BAB 2
LANDASAN TEORI

2.1 Indeks Harga Saham Gabungan

Indeks Harga Saham Gabungan atau IHSG merupakan indeks yang mengukur pergerakan dari semua harga saham yang ada di bursa efek Indonesia [12]. IHSG dapat memberikan gambaran bagaimana kondisi dari pasar modal di Indonesia. Perubahan nilai pada indeks saham ini dapat dipengaruhi faktor eksternal seperti nilai tukar rupiah, kebijakan pemerintah, inflasi, dan sebagainya.

2.2 LQ45

Pasar modal Indonesia mempunyai banyak daftar indeks saham yang berisi daftar perusahaan yang memenuhi kriteria untuk dapat masuk ke indeks tertentu. Salah satu indeks saham adalah LQ45 yang berisi daftar dari 45 perusahaan yang memiliki likuiditas yang tinggi dan kapitalisasi pasar besar serta didukung oleh fundamental yang baik [12]. Daftar emiten yang masuk ke indeks LQ45 dapat dilihat pada Gambar 2.1. Daftar indeks LQ45 lebih lengkap dapat dilihat pada lampiran.

Lampiran Pengumuman BEI No. Peng-00012/BEI.POP/01-2025 tanggal 22 Januari 2025

Nama Indeks : **LQ45**

Evaluasi : **Mayor**

Periode Efektif Konsituen : **03 Februari 2025 s.d. 30 April 2025**

Periode Efektif Jumlah Saham Penghitungan Indeks : **03 Februari 2025 s.d. 30 April 2025**

No.	Kode	Rasio Free Float	Jumlah Saham untuk Indeks (lembar)			Bobot pada Indeks		
			Pra Evaluasi	Pasca Evaluasi * (15% Cap)	Keterangan Tetap/Naik/ Turun/Baru	Pra Evaluasi	Pasca Evaluasi	Keterangan Tetap/Naik/ Turun/Baru
1	ACES	39.99%	6,851,579,958	6,846,443,841	Turun	0.30%	0.30%	Tetap
2	ADMR	13.15%	5,502,761,820	5,376,026,592	Turun	0.33%	0.32%	Turun
3	ADRO	32.06%	10,374,898,008	9,861,228,288	Turun	1.42%	1.36%	Turun
4	AKRA	32.87%	6,598,151,101	6,598,151,101	Tetap	0.48%	0.49%	Naik
5	AMMN	17.83%	13,038,775,535	12,929,998,208	Turun	6.26%	6.28%	Naik
6	AMRT	42.48%	17,602,236,271	17,639,608,322	Naik	2.85%	2.89%	Naik
7	ANTM	34.84%	8,372,318,430	8,372,318,430	Tetap	0.74%	0.74%	Tetap
8	ARTO	26.92%	3,683,403,476	3,694,169,253	Naik	0.55%	0.56%	Naik
9	ASII	44.99%	18,221,647,268	18,213,550,558	Turun	5.21%	5.27%	Naik
10	BBCA	42.83%	27,712,397,884	26,460,786,623	Turun	15.53%	15.00%	Turun
11	BBNI	40.01%	14,880,508,934	14,773,428,348	Turun	4.00%	4.02%	Naik
12	BBRI	46.64%	60,727,785,486	60,351,912,618	Turun	14.92%	15.00%	Naik
13	BBTN	40.18%	5,582,649,368	5,582,649,368	Tetap	0.36%	0.36%	Tetap
14	BMRI	39.09%	36,193,079,998	36,119,159,998	Turun	12.64%	12.76%	Naik
15	BRIS	9.12%	4,142,076,502	4,164,910,441	Naik	0.67%	0.68%	Naik
16	BRPT	27.25%	25,339,873,037	25,546,116,917	Naik	1.39%	1.41%	Naik
17	CPIN	34.14%	5,598,277,200	5,598,277,200	Tetap	1.51%	1.53%	Naik
18	**CTRA**	**42.96%**	**-**	**7,962,934,682**	**Baru**	**-**	**0.45%**	**Baru**
19	ESSA	53.96%	9,292,230,693	9,295,676,088	Naik	0.46%	0.46%	Tetap
20	EXCL	33.19%	4,357,326,138	4,357,326,138	Tetap	0.57%	0.58%	Naik
21	GOTO	77.94%	932,414,039,327	888,962,804,471	Turun	4.67%	4.50%	Turun
22	ICBP	19.47%	2,270,573,488	2,270,573,488	Tetap	1.39%	1.41%	Naik
23	INCO	20.39%	2,149,062,066	2,149,062,066	Tetap	0.45%	0.45%	Tetap
24	INDF	47.98%	4,201,434,080	4,212,848,635	Naik	1.80%	1.83%	Naik
25	INKP	38.29%	1,901,713,670	2,094,839,368	Naik	0.74%	0.83%	Naik
26	ISAT	16.36%	5,279,457,754	5,276,232,673	Turun	0.69%	0.70%	Naik
27	ITMG	34.73%	392,422,953	392,422,953	Tetap	0.59%	0.60%	Naik
28	**JPFA**	**41.01%**	**-**	**4,809,068,490**	**Baru**	**-**	**0.55%**	**Baru**
29	JSMR	24.81%	1,797,774,696	1,800,677,845	Naik	0.44%	0.45%	Naik
30	KLBF	39.63%	18,773,486,405	18,576,610,892	Turun	1.33%	1.33%	Tetap
31	**MAPA**	**30.70%**	**-**	**8,750,728,000**	**Baru**	**-**	**0.49%**	**Baru**
32	MAPI	48.64%	8,074,240,000	8,074,240,000	Tetap	0.64%	0.64%	Tetap
33	MBMA	29.48%	31,621,058,947	31,837,049,787	Naik	0.78%	0.80%	Naik
34	MDKA	47.51%	13,589,847,888	11,627,114,590	Turun	1.34%	1.16%	Turun
35	MEDC	24.93%	6,289,085,059	6,266,462,451	Turun	0.43%	0.43%	Tetap
36	PGAS	43.02%	10,431,120,977	10,428,696,826	Turun	0.98%	0.99%	Naik
37	PGEO	10.24%	4,250,421,673	4,250,421,673	Tetap	0.23%	0.23%	Tetap
38	PTBA	32.76%	3,774,167,970	3,774,167,970	Tetap	0.58%	0.59%	Naik
39	SIDO	22.12%	6,627,000,000	6,636,000,000	Naik	0.23%	0.23%	Tetap
40	SMGR	48.72%	3,291,375,793	3,289,350,331	Turun	0.56%	0.57%	Naik
41	SMRA	49.41%	8,156,883,626	8,156,883,626	Tetap	0.22%	0.22%	Tetap
42	TLKM	47.15%	46,717,741,349	46,707,835,127	Turun	7.10%	7.18%	Naik
43	TOWR	37.53%	19,145,788,763	19,145,788,763	Tetap	0.74%	0.75%	Naik
44	UNTR	35.15%	1,311,142,500	1,311,142,500	Tetap	1.98%	2.01%	Naik

