dalam rangkaian waktu [8]. LSTM dirancang untuk mengatasi masalah *vanishing gradient* [32] pada model jaringan saraf tiruan tradisional, yang terjadi ketika gradien penyesuaian model menjadi sangat kecil sehingga tidak lagi mampu memperbarui parameter model yang diperlukan untuk pembelajaran yang efektif.

LSTM memiliki arsitektur yang kompleks, terdiri dari tiga gerbang utama: gerbang input (*input gate*), gerbang lupa (*forget gate*), dan gerbang keluaran (*output gate*) [33]. Ketiga gerbang ini bekerja bersama untuk mengatur aliran informasi melalui sel memori LSTM. Sel memori LSTM adalah unit dasar dalam LSTM, yang bertanggung jawab untuk menyimpan dan memodifikasi informasi selama proses pembelajaran.

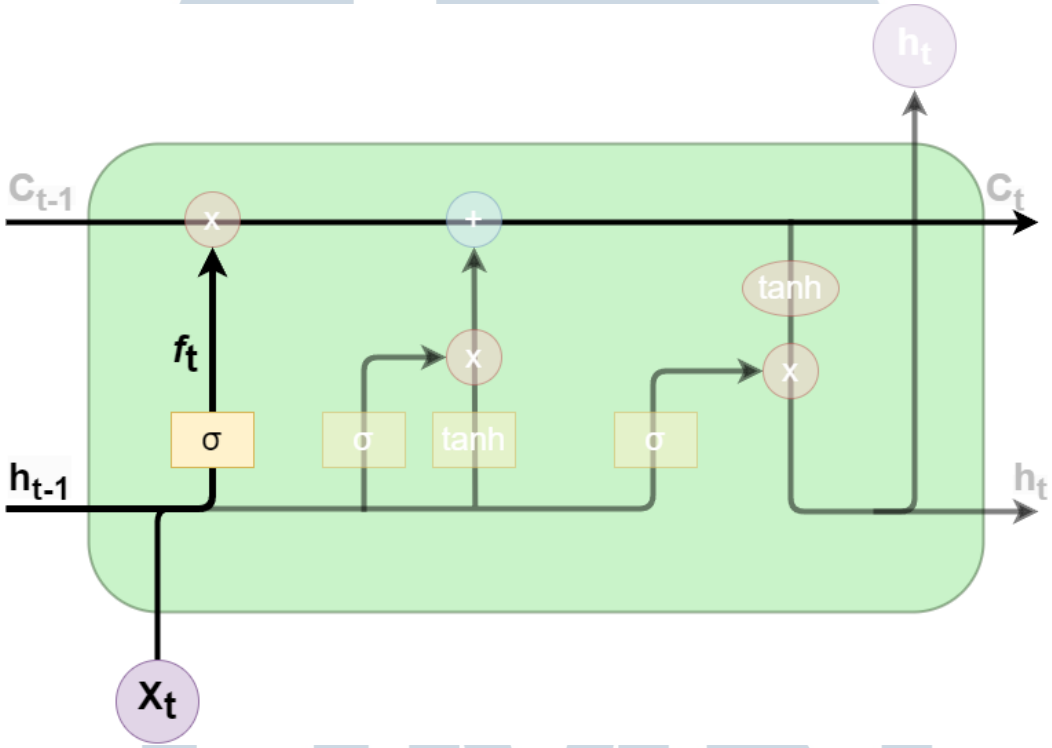
Gerbang input (*input gate*) digunakan untuk mengontrol aliran informasi baru ke dalam sel memori LSTM. Sedangkan, gerbang lupa (*forget gate*) digunakan untuk mengatur aliran informasi yang akan dihapus dari sel memori LSTM. Gerbang keluaran (*output gate*) mengatur aliran informasi yang akan digunakan sebagai output dari sel memori LSTM [34]. Gambar 2.2 menunjukkan arsitektur LSTM.




Gambar 2.2. Arsitektur LSTM
sumber: [34]

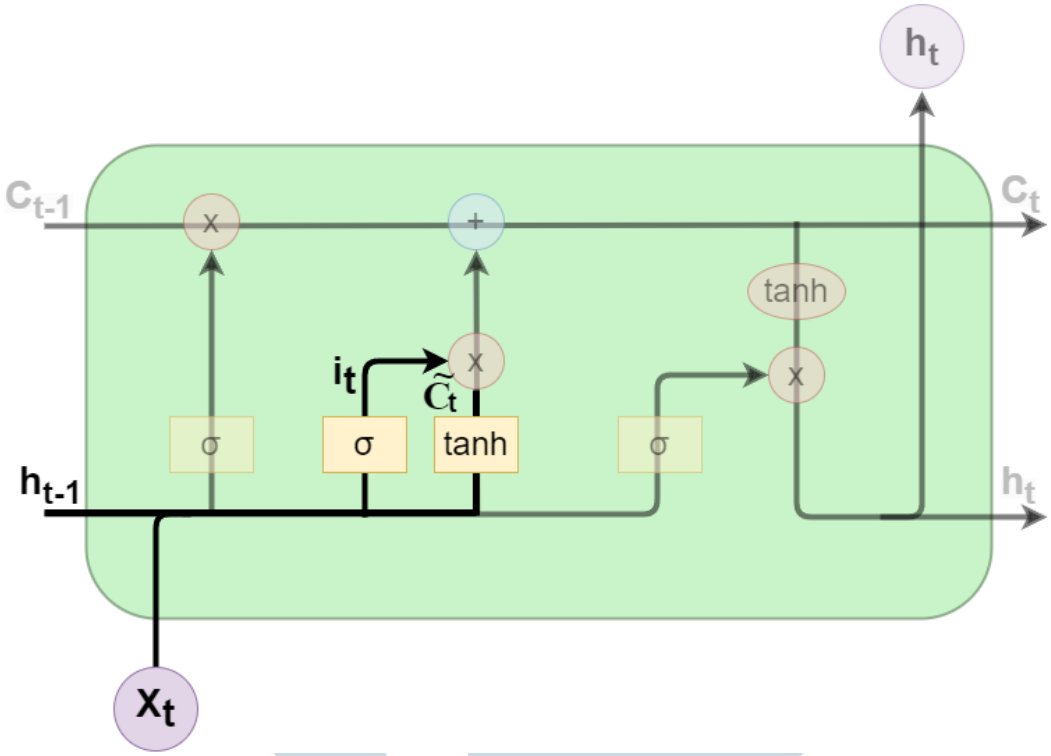
Langkah pertama dalam arsitektur Long Short-Term Memory (LSTM) adalah menentukan informasi yang harus disimpan atau diabaikan dalam sel. Hal ini dilakukan oleh sebuah bagian yang disebut Forget Gate yang berfungsi untuk

membuat keputusan tentang informasi apa yang harus diabaikan dari sel. Forget Gate akan memperhatikan nilai output sebelumnya ($h_{t-1}$) dan nilai masukan saat ini ($x_t$), yang kemudian menghasilkan keluaran 0 atau 1. Keluaran 0 menunjukkan bahwa informasi harus diabaikan, sedangkan keluaran 1 menunjukkan bahwa informasi harus disimpan dalam sel [34]. Ilustrasi dari langkah ini dapat dilihat pada Gambar 2.3.



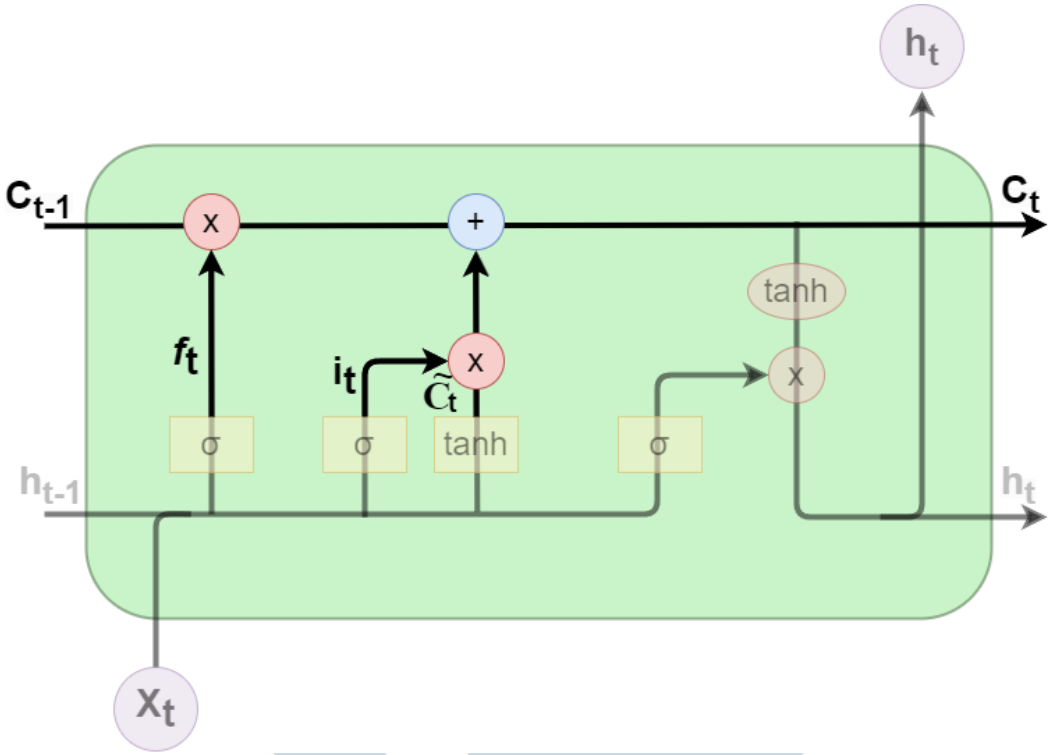

Gambar 2.3. Struktur *forget gate*
sumber: [34]