GO PUBLIC Lebih Baik! IDXCarbon IDX Mobile

Indonesia Stock Exchange Building, Tower I, 6th Floor, Jl. Jend. Sudirman Kav 52-53, Jakarta 12190, Indonesia

Gambar 2.1. Daftar emiten LQ45 (halaman 1)

Sumber: IDX

2.3 Moving Average

Moving average merupakan indikator teknis yang digunakan investor untuk mengetahui tren pergerakan harga saham berdasarkan rata-rata harga selama

periode tertentu [10]. Tren dapat dianalisis dengan membandingkan *moving average* jangka pendek dengan jangka panjang, apabila *moving average* jangka pendek berada di atas *moving average* jangka panjang maka saham cenderung menunjukkan tren positif. Sebaliknya, apabila *moving average* jangka pendek berada di bawah *moving average* jangka panjang, maka saham tersebut cenderung mengalami tren negatif.

2.3.1 Simple Moving Average (SMA)

Simple Moving Average (SMA) adalah metode *moving average* yang menggunakan perhitungan yang simpel tanpa memerlukan adanya *weight*. SMA dihitung dengan mendapatkan nilai rata-rata dari suatu harga. *Moving average* ini bergerak lebih lambat sehingga menyebabkan terjadinya efek *lagging* [13]. Rumus untuk menghitung SMA dapat dilihat pada rumus Persamaan 2.1. P_m merupakan nilai data pada waktu m dan n merupakan jumlah data yang digunakan pada kalkulasi metode *moving average* ini [14].

$$SMA = \frac{P_M + P_{M-1} + ... + P_{M-(n-1)}}{n} \tag{2.1}$$

Keterangan:

- SMA: SMA pada waktu ke-t
- P_M: nilai data pada waktu ke-M
- n: total data atau hari yang digunakan

2.3.2 Weighted Moving Average (WMA)

Weighted Moving Average (WMA) merupakan perbaikan dari SMA dengan menambahkan *weight* pada data. Data yang terbaru menggunakan *weight* yang lebih besar dari data yang lama. *Weight* dihitung dari jumlah hari yang digunakan pada WMA. Rumus untuk menghitung WMA dapat dilihat pada rumus Persamaan 2.2. Pada rumus ini n adalah jumlah data yang digunakan dan P_m adalah nilai data pada waktu ke m.

$$WMA = \frac{nP_M + (n-1)P_M - 1 + ... + 2P_(M-n+2) + P_(M-n+1)}{n + (n-1) + ... + 2 + 1} \tag{2.2}$$

Keterangan:

- WMA: WMA pada waktu ke-t
- P_M: nilai data pada waktu ke-M
- n: total data atau hari yang digunakan

2.3.3 Exponential Moving Average (EMA)

Exponential Moving Average (EMA) merupakan tipe dari WMA yang ditambahkan *weight* pada setiap nilai dari data *series*. Sama seperti WMA, data yang terbaru menggunakan *weight* yang lebih besar dari data yang lama. Adanya *weight* ini dapat mengurangi terjadinya efek *lagging* sehingga EMA dapat bergerak lebih cepat dibanding dengan SMA [13]. Rumus untuk menghitung EMA saat nilai waktu atau t lebih dari satu dapat dilihat pada rumus Persamaan 2.4. Sedangkan pada waktu sama dengan satu digunakan rumus Persamaan 2.5. Pada EMA, α merepresentasikan derajat dari pengurangan *weight* yang dimana nilainya merupakan konstanta *smoothing factor* dari 0 sampai 1. Selanjutnya, s_t adalah nilai EMA pada periode t dan Y_t adalah nilai data pada waktu t.

$$\alpha = \frac{2}{(n+1)} \tag{2.3}$$

$$S_t = \alpha \times Y_t + (1-\alpha) \times S_{t-1} \tag{2.4}$$

$$S_1 = Y_1 \tag{2.5}$$

Keterangan:

- α: derajat dari pengurangan *weight*
- S_t: EMA pada waktu ke-t
- Y_t: nilai data pada waktu ke-t
- n: total data atau hari yang digunakan

2.3.4 Weighted Exponential Moving Average

Weighted Exponential Moving Average atau WEMA merupakan pendekatan *moving average* baru yang menggabungkan *weighing* dari *weighted moving average* (WMA) dengan *exponential moving average* (EMA) [14]. Hal yang membedakan WEMA dengan EMA adalah WEMA menggunakan beberapa data lama yang diberikan pada periode tertentu, sedangkan EMA hanya menggunakan data terbaru untuk memprediksi nilai. Proses kalkulasi WEMA dimulai dengan mendapatkan nilai prediksi WMA dengan rumus Persamaan 2.2 yang kemudian digunakan sebagai nilai dasar *Ht* pada rumus Persamaan 2.6. Pada rumus WEMA, nilai α dan Y_t sama dengan yang digunakan pada rumus EMA pada persamaan Persamaan 2.4 dengan rumus α menggunakan rumus Persamaan 2.3 dan Y_t adalah data pada waktu t.

$$WEMA_t = \alpha \times Y_t + (1 - \alpha) \times H_t \tag{2.6}$$

Keterangan:

- $WEMA_t$: *Weighted Exponential Moving Average* atau WEMA pada waktu ke-t
- α: derajat dari pengurangan *weight*
- Y_t: nilai data pada waktu ke-t
- H_t: nilai *Weighted Moving Average* atau WMA pada waktu ke-t

2.4 Deep Learning

Deep learning merupakan cabang dari teknologi *machine learning* dan *artificial intelligence* yang kini dianggap sebagai teknologi inti dari revolusi industri keempat [15]. Kemampuan teknologi *deep learning* untuk belajar dari data menjadikan teknologi ini sebagai topik yang hangat dibicarakan dalam dunia komputasi dan banyak diterapkan di berbagai macam bidang seperti, analisis teks, keamanan siber, kesehatan, dan sebagainya.