Tahap berikutnya dalam proses LSTM adalah menentukan informasi baru yang akan disimpan dalam sel. Proses ini terdiri dari dua tahapan. Tahap pertama adalah layer sigmoid yang disebut "*input gate layer*" yang akan menentukan nilai mana yang perlu diperbarui. Selanjutnya, sebuah layer tanh akan membuat vektor sebagai nilai kandidat baru ($\tilde{C}_t$) yang dapat ditambahkan ke state [34]. Kedua tahapan ini akan digabungkan untuk memperbarui state, seperti yang ditunjukkan pada Gambar 2.4.



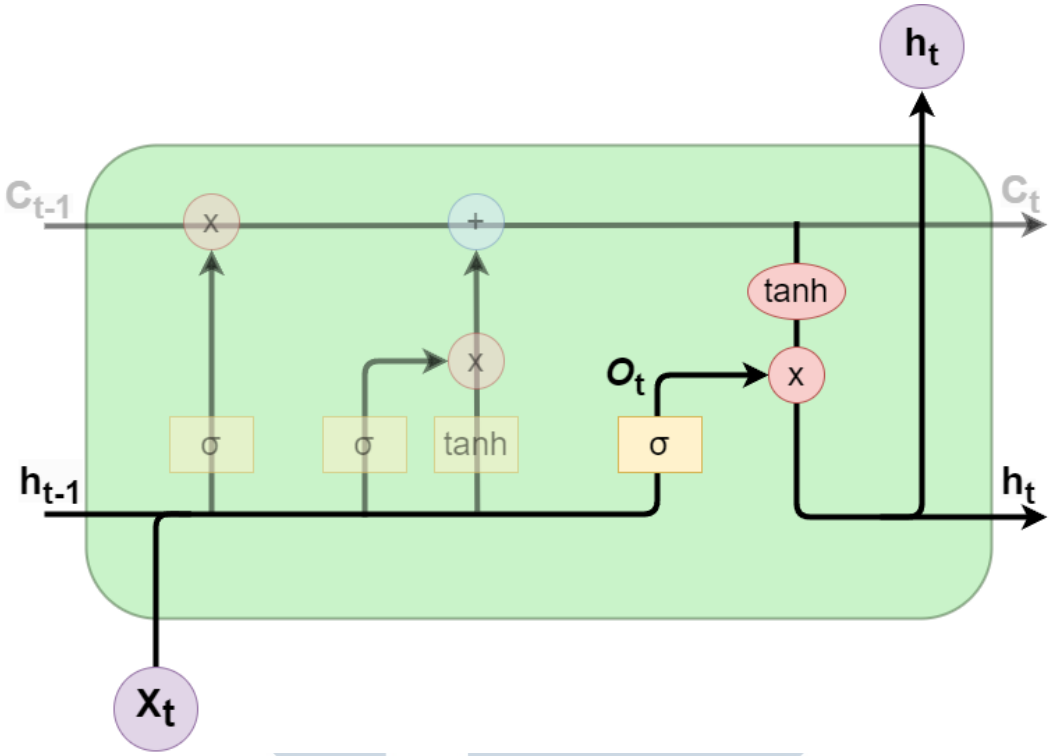

Gambar 2.4. Struktur *input gate*
sumber: [34]

Selanjutnya, state lama ($C_{t-1}$) akan diperbarui menjadi state baru ($C_t$). Pembaruan ini dilakukan dengan mengalikan state lama dengan $f_t$, yang mengabaikan informasi yang telah dilupakan sebelumnya. Selanjutnya, keluaran dari $i_t$ akan ditambahkan ke state yang diperbarui ($C_t$) [34]. Pembaruan state ini dilakukan melalui jaringan saraf yang relevan seperti yang terlihat pada Gambar 2.5.




Gambar 2.5. Pembaharuan struktur *input gate*
sumber: [34]

Tahapan akhir adalah menentukan hasil keluaran. Tahap pertama mencakup pengoperasian layer sigmoid terhadap sel yang ingin dihasilkan. Setelah itu, sel tersebut akan melewati layer tanh yang berfungsi untuk menyesuaikan nilai pada rentang antara -1 hingga 1, dan akan dikalikan dengan keluaran dari pintu gerbang sigmoid [34], seperti yang dapat dilihat pada Gambar 2.6.




Gambar 2.6. Struktur *output gate*
sumber: [34]

Secara matematis, LSTM dapat dirumuskan sebagai berikut [34]:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.3}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.4}$$

$$\tilde{C}t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.5}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}t \tag{2.6}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.7}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{2.8}$$

Tabel 2.1 berisi parameter arsitektur LSTM. Parameter-parameter ini menggunakan simbol-simbol rumus dalam LSTM beserta keterangannya. Tabel ini memberikan gambaran tentang simbol-simbol yang digunakan dalam persamaan LSTM dan penjelasan singkat tentang masing-masing simbol.

Tabel 2.1. Parameter arsitektur LSTM

| Simbol | Keterangan |
|---|---|
| $x_t$ | Input pada waktu t |
| $b$ | Bias |
| $c_t$ | Unit memori aktual pada waktu t |
| $h_t$ | Keluaran sel memori saat ini |
| $c_{t-1}$ | Unit memori pada waktu sebelumnya (t-1) |
| $h_{t-1}$ | Keluaran sel memori sebelumnya |
| $f, i, c, o$ | Status forget, input, cell, output |
| $W_{f,i,c,o}$ | Bobot yang diterapkan pada input f,i,c,o |

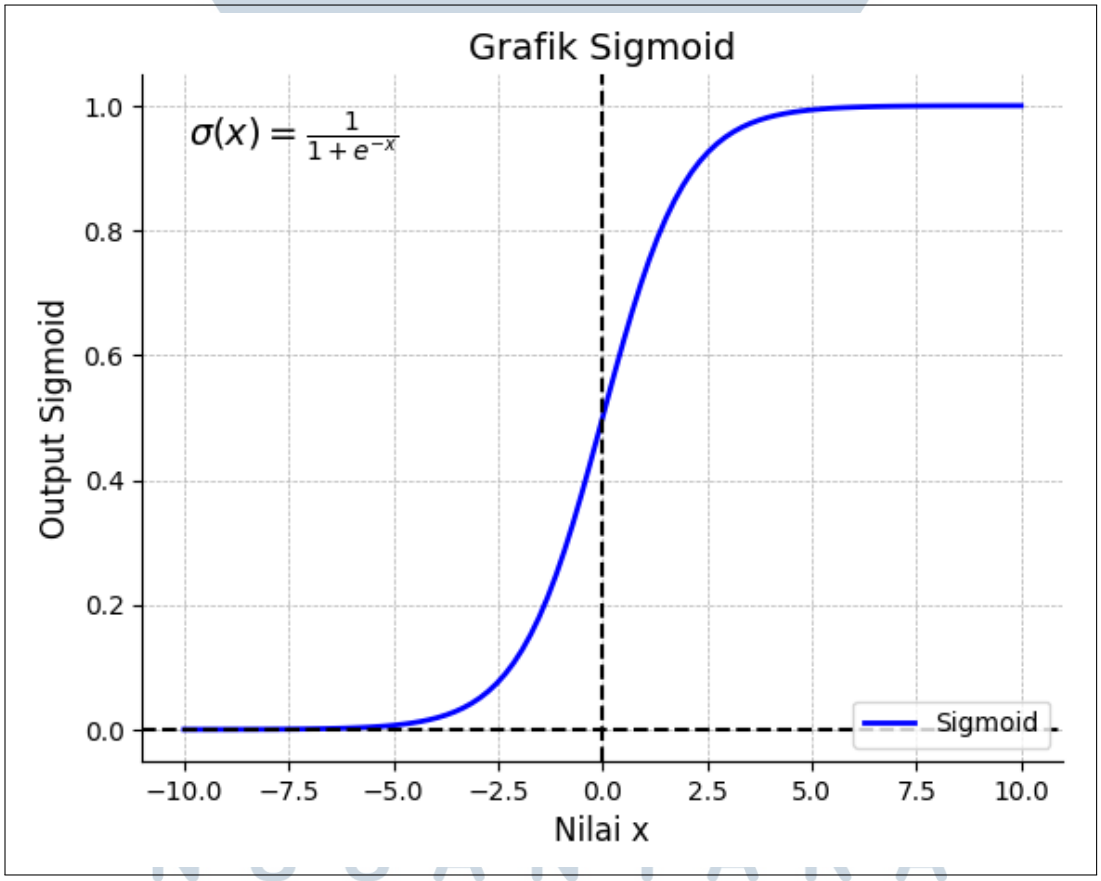
Tabel 2.1 menyajikan parameter-parameter dalam arsitektur LSTM. Model ini terdiri dari empat status dalam LSTM yang masing-masing bertanggung jawab dalam mengatur aliran informasi: gerbang lupa (forget gate), gerbang input (input gate), gerbang keluaran (output gate), dan sel memori baru. Pada setiap langkah waktu t, input saat ini $X_t$ adalah vektor 1-D dengan ukuran 1xm yang mengandung m fitur yang akan dilatih, dan keluaran memiliki dimensi 1xn. Dalam struktur konvensional LSTM, bias yang digunakan dalam model ini memiliki angka acak yang tetap. Parameter-parameter dalam model ini termasuk matriks bobot $W_f, W_i, W_c$, dan $W_o$ yang menghubungkan input dengan masing-masing gerbang dan unit memori baru. Pada awalnya, bias dari status-status $f, i, c, o$ dibuat sama, tetapi selama proses pelatihan, mereka dapat disesuaikan secara independen melalui penyesuaian bobot. Dengan demikian, bias dapat disesuaikan secara detail untuk meningkatkan akurasi output gerbang dan mengoptimalkan gradien yang digunakan dalam proses back-propagation. Oleh karena itu, vektor input keseluruhan adalah $[X_t, b]$ dengan dimensi 1x(m+1), dan matriks keluaran yang sesuai memiliki ukuran k x n, di mana k adalah panjang urutan (sequence length) yang ditentukan dalam pelatihan LSTM [35].