2.5 Recurrent Neural Network

Recurrent Neural Network atau RNN adalah model *deep learning* yang dirancang untuk memproses dan mengubah data berurutan menjadi *output* tertentu. RNN dapat digunakan untuk memprediksi data berurutan seperti kata, kalimat, atau data *time series*. Model ini bekerja sebagai sistem perangkat lunak dengan komponen yang saling terhubung dengan meniru cara manusia memproses data berurutan.

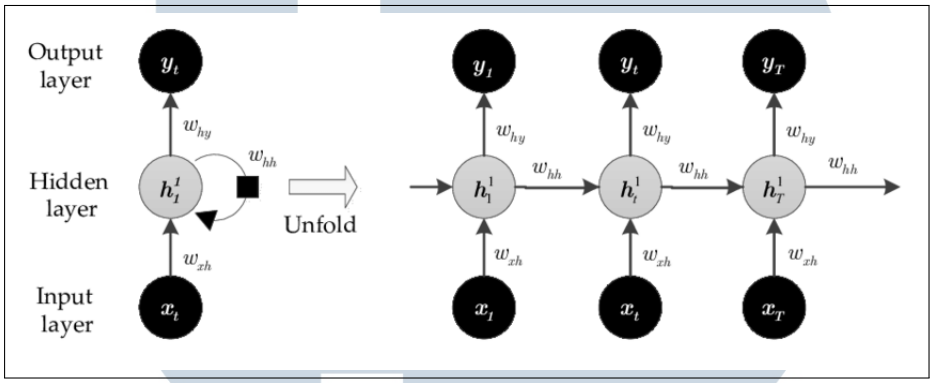
RNN memiliki kelemahan yaitu terdapat masalah *vanishing gradient* dan *exploding gradient* yang terjadi ketika menangani data dengan dependensi jangka panjang [6]. *Vanishing gradient* terjadi ketika saat proses *backpropagation* nilai gradien di *hidden layer* atau *hidden state* sangat kecil [16]. Sebaliknya, *exploding gradient* terjadi ketika gradien di *hidden layer* sangat besar saat proses *backpropagation*. Untuk mengatasi masalah ini, dapat digunakan varian dari RNN, yaitu *Gated Recurrent Unit* dan *Long Short-Term Memory* (LSTM) [16]. Kedua model tersebut menyelesaikan masalah gradien itu dengan menambahkan gerbang pada model untuk melupakan atau menggunakan data dari ingatan sebelumnya saat proses perhitungan pada waktu (t). Rumus RNN dapat dilihat pada Persamaan 2.7.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{x_h}x_t + b_h) \tag{2.7}$$
$$y_t = \sigma(W_{hy}h_t + b_y)$$

Keterangan:

- x_t: *Input* pada waktu ke-t
- h_t: *Hidden state* pada waktu ke-t
- h_{t-1}: *Hidden state* pada waktu sebelumnya
- y_t: *Output* pada waktu ke-t
- W_{x_h}: Bobot dari input ke *hidden state*
- W_{hh}: Bobot dari *hidden state* sebelumnya ke *hidden state* saat ini
- W_{hy}: Bobot dari *hidden state* ke *output*

- b_h: Bias untuk *hidden state*
- b_y: Bias untuk *output*
- tanh: Fungsi aktivasi tanh
- σ: Fungsi aktivasi sigmoid, digunakan untuk menghitung *output*



Gambar 2.2. Perhitungan *Recurrent Neural Network*
Sumber: *Medium*

RNN memiliki tiga lapisan seperti *Input layer*, *Hidden layer*, *Output layer*. *Input layer* (x_t) berisi nilai data asli yang digunakan dalam memprediksi nilai pada waktu (t). Pada *Hidden layer* (h_t), memori akan disimpan pada *hidden layer* atau *hidden state* dan akan diperbarui pada setiap iterasi waktu (t). *Output layer* (y_t) merupakan tempat keluaran dari nilai prediksi pada RNN. Pada RNN semakin banyak iterasi yang dilakukan dapat menyebabkan masalah seperti *vanishing gradient* atau *exploding gradient*. Hal ini dapat terjadi pada saat proses *backpropagation* untuk memperbarui *weight* dan *bias*. Ketika nilai *weight* konstan (w_{hh}) bernilai kurang dari satu maka ketika *backpropagation* melewati waktu, gradien yang digunakan untuk memperbarui *weight* atau *bias* menjadi sangat kecil karena nilai weight akan dipangkatkan sebanyak waktu atau *step* yang digunakan untuk memprediksi [17]. Sebaliknya, saat nilai weight konstan (w_{hh}) bernilai lebih dari satu maka saat memperbarui nilai *weight* atau *bias* gradien menjadi sangat besar sehingga sulit untuk menemukan titik terendah dari fungsi *loss* untuk memperbarui *weight* atau *bias*.

Selanjutnya terdapat contoh perhitungan RNN menggunakan data saham A pada Tabel 2.1 untuk memprediksi harga saham pada hari keempat dengan nilai W_{hh}, W_{xh}, b_h, h_0, W_{h_y}, dan b_y adalah 0.6, 0.9, 0.1, 0, 1.2, dan 0.05.

Tabel 2.1. Contoh Saham A

Hari	Harga	Harga Normalisasi
1	1000	0.0
2	1200	1.0
3	1100	0.5

1. Hari Pertama

$$h_1 = \tanh(0.6 \times 0 + 0.9 \times 0.0 + 0.1) = \tanh(0.1) \approx 0.0997$$

2. Hari Kedua

$$h_2 = \tanh(0.6 \times 0.0997 + 0.9 \times 1.0 + 0.1) = \tanh(1.0598) \approx 0.786$$

3. Hari ketiga

$$h_3 = \tanh(0.6 \times 0.786 + 0.9 \times 0.5 + 0.1) = \tanh(1.0216) \approx 0.771$$

4. Hari Keempat (Prediksi)

$$y_3 = \sigma(1.2 \times 0.771 + 0.05) = \sigma(0.9752) = 0.726$$

Hasil prediksi hari keempat adalah $y_3 = 0.726$. Nilai ini masih dalam bentuk normalisasi sehingga perlu dilakukan *denormalization* data terlebih dahulu dengan rumus 2.8 sebelum diketahui nilai aslinya. Setelah proses *denormalization*, didapat harga prediksi saham pada hari keempat adalah 1145.

$$X_i = X_{normalization} \times (X_{max} - X_{min}) + X_{min} \tag{2.8}$$

$$X_i = 0.726 \times (1200 - 1000) + 1000 = 1145$$

2.6 Gated Recurrent Unit

Gated Recurrent Unit (GRU) adalah salah satu varian dari RNN. Seperti LSTM, GRU juga dapat memecahkan masalah *vanishing gradient* yang terdapat

pada RNN [18]. GRU mempunyai dua gerbang yaitu *update gate* dan *reset gate*. Update gate berfungsi untuk memperbarui kecepatan update dari hidden state dan reset gate berfungsi menentukan seberapa banyak informasi untuk dilupakan dengan mereset bagian memori.

Model GRU mempunyai beberapa kelebihan saat digunakan untuk sistem prediksi. Model ini mempunyai arsitektur yang lebih simpel dengan menggabungkan *forget gate* dan *update gate* yang biasa terdapat pada model LSTM sehingga GRU mampu memprediksi data dengan lebih cepat dan efisien [19]. GRU juga fleksibel dalam beradaptasi terhadap ukuran dataset yang berbeda.

Walaupun begitu, GRU juga mempunyai beberapa kekurangan. Kapasitas ingatan pada GRU sangat terbatas dan tidak mampu mempelajari data dengan jangka waktu yang lebih panjang seperti yang dapat dilakukan oleh LSTM [19]. GRU juga dapat mengalami kesulitan untuk generalisasi ketika data yang digunakan untuk prediksi terlalu kompleks sehingga dapat mengurangi kinerja model dan menghasilkan akurasi yang lebih rendah.