### 2.7.1 Sigmoid Activation Function

Fungsi Sigmoid merupakan fungsi non-linear yang umumnya digunakan dalam *feed forward neural networks*. Fungsi ini memiliki batasan nilai, dapat dihitung, dan didefinisikan untuk nilai input nyata. Selain itu, fungsi ini memiliki turunan positif di seluruh nilai input dan tingkat kehalusan tertentu [36]. Fungsi sigmoid dapat dinyatakan melalui Persamaan 2.9.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.9}$$

Fungsi sigmoid umumnya digunakan di lapisan output dalam arsitektur Deep Learning (DL) untuk memprediksi output berbasis probabilitas. Fungsi ini telah berhasil diterapkan dalam masalah klasifikasi biner, pemodelan tugas regresi logistik, dan domain jaringan saraf lainnya. Keunggulan utama dari fungsi sigmoid, seperti yang diungkapkan oleh Neal [37], adalah kemudahan pemahamannya dan penggunaannya yang umumnya terbatas pada jaringan dangkal. Namun demikian, Glorot dan Bengio [38] merekomendasikan untuk menghindari penggunaan fungsi sigmoid saat menginisialisasi jaringan saraf dengan bobot acak kecil, karena dapat mengakibatkan dinamika pembelajaran yang buruk, dengan mencapai titik jenuh pada lapisan tersembunyi teratas pada tahap awal. Gambar 2.7 merupakan representasi grafis dari respons fungsi aktivasi sigmoid.



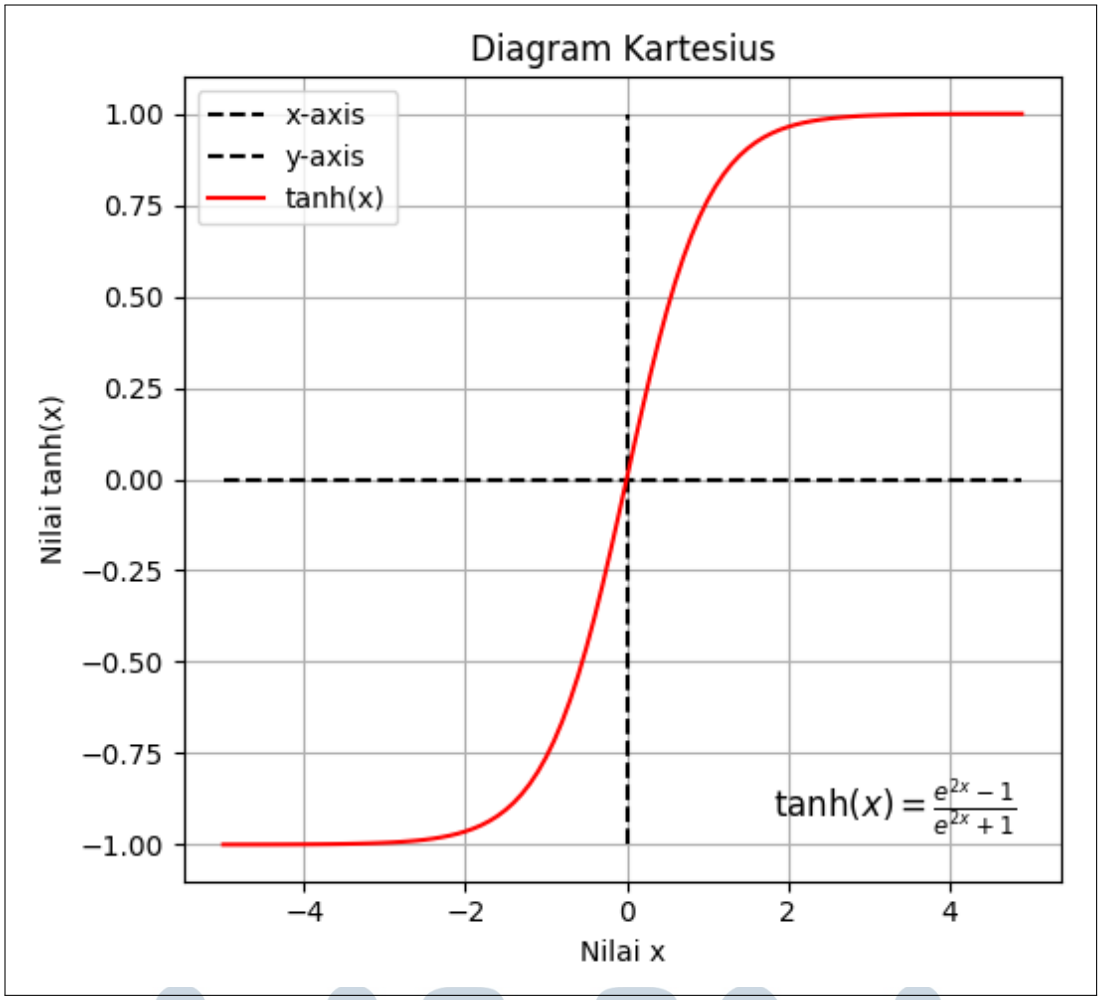

Gambar 2.7. Representasi grafis dari respons fungsi aktivasi sigmoid

### 2.7.2 Hyperbolic Tangent Function

Fungsi tangen hiperbolik adalah jenis fungsi aktivasi yang digunakan dalam deep learning, dan memiliki beberapa varian yang digunakan dalam aplikasi deep learning. Fungsi tangen hiperbolik, yang juga dikenal sebagai fungsi tangen hiperbolik, adalah fungsi yang lebih halus dan berpusat di nol, dengan rentang nilai antara -1 hingga 1 [39]. Fungsi tangen hiperbolik dapat dinyatakan melalui Persamaan 2.10.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.10}$$

Fungsi tangen hiperbolik menjadi pilihan yang lebih disukai dibandingkan dengan fungsi sigmoid dalam konteks jaringan saraf multi-lapisan, karena memberikan kinerja pelatihan yang lebih baik [37, 40]. Namun, fungsi tangen hiperbolik juga tidak dapat mengatasi sepenuhnya masalah *vanishing gradient* yang terjadi pada fungsi sigmoid. Kelebihan utama dari fungsi ini adalah menghasilkan keluaran dengan pusat nol, sehingga mendukung proses *back-propagation* dalam jaringan saraf. Gambar 2.8 merupakan representasi grafis dari respons fungsi tangen hiperbolik.

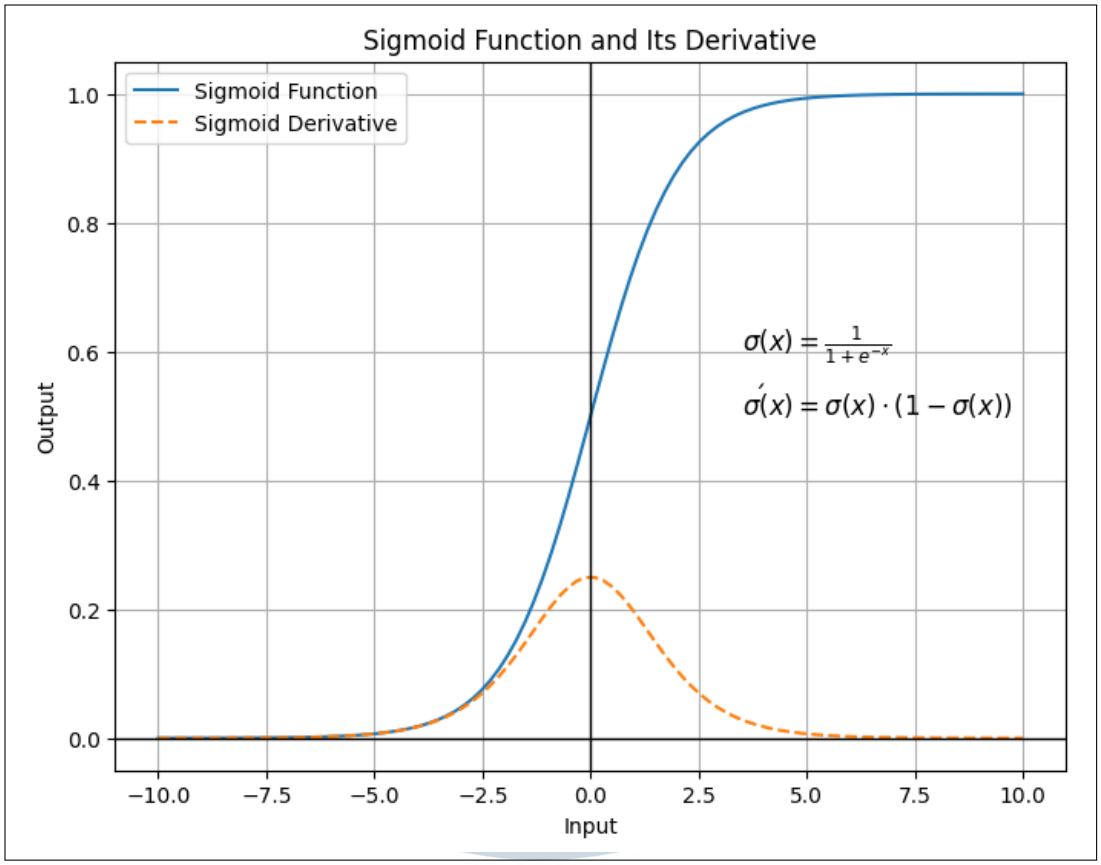



Gambar 2.8. Representasi grafis dari respons fungsi aktivasi tangen hiperbolik

### 2.7.3 Vanishing Gradient

*Vanishing gradient* mengacu pada fenomena yang berlawanan, di mana komponen-komponen jangka panjang secara eksponensial menurun hingga mencapai nilai nol, yang membuat model menjadi tidak mampu mempelajari hubungan antara peristiwa-peristiwa temporal yang terpisah secara signifikan [41].

Masalah *vanishing gradient* terjadi ketika perubahan gradien yang dikirim kembali ke lapisan awal model menjadi semakin kecil seiring dengan peningkatan kedalaman jaringan. Fenomena ini mengakibatkan penurunan kualitas model pelatihan karena informasi yang terkandung dalam fitur-fitur data menjadi kurang efektif dalam mengarahkan pembaruan bobot dan bias pada lapisan awal model. Hal ini dapat menghambat kemampuan model dalam mengekstraksi dan merepresentasikan fitur yang penting dalam data yang dihadapinya [42].




Gambar 2.9. Fungsi simoid dan turunannya

Sebagai contoh, pada Gambar 2.9 terdapat fungsi sigmoid dan turunannya. Dapat diamati bahwa ketika input fungsi sigmoid menjadi lebih besar atau lebih kecil (ketika nilai absolut dari $x$ semakin besar), turunannya mendekati nol. Hal ini memiliki signifikansi yang penting. Pada jaringan neural dengan sedikit lapisan yang menggunakan aktivasi ini, hal ini bukan merupakan masalah yang signifikan. Namun, ketika jaringan menggunakan lebih banyak lapisan, hal ini dapat menyebabkan gradien menjadi terlalu kecil sehingga pelatihan menjadi tidak efektif [43].

Gradien pada jaringan saraf ditemukan menggunakan metode *backpropagation*. Secara sederhana, *backpropagation* mencari turunan dari jaringan dengan bergerak dari lapisan terakhir ke lapisan awal. Dengan aturan rantai, turunan dari setiap lapisan dikalikan secara berurutan dari lapisan terakhir hingga lapisan awal untuk menghitung turunan lapisan awal. Namun, ketika terdapat $n$ lapisan tersembunyi yang menggunakan aktivasi seperti fungsi sigmoid, hasil perkalian dari $n$ turunan kecil tersebut akan menurun secara eksponensial saat propagasi dilakukan hingga mencapai lapisan awal. Gradien yang kecil

menunjukkan bahwa bobot dan bias pada lapisan awal tidak akan diperbarui secara efektif setiap kali sesi pelatihan dilakukan. Karena lapisan awal ini memiliki peran penting dalam mengenali elemen inti dari data masukan, hal ini dapat mengakibatkan ketidakakuratan secara keseluruhan pada jaringan [43].

### 2.7.4 Hyperparameter pada LSTM

Arsitektur LSTM (Long Short-Term Memory) adalah jenis arsitektur jaringan saraf rekuren (RNN) yang dirancang khusus untuk mengatasi masalah *vanishing gradient* [32] yang sering terjadi pada RNN tradisional. Arsitektur ini terdiri dari unit sel LSTM yang memiliki komponen utama, yaitu *forget gate*, *input gate*, *output gate* [33]. Gerbang input (*input gate*) digunakan untuk mengontrol aliran informasi baru ke dalam sel memori LSTM. Sedangkan, gerbang lupa (*forget gate*) digunakan untuk mengatur aliran informasi yang akan dihapus dari sel memori LSTM. Gerbang keluaran (*output gate*) mengatur aliran informasi yang akan digunakan sebagai output dari sel memori LSTM [34].

Selain itu, LSTM memiliki berbagai *hyperparameter* yang dapat dikonfigurasi secara fleksibel. Memilih *hyperparameter* yang tepat sangat penting karena secara langsung mempengaruhi kinerja model. Namun, mencari *hyperparameter* secara acak dan berulang-ulang adalah sebuah proses yang memakan waktu dan tidak dapat diandalkan. Teknik uji coba dan kesalahan semacam ini melambatkan seluruh proses Machine Learning [44].

Algoritma penyetelan *hyperparameter* biasanya bekerja dengan menyeimbangkan dua faktor utama yang mempengaruhi kekuatan optimasi algoritma. Faktor pertama adalah kemampuan algoritma untuk mengeksplorasi berbagai konfigurasi *hyperparameter* yang berbeda. Faktor kedua adalah kemampuan algoritma untuk memanfaatkan sumber daya secara efisien dan memperoleh hasil yang optimal [44]. Berikut adalah beberapa hyperparameter yang umumnya diuji dalam model LSTM :

1. Jumlah layer LSTM, terdiri dari blok-blok memori yang saling terhubung secara berulang. Setiap blok memori terdiri dari satu sel memori yang mengandung tiga gerbang perkalian [45]. Lapisan LSTM bertanggung jawab untuk menentukan informasi mana yang sebaiknya disimpan dalam status sel jaringan ($c_t$). Hal ini melibatkan pemilihan nilai kandidat yang memiliki potensi untuk ditambahkan ke status sel, serta nilai aktivasi ($i_t$) dari gerbang masukan [46].

2. Jumlah unit LSTM, unit dalam arsitektur LSTM disebut sebagai sel. Sel-sel tersebut menerima kombinasi antara keadaan sebelumnya dan input saat ini sebagai inputnya. Dalam konteks penelitian ini, sel-sel tersebut memainkan peran penting dalam memutuskan informasi apa yang akan disimpan dalam memori dan informasi apa yang akan dieliminasi [47]. Peningkatan jumlah sel LSTM secara bertahap pada awalnya meningkatkan kinerja model. Hal ini disebabkan oleh kemampuan model yang semakin baik dalam memahami dan mengadaptasi pola-pola dalam data. Namun, perlu diingat bahwa setelah mencapai suatu titik, peningkatan lebih lanjut dalam jumlah sel LSTM justru dapat menyebabkan overfitting, di mana model terlalu memfokuskan diri pada data pelatihan secara spesifik dan tidak mampu menggeneralisasi dengan baik pada data yang baru [48].

3. Jumlah layer Dense, lapisan yang memiliki koneksi yang kuat dengan lapisan sebelumnya, di mana setiap neuron dalam lapisan tersebut terhubung dengan setiap neuron dari lapisan sebelumnya [49]. Lapisan Dense bertugas untuk mengubah output dari lapisan sebelumnya menjadi nilai prediksi [21].