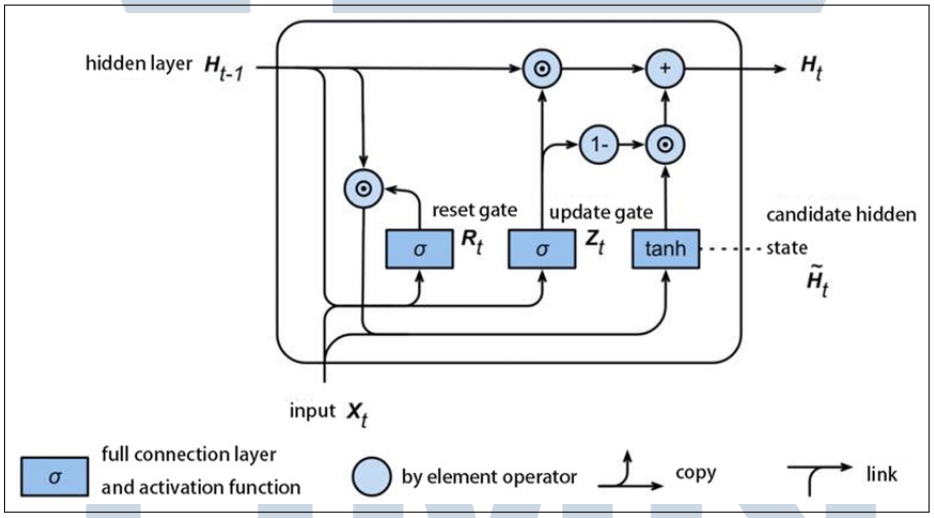
Dibanding dengan GRU, model LSTM mempunyai kelebihan pada ingatan dan kinerja model [19]. LSTM mampu lebih baik ketika memproses data dengan dependensi yang lebih lama karena mempunyai arsitektur yang lebih kompleks. LSTM mempunyai tiga gerbang seperti *forget gate*, *input gate*, dan *output gate* serta dua tipe ingatan yaitu *cell state* dan *hidden state*. Saat digunakan sebagai sistem prediksi, LSTM mempunyai kinerja yang lebih baik dibanding dengan GRU [19]. Berdasarkan penelitian [9], LSTM disebut lebih unggul saat memprediksi tren pada harga saham.

$$
\begin{aligned}
u_t &= \sigma(W_{uh} h_{t-1} + W_{ux} x_t + b_u) \\
r_t &= \sigma(W_{rh} h_{t-1} + W_{rx} x_t + b_r) \\
\tilde{h}_t &= \tanh(W_{hh}(r_t \odot h_{t-1}) + W_{hx} x_t + b_h) \\
h_t &= u_t \odot h_{t-1} + (1 - u_t) \odot \tilde{h}_t
\end{aligned}
\tag{2.9}
$$

Keterangan:

- x_t: Input pada waktu ke-t
- h_{t-1}: *Hidden state* pada waktu sebelumnya

- u_t: *Update gate*
- r_t: *Reset gate*
- $\tilde{h}_t$: Kandidat *hidden state*
- h_t: *Hidden state* baru
- $W_{uh}, W_{ux}, W_{rh}, W_{rx}, W_{hh}, W_{hx}$: Bobot
- b_u, b_r, b_h: Bias
- σ: Fungsi aktivasi sigmoid
- *tanh*: Fungsi aktivasi tanh



Gambar 2.3. *Gated Recurrent Unit*
Sumber: *Dive into Deep Learning*

Model GRU memiliki dua gerbang utama yaitu *reset gate* (r_t) dan *update gate* (u_t). *Reset gate* itu berisi fungsi aktivasi sigmoid yang menentukan apakah *hidden state* pada waktu (t) sebelumnya tetap digunakan atau tidak. Jika keluaran dari *reset gate* adalah nol maka *hidden state* sebelumnya dihapus dan jika keluarannya adalah satu maka nilai *hidden state* tetap digunakan sedangkan jika nilai berada di antara nol sampai satu maka *hidden state* sebelumnya akan digunakan sesuai dengan persentase nilai tersebut pada iterasi sekarang. *Update gate* berisi keluaran fungsi aktivasi sigmoid mengenai berapa banyak informasi yang dipertahankan pada iterasi (t).

Selanjutnya terdapat contoh perhitungan GRU menggunakan data saham pada Tabel 2.2 untuk memprediksi harga saham pada hari keempat dengan *weights* dan *bias* yang sudah tersedia pada Tabel 2.4 dan 2.3. Nilai dari h_0 adalah nol.

Tabel 2.2. Contoh Data Harga Saham

Hari	Harga	Harga Normalisasi
1	2500	0.00
2	2550	1.00
3	2525	0.50

Tabel 2.3. Contoh nilai *bias* untuk perhitungan *Gated Recurrent Unit*

Parameter	Bias
b_u (Update Gate)	0.1
b_r (Reset Gate)	0.2
b_h (Candidate)	0.05
b_y (Output)	0.1

Tabel 2.4. Contoh nilai bobot untuk perhitungan *Gated Recurrent Unit*

Parameter	Bobot
W_{ux} (Input u_t)	0.3
W_{rx} (Input r_t)	0.25
W_{hx} (Input $\tilde{h}_t$)	0.4
W_{uh} (Recurrent u_t)	0.35
W_{rh} (Recurrent r_t)	0.45
W_{hh} (Recurrent $\tilde{h}_t$)	0.3