4. Fungsi aktivasi, memilih fungsi aktivasi yang tepat untuk layer LSTM dan output layer. Fungsi aktivasi yang sesuai dapat membantu model menggambarkan perubahan dengan lebih baik [50].

5. Optimizer, menentukan algoritma optimasi yang digunakan untuk melatih model. Berbagai optimizer seperti Adam, RMSprop, atau SGD memiliki pengaturan hyperparameter yang perlu dituning, seperti learning rate, momentum, atau decay rate [51].

6. Learning rate, mengontrol seberapa cepat model beradaptasi dengan masalah. Learning rate yang lebih kecil memerlukan lebih banyak epoch pelatihan karena perubahan yang lebih kecil pada bobot setiap pembaruan, sedangkan learning rate yang lebih besar menghasilkan perubahan yang cepat dan memerlukan lebih sedikit epoch pelatihan. Learning rate yang terlalu besar dapat menyebabkan model cepat konvergen ke solusi yang kurang optimal, sedangkan learning rate yang terlalu kecil dapat menyebabkan proses model tidak dapat mencapai solusi yang lebih baik atau memperbaiki kinerjanya karena perubahan yang terlalu kecil pada bobotnya [52].

7. Batch size, menunjukkan jumlah sampel data yang diproses dalam satu iterasi saat melatih model. Saat melatih model dengan dataset yang besar,

penggunaan batch size memungkinkan pemrosesan yang lebih efisien dan mengurangi penggunaan memori. Nilai batch size yang lebih kecil dapat menghasilkan pembelajaran yang lebih stabil, tetapi membutuhkan lebih banyak waktu untuk melatih model karena diperlukan lebih banyak iterasi. Sebaliknya, nilai batch size yang lebih besar dapat mempercepat waktu pelatihan, tetapi dapat menghasilkan pembelajaran yang lebih tidak stabil [53].

8. Epoch, menunjukkan seberapa sering dataset diberikan ke jaringan saat melatihnya. Dalam algoritma deep learning, perlu dilakukan pembaruan pada bobot dan menjalankan seluruh dataset beberapa kali agar model prediksi dapat dioptimalkan secara lebih baik dan lebih akurat melalui penurunan gradien. Namun, tidak jelas berapa jumlah epoch yang diperlukan untuk melatih model dengan dataset yang sama hingga mencapai bobot optimal. Setiap dataset yang berbeda dapat diproses dengan cara yang berbeda oleh jaringan, sehingga mempengaruhi jumlah epoch yang diperlukan untuk mencapai hasil terbaik [54].

### 2.7.5 Implementasi Rumus LSTM secara Matematis

Rumus LSTM terdiri dari beberapa persamaan yang menggambarkan bagaimana LSTM menghitung informasi pada setiap time-step. Pada setiap time-step, LSTM menerima input $x_t$ dan menghasilkan output $h_t$ serta menghitung cell state $C_t$ yang menyimpan informasi yang relevan dari input sebelumnya. Ada tiga gerbang (gate) pada LSTM yaitu forget gate $f_t$, input gate $i_t$, dan output gate $o_t$. Setiap gerbang menghasilkan nilai antara 0 dan 1 yang menentukan seberapa banyak informasi yang akan dilewatkan atau disimpan pada cell state [34]. Berikut ini diberikan contoh perhitungan yang dilakukan dengan menggunakan matriks input:

$$x_t = \begin{bmatrix} 0.8 & 0.4 & 0.1 \end{bmatrix}, \quad h_{t-1} = \begin{bmatrix} 0.5 & -0.2 & 0.3 \end{bmatrix}, \quad C_{t-1} = \begin{bmatrix} 0.4 & -0.1 & 0.2 \end{bmatrix}$$

1. Menginisialisasi nilai bobot dan bias untuk setiap gerbang LSTM
   Pada tahap ini, nilai bobot ($W$) dan bias ($b$) diinisialisasi untuk setiap gerbang dalam jaringan LSTM. Bobot dan bias ini akan digunakan dalam perhitungan rumus gerbang LSTM. Dalam contoh studi kasus ini, contoh nilai bobot dan

bias diberikan untuk setiap gerbang LSTM, termasuk Forget Gate ($W_f$ dan $b_f$), Input Gate ($W_i$ dan $b_i$), Output Gate ($W_o$ dan $b_o$), serta Sel Memori Baru ($W_c$ dan $b_c$). Bobot dan bias untuk setiap gerbang LSTM diinisialisasi sebagai berikut:

$$W_f = \begin{bmatrix} 0.1 & 0.2 & -0.1 & 0.3 & -0.2 & 0.1 \\ -0.3 & 0.1 & 0.2 & 0.2 & -0.1 & -0.2 \\ 0.2 & 0.1 & -0.2 & -0.1 & 0.3 & 0.2 \end{bmatrix}, \quad b_f = \begin{bmatrix} 0.2 & 0.1 & 0.3 \end{bmatrix}$$

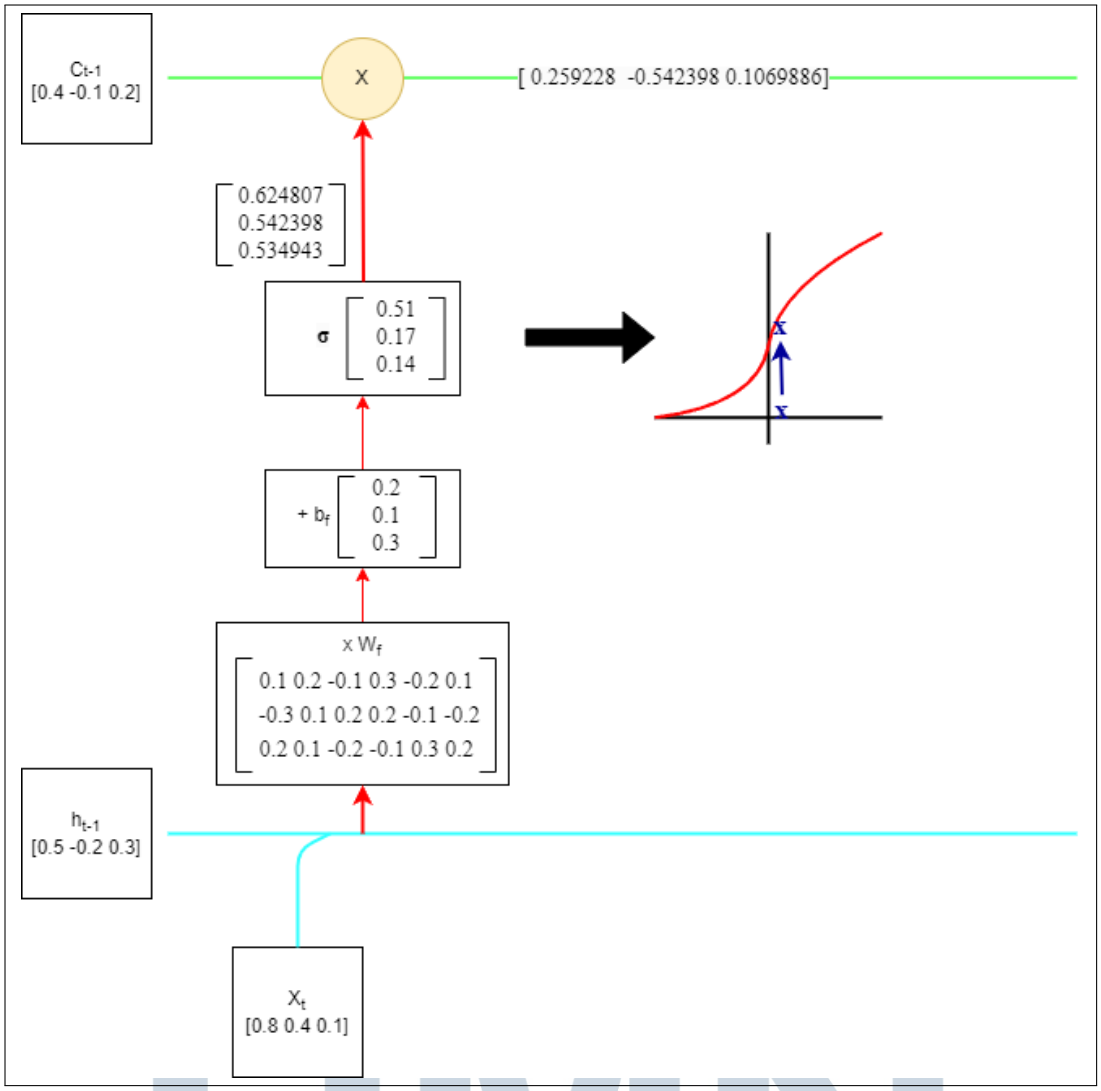
$$W_i = \begin{bmatrix} -0.2 & 0.1 & -0.3 & 0.2 & -0.2 & 0.1 \\ 0.1 & -0.2 & 0.1 & -0.1 & -0.2 & 0.3 \\ -0.1 & -0.3 & 0.2 & 0.3 & -0.1 & -0.2 \end{bmatrix}, \quad b_i = \begin{bmatrix} 0.2 & 0.3 & -0.1 \end{bmatrix}$$

$$W_c = \begin{bmatrix} 0.3 & -0.2 & 0.1 & 0.1 & -0.2 & 0.3 \\ 0.2 & -0.1 & -0.2 & -0.2 & 0.3 & 0.1 \\ -0.1 & 0.3 & 0.2 & 0.3 & -0.1 & -0.2 \end{bmatrix}, \quad b_c = \begin{bmatrix} 0.1 & 0.3 & -0.2 \end{bmatrix}$$

$$W_o = \begin{bmatrix} -0.1 & -0.2 & 0.3 & -0.2 & 0.3 & -0.1 \\ -0.2 & 0.3 & 0.1 & 0.1 & -0.2 & 0.3 \\ 0.3 & -0.1 & -0.2 & 0.2 & -0.3 & 0.1 \end{bmatrix}, \quad b_o = \begin{bmatrix} 0.3 & -0.2 & 0.1 \end{bmatrix}$$