1. Hari Pertama

$$u_1 = \sigma(0.35 \times 0 + 0.3 \times 0.00 + 0.1) = \sigma(0.1) \approx 0.525$$

$$r_1 = \sigma(0.45 \times 0 + 0.25 \times 0.00 + 0.2) = \sigma(0.2) \approx 0.550$$

$$\tilde{h}_1 = \tanh(0.3 \times (0.550 \times 0) + 0.4 \times 0.00 + 0.05) = \tanh(0.05) \approx 0.050$$

$$h_1 = 0.525 \times 0 + (1 - 0.525) \times 0.050 = 0.024$$

2. Hari Kedua

$$u_2 = \sigma(0.35 \times 0.024 + 0.3 \times 1.00 + 0.1) = \sigma(0.4084) \approx 0.601$$

$$r_2 = \sigma(0.45 \times 0.024 + 0.25 \times 1.00 + 0.2) = \sigma(0.4608) \approx 0.614$$

$$\tilde{h}_2 = \tanh(0.3 \times (0.614 \times 0.024) + 0.4 \times 1.00 + 0.05) = \tanh(0.4544) \approx 0.426$$

$$h_2 = 0.601 \times 0.024 + (1 - 0.601) \times 0.426 = 0.184$$

3. Hari Ketiga

$$u_3 = \sigma(0.35 \times 0.184 + 0.3 \times 0.50 + 0.1) = \sigma(0.3144) \approx 0.578$$

$$r_3 = \sigma(0.45 \times 0.184 + 0.25 \times 0.50 + 0.2) = \sigma(0.4078) \approx 0.600$$

$$\tilde{h}_3 = \tanh(0.3 \times (0.600 \times 0.184) + 0.4 \times 0.50 + 0.05) = \tanh(0.2831) \approx 0.275$$

$$h_3 = 0.578 \times 0.184 + (1 - 0.578) \times 0.275 = 0.223$$

Harga saham pada hari keempat dapat diprediksi menggunakan nilai $h_3 = 0.223$. Nilai ini masih dalam bentuk normalisasi sehingga perlu dilakukan *denormalization* data terlebih dahulu sebelum diketahui nilai aslinya. Setelah proses *denormalization*, didapat harga prediksi saham pada hari keempat adalah 2511.

2.7 Long Short-Term Memory

Long Short-Term Memory (LSTM) merupakan bentuk perbaikan dari RNN yang dapat digunakan untuk mempelajari data dengan rentang waktu yang lebih lama. Setiap *unit* dalam LSTM mempunyai struktur khusus yang memungkinkan informasi dipertahankan yang memungkinkan informasi untuk dipertahankan atau dilupakan secara selektif. LSTM mempunyai dua tipe ingatan yaitu ingatan jangka panjang dan ingatan jangka pendek. Gerbang pada LSTM mampu menentukan memori jangka panjang dan memori jangka pendek [20]. Komponen yang terdapat pada LSTM adalah *cell state*, *hidden state*, *forget gate*, *input gate*, dan *output gate*. Adanya dua tipe ingatan dan tiga gerbang pada model ini yang membedakan LSTM dengan RNN.

Cell state adalah komponen yang mengirimkan data tanpa banyak mengubah data tersebut sehingga dapat menjaga konteks jauh ke depan. *Hidden state* adalah