2. Forget Gate ($f_t$)

   Forget Gate digunakan untuk mengatur seberapa banyak informasi sebelumnya yang akan dilupakan dalam Sel Memori Terkini ($C_t$). Rumus yang digunakan adalah sigmoid dari hasil perkalian matriks input ($h_{t-1}$ dan $x_t$) dengan bobot ($W_f$), ditambah dengan bias ($b_f$). Hasil perhitungan ini menghasilkan matriks $f_t$, yang menunjukkan seberapa banyak informasi yang akan diabaikan. Gambar 2.10 merupakan representasi model LSTM pada forget gate.




Gambar 2.10. Representasi model forget gate LSTM

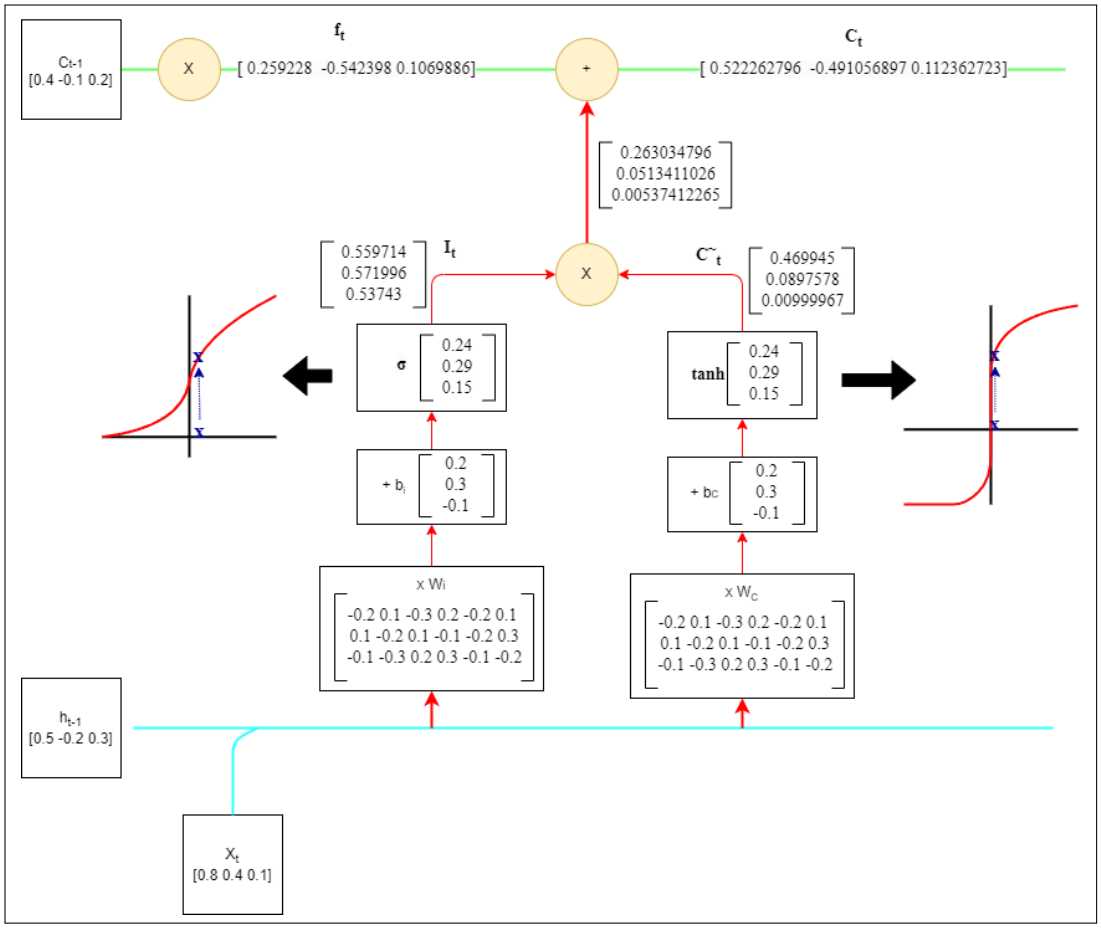
Dalam proses perhitungan, nilai forget gate LSTM diperoleh dengan

menggunakan metode berikut:

$$
\begin{aligned}
f_t &= \sigma\left(\begin{bmatrix} 0.1 & 0.2 & -0.1 & 0.3 & -0.2 & 0.1 \\ -0.3 & 0.1 & 0.2 & 0.2 & -0.1 & -0.2 \\ 0.2 & 0.1 & -0.2 & -0.1 & 0.3 & 0.2 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ -0.2 \\ 0.3 \\ 0.8 \\ -0.4 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.1 \\ 0.3 \end{bmatrix}\right) \\
&= \sigma\left(\begin{bmatrix} 0.31 \\ 0.07 \\ -0.16 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.1 \\ 0.3 \end{bmatrix}\right) \\
&= \sigma\left(\begin{bmatrix} 0.51 \\ 0.17 \\ 0.14 \end{bmatrix}\right) \\
&\approx \begin{bmatrix} 0.624807 \\ 0.542398 \\ 0.534943 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}
\end{aligned}
$$

3. Input Gate ($i_t$)

   Input Gate bertanggung jawab untuk menentukan seberapa banyak informasi baru yang akan disimpan dalam Sel Memori Terkini ($C_t$). Rumusnya sama dengan Forget Gate, yaitu sigmoid dari hasil perkalian matriks input ($h_{t-1}$ dan $x_t$) dengan bobot ($W_i$), ditambah dengan bias ($b_i$). Hasil perhitungan ini menghasilkan matriks $i_t$, yang menunjukkan seberapa banyak informasi baru yang akan disimpan. Gambar 2.10 merupakan representasi model LSTM pada input gate.




Gambar 2.11. Representasi model input gate dan sel memori LSTM

Dalam proses perhitungan, nilai input gate LSTM diperoleh dengan

menggunakan metode berikut:

$$
\begin{aligned}
i_t &= \sigma\left(\begin{bmatrix} -0.2 & 0.1 & -0.3 & 0.2 & -0.2 & 0.1 \\ 0.1 & -0.2 & 0.1 & -0.1 & -0.2 & 0.3 \\ -0.1 & -0.3 & 0.2 & 0.3 & -0.1 & -0.2 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ -0.2 \\ 0.3 \\ 0.8 \\ -0.4 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.3 \\ -0.1 \end{bmatrix}\right) \\
&= \sigma\left(\begin{bmatrix} 0.04 \\ -0.01 \\ 0.25 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.3 \\ -0.1 \end{bmatrix}\right) \\
&= \sigma\left(\begin{bmatrix} 0.24 \\ 0.29 \\ 0.15 \end{bmatrix}\right) \\
&\approx \begin{bmatrix} 0.559714 \\ 0.571996 \\ 0.53743 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}
\end{aligned}
$$

4. Menghitung nilai Sel Memori Baru ($\tilde{C}t$)

   Sel Memori Baru ($\tilde{C}t$) adalah informasi baru yang akan disimpan dalam LSTM. Rumus yang digunakan adalah tangen hiperbolik dari hasil perkalian matriks input ($h_{t-1}$ dan $x_t$) dengan bobot ($W_c$), ditambah dengan bias ($b_c$). Hasil perhitungan ini menghasilkan matriks $C_t$, yang merupakan nilai Sel Memori Baru yang akan digunakan dalam perhitungan selanjutnya. Gambar 2.10 merupakan representasi sel memori baru pada model LSTM. Dalam proses perhitungan, nilai sel memori baru diperoleh dengan menggunakan

metode berikut:

$$\tilde{C}t = \tanh\left(\begin{bmatrix} 0.3 & -0.2 & 0.1 & 0.1 & -0.2 & 0.3 \\ 0.2 & -0.1 & -0.2 & -0.2 & 0.3 & 0.1 \\ -0.1 & 0.3 & 0.2 & 0.3 & -0.1 & -0.2 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ -0.2 \\ 0.3 \\ 0.8 \\ -0.4 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.3 \\ -0.2 \end{bmatrix}\right)$$

$$= \tanh\left(\begin{bmatrix} 0.41 \\ -0.21 \\ 0.21 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.3 \\ -0.2 \end{bmatrix}\right)$$

$$= \tanh\left(\begin{bmatrix} 0.51 \\ 0.09 \\ 0.01 \end{bmatrix}\right)$$

$$\approx \begin{bmatrix} 0.469945 \\ 0.0897578 \\ 0.00999967 \end{bmatrix}$$

5. Sel Memori Terkini ($C_t$)

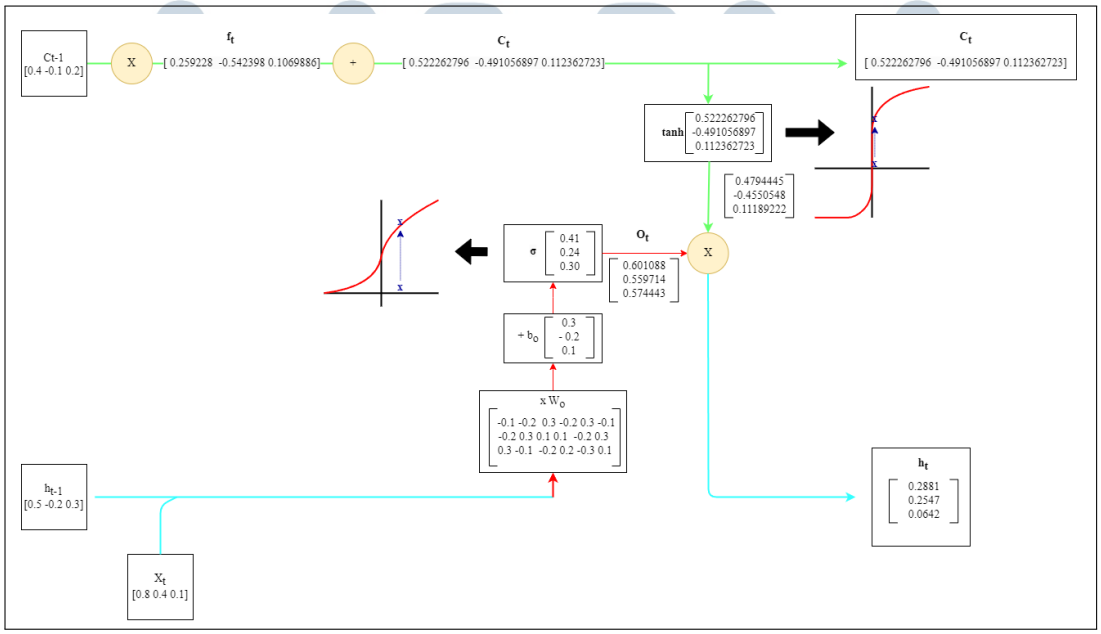
   Langkah terakhir dalam perhitungan LSTM adalah menggabungkan informasi dari Sel Memori Sebelumnya ($C_{t-1}$), gerbang Forget Gate ($f_t$), gerbang Input Gate ($i_t$), dan Sel Memori Baru ($C_t$) untuk menghasilkan Sel Memori Terkini ($C_t$). Rumus yang digunakan adalah perkalian antara gerbang Forget Gate dengan Sel Memori Sebelumnya, ditambah dengan perkalian antara gerbang Input Gate dengan Sel Memori Baru. Gambar 2.10 merupakan representasi sel memori terkini dalam model LSTM. Dalam proses perhitungan, nilai sel memori terkini diperoleh dengan menggunakan

metode berikut:

$$
\begin{aligned}
C_t &= \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 0.4 \\ -0.1 \\ 0.2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 0.469945 \\ 0.0897578 \\ 0.00999967 \end{bmatrix} \\
&= \begin{bmatrix} 0.4 \\ -0.1 \\ 0.2 \end{bmatrix} + \begin{bmatrix} 0.469945 \\ 0.0897578 \\ 0.00999967 \end{bmatrix} \\
&= \begin{bmatrix} 0.869945 \\ -0.010242 \\ 0.21 \end{bmatrix}
\end{aligned}
$$

6. Output Gate ($o_t$)

   Output Gate mengendalikan seberapa banyak informasi dalam Sel Memori Terkini ($C_t$) yang akan diungkapkan sebagai output dari LSTM. Rumusnya juga sigmoid dari hasil perkalian matriks input ($h_{t-1}$ dan $x_t$) dengan bobot ($W_o$), ditambah dengan bias ($b_o$). Hasil perhitungan ini menghasilkan matriks $o_t$, yang menunjukkan seberapa banyak informasi yang akan diungkapkan sebagai output. Gambar 2.12 merupakan representasi model LSTM pada output gate.




Gambar 2.12. Representasi model output gate LSTM

Dalam proses perhitungan, nilai output gate LSTM diperoleh dengan menggunakan metode berikut:

$$o_t = \sigma\left(\begin{bmatrix} -0.1 & -0.2 & 0.3 & -0.2 & 0.3 & -0.1 \\ -0.2 & 0.3 & 0.1 & 0.1 & -0.2 & 0.3 \\ 0.3 & -0.1 & -0.2 & 0.2 & -0.3 & 0.1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ -0.2 \\ 0.3 \\ 0.8 \\ -0.4 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.3 \\ -0.2 \\ 0.1 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix} 0.21 \\ -0.06 \\ 0.4 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.3 \\ -0.1 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix} 0.41 \\ 0.24 \\ 0.3 \end{bmatrix}\right)$$

$$\approx \begin{bmatrix} 0.601088 \\ 0.559714 \\ 0.574443 \end{bmatrix}$$

7. Menghitung nilai Output ($h_t$)

   Output Gate ($o_t$) memiliki peran penting dalam proses LSTM. Selain digunakan untuk mengontrol aliran informasi dari Sel Memori Terkini ($C_t$), Output Gate juga berperan dalam menghasilkan Output ($h_t$) dari LSTM. Untuk menghasilkan Output ($h_t$), Output Gate ($o_t$) digunakan untuk mengalikan Sel Memori Terkini ($C_t$) dengan fungsi tangen hiperbolik. Gambar 2.12 merupakan representasi nilai output dari model LSTM. Dalam proses perhitungan, nilai output diperoleh dengan menggunakan metode

berikut:

$$h_t = \begin{bmatrix} 0.601088 \\ 0.559714 \\ 0.574443 \end{bmatrix} \cdot \tanh\left(\begin{bmatrix} 0.869945 \\ -0.010242 \\ 0.21 \end{bmatrix}\right)$$
$$= \begin{bmatrix} 0.601088 \\ 0.559714 \\ 0.574443 \end{bmatrix} \cdot \begin{bmatrix} 0.701346 \\ -0.0102416 \\ 0.206966 \end{bmatrix}$$
$$= \begin{bmatrix} 0.421571 \\ -0.005732 \\ 0.118890 \end{bmatrix}$$

Dengan memanfaatkan rumus-rumus di atas dan matriks input yang telah diberikan, nilai-nilai gerbang LSTM dan output yang dihasilkan dapat dihitung. Hal ini memberikan pemahaman yang lebih baik tentang bagaimana LSTM memproses informasi dan mengatur aliran data melalui gerbang-gerbangnya. Proses tersebut memberikan gambaran yang lebih jelas tentang mekanisme LSTM dalam mengolah informasi.

## 2.8 Evaluasi dan Validasi

Evaluasi dan validasi model pada machine learning penting untuk menentukan kinerja dan keandalan model. Dalam penelitian ini, metrik evaluasi MSE, RMSE, dan MAPE digunakan untuk evaluasi, sedangkan K-fold Cross Validation dan Time Series Split digunakan untuk validasi pada model pergerakan harga emas di pasar Indonesia menggunakan algoritma LSTM.

### 2.8.1 Evaluasi

Dalam konteks machine learning, evaluasi merujuk pada proses mengukur kinerja algoritma atau model machine learning. Evaluasi yang komprehensif sangat penting dalam pengembangan model machine learning karena dapat memberikan pemahaman yang baik tentang kinerja model, mengidentifikasi kelemahan yang ada, dan memungkinkan perbaikan model guna mencapai hasil yang optimal. Evaluasi yang cermat dan akurat merupakan langkah kritis dalam tahapan pengembangan model machine learning, karena dapat memberikan wawasan yang berharga dalam menggambarkan sejauh mana model dapat berfungsi secara efektif

dalam memenuhi tujuan penelitian atau proyek machine learning yang dijalankan [55].

1. **Mean Squared Error (MSE) dan Root Mean Squared Error (RMSE)**

   Dalam mengukur akurasi dan kinerja prediksi model yang berbeda, terdapat beberapa parameter evaluasi yang dapat digunakan, antara lain Mean Square Error (MSE) dan Root Mean Square Error (RMSE). Parameter evaluasi ini umum digunakan dalam konteks prediksi dalam skripsi untuk mengukur seberapa akurat hasil prediksi dari model yang dikembangkan. MSE menghitung rata-rata dari kuadrat selisih antara nilai prediksi dan nilai sebenarnya, sedangkan RMSE menghitung akar kuadrat dari MSE. Kedua parameter evaluasi ini memberikan gambaran tentang seberapa dekat hasil prediksi dengan nilai sebenarnya, di mana semakin rendah nilai MSE dan RMSE, semakin akurat prediksi yang diperoleh. Penggunaan parameter evaluasi ini dapat membantu menguji dan memvalidasi kinerja model dalam konteks prediksi nilai emas [3]. Mean Square Error (MSE) didefinisikan sebagai:

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

   di mana $y_i$ adalah nilai sebenarnya, $\hat{y}_i$ adalah nilai prediksi, dan $n$ adalah jumlah sampel.

   Root Mean Square Error (RMSE) dapat dihitung dengan mengambil akar kuadrat dari MSE:

$$RMSE = \sqrt{MSE} \tag{2.12}$$

2. **Mean Absolute Percentage Error (MAPE)**

   Mean Absolute Percentage Error (MAPE) adalah ukuran kesalahan yang menghitung persentase deviasi antara data aktual dan data peramalan. Ini dihitung dengan mengambil kesalahan absolut untuk setiap periode, membaginya dengan nilai observasi aktual untuk periode tersebut, dan kemudian mencari rata-rata kesalahan persentase absolut [56]. Pendekatan ini berguna ketika ukuran atau besaran variabel peramalan menjadi penting dalam mengevaluasi akurasi peramalan [57].

   MAPE memberikan indikasi tentang besarnya kesalahan peramalan dibandingkan dengan nilai aktual. Ini adalah alat yang berguna untuk

mengevaluasi akurasi peramalan dan menilai tingkat keakuratan antara nilai peramalan dan nilai yang terealisasi [56]. Rumus untuk menghitung MAPE adalah sebagai berikut:

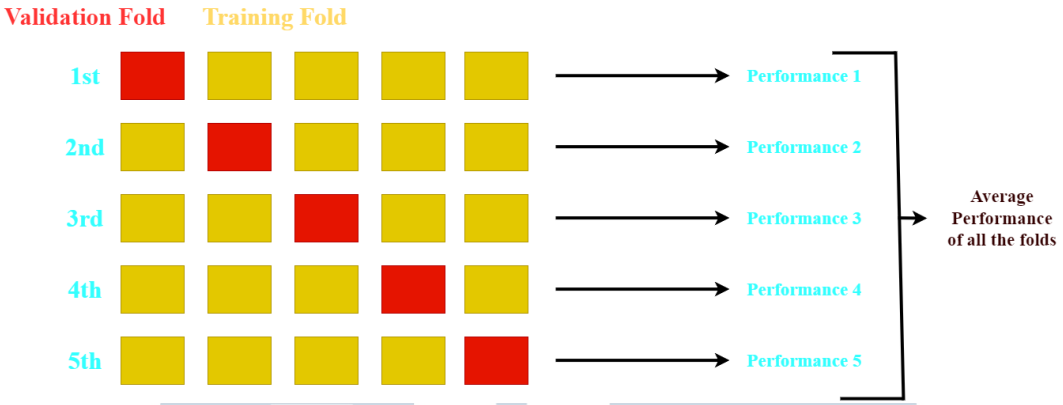
$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{X_i - F_i}{X_i}\right| \times 100\% \tag{2.13}$$

### 2.8.2 Validasi

Validasi merujuk pada suatu tahap evaluasi yang bertujuan untuk mengukur kinerja model machine learning atau statistik dengan menggunakan data yang tidak digunakan dalam proses pelatihan model. Tujuan dari validasi adalah untuk menguji sejauh mana model yang telah dilatih mampu menggeneralisasi hasilnya dengan baik pada data baru yang belum pernah dilihat sebelumnya. Validasi ini dilakukan untuk memastikan kehandalan dan keakuratan model yang telah dikembangkan sebelum diterapkan pada data yang sebenarnya [58].

1. **K-fold Cross Validation**

   K-fold Cross Validation merupakan salah satu teknik validasi model yang sering digunakan dalam machine learning untuk mengukur kinerja model pada dataset yang terbatas [59, 60]. Metode ini melibatkan pemisahan data menjadi k subset yang memiliki ukuran yang sama atau hampir sama, yang dikenal sebagai ”fold” [61]. Proses ini dilakukan secara berulang, di mana dalam setiap iterasi, satu fold digunakan sebagai data validasi, sementara fold yang lain digunakan sebagai data pelatihan. Dengan demikian, setiap fold akan berperan sebagai data validasi tepat satu kali selama proses validasi K-fold. Teknik K-fold Cross-Validation memungkinkan model untuk dievaluasi dengan lebih andal [62], menghindari overfitting [63], dan memberikan estimasi yang lebih akurat tentang kinerja model pada data yang terbatas. Secara detail ilustrai 5 *k-folds cross validation* dijelaskan pada gambar 2.13.



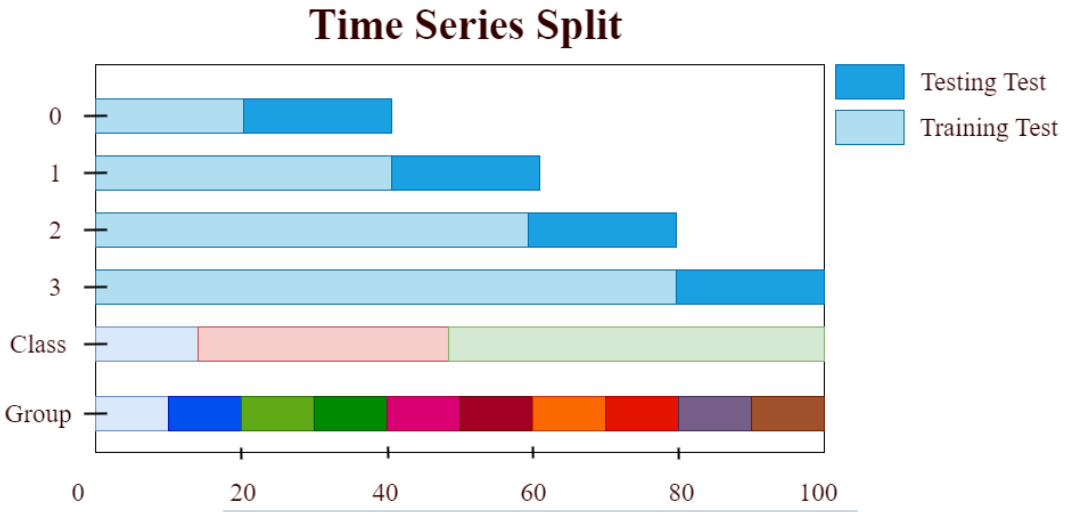

Gambar 2.13. Ilustrasi 5 *k-folds cross validation*
sumber: [64]

2. **Time Series Split**

Time Series Split atau pembagian rangkaian waktu adalah suatu metode dalam machine learning yang digunakan untuk membagi dataset rangkaian waktu menjadi set pelatihan dan set pengujian dengan mempertimbangkan urutan waktu data. Metode ini sangat berguna ketika ingin melatih model untuk melakukan prediksi atau analisis data berdasarkan urutan waktu, seperti prediksi harga saham, data cuaca, atau data ekonomi. Dalam skripsi ini, metode Time Series Split digunakan untuk memisahkan dataset rangkaian waktu menjadi bagian pelatihan dan pengujian, sehingga memungkinkan evaluasi yang lebih akurat terhadap performa model dalam memprediksi data pada masa depan berdasarkan data historis. Metode Time Series Split juga membantu menghindari informasi bocor atau leakage yang dapat mengganggu validitas evaluasi model, karena memperhatikan urutan waktu data saat membagi dataset menjadi set pelatihan dan pengujian [65].

Metode Time Series Split merupakan pendekatan yang digunakan untuk membagi dataset rangkaian waktu ke dalam beberapa lipatan (folds) dengan urutan waktu yang berbeda. Setiap lipatan terdiri dari dua bagian, yakni set pelatihan (*training set*) dan set pengujian (*testing set*). Set pelatihan digunakan untuk melatih model, sementara set pengujian digunakan untuk menguji performa model. Dalam setiap lipatan, data di bagian pengujian selalu berada setelah data di bagian pelatihan dalam urutan waktu, sehingga tidak ada informasi masa depan yang dapat bocor ke dalam pelatihan model. Pendekatan ini digunakan untuk menghindari kontaminasi data antara

pelatihan dan pengujian, sehingga memberikan hasil evaluasi yang lebih akurat dan kredibel terhadap kinerja model pada data masa depan [65]. Secara detail ilustrai time series split dijelaskan pada Gambar 2.14.




Gambar 2.14. Ilustrai *time series split*
sumber: [65]