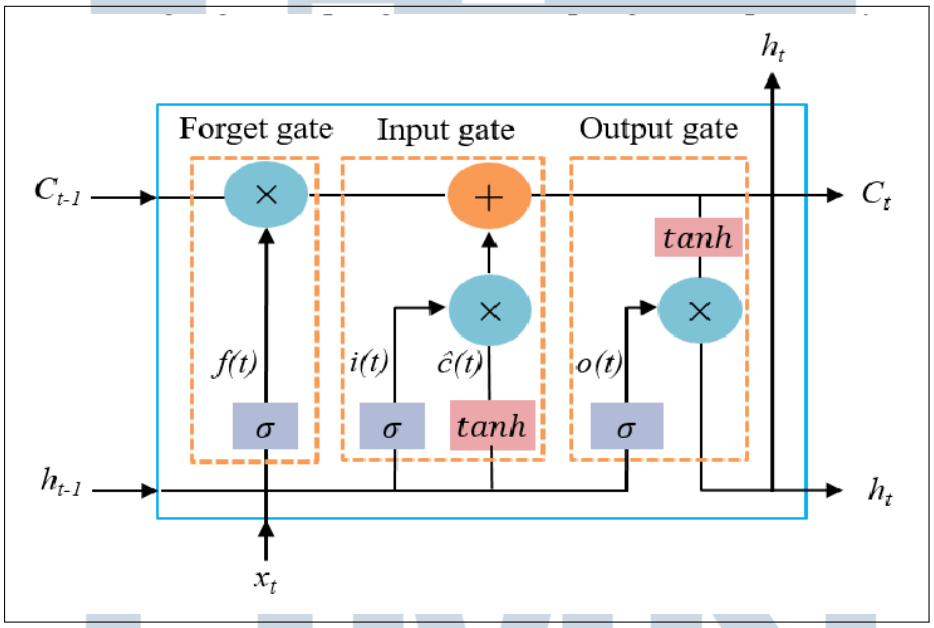
data yang dapat digunakan sebagai ingatan jangka pendek atau keluaran dari LSTM. *Forget gate* adalah komponen yang memutuskan data apa yang akan dibuang dari *cell state* [21]. *Input gate* bertujuan untuk menentukan data mana yang harus disimpan di *cell state*. *Output gate* merupakan komponen yang akan menentukan bagian mana dari *cell state* yang akan digunakan sebagai *output*. Pada setiap *cell memory* LSTM terdapat tiga fungsi aktivasi sigmoid dan satu fungsi aktivasi tanh [22]. Fungsi aktivasi sigmoid digunakan untuk menentukan berapa banyak data yang dapat digunakan pada LSTM. LSTM dapat digunakan sebagai model untuk memprediksi data *time series*, *Natural Language Processing*, *sentiment analysis*, dll [23]. Berdasarkan persamaan 2.10, f_t adalah *forget gate*, i_t adalah *input gate*, o_t adalah *output gate*, C_t adalah *state* saat ini, $\tilde{C}_t$ adalah kandidat sel, dan h_t adalah keluaran atau *output* dari LSTM.

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{C}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
h_t &= o_t \odot \tanh(C_t)
\end{aligned}
\tag{2.10}
$$

Keterangan:

- x_t: *Input* pada waktu ke-*t*
- h_t: *Hidden state* pada waktu ke-*t*
- h_{t-1}: *Hidden state* pada waktu sebelumnya
- C_t: *Cell state* pada waktu ke-*t*
- C_{t-1}: *Cell state* pada waktu sebelumnya
- $\tilde{C}_t$: Kandidat nilai *cell state* baru
- f_t: *Forget gate*
- i_t: *Input gate*

- o_t: *Output gate*
- W_f, W_i, W_o, W_c: Bobot untuk *input* x_t
- U_f, U_i, U_o, U_c: Bobot untuk *hidden state* sebelumnya h_{t-1}
- b_f, b_i, b_o, b_c: Bias untuk masing-masing *gate*
- σ: Fungsi aktivasi sigmoid
- tanh: Fungsi aktivasi tanh



Gambar 2.4. *Long Short Term Memory*
Sumber: Penelitian [24]

Berdasarkan Gambar 2.4 terdapat 3 gerbang pada LSTM yaitu *Forget gate* (f_t), *Input gate* (i_t), *Output gate* (o_t). Sebelum dimasukkan ke gerbang LSTM, nilai memori pada iterasi sebelumnya atau h_{t-1} ditambah dengan nilai *input* pada iterasi sekarang dan nilai bias model. Pada *forget gate*, nilai yang dimasukkan ke gerbang memasuki fungsi aktivasi Sigmoid yang akan mengeluarkan nilai rentang nol sampai satu kemudian keluaran dari *forget gate* dikalikan dengan nilai dari memori jangka panjang sebelumnya atau *cell state* untuk menentukan apakah *cell state* tetap digunakan pada waktu (t). Nilai nol akan menghapus nilai *cell state* sebelumnya dan satu tetap menggunakan nilai *cell state* sebelumnya. Jika nilainya berada di antara nol dan satu, maka *cell state* sebelumnya akan digunakan sesuai

dengan persentase nilai tersebut pada iterasi sekarang. Perhitungan fungsi aktivasi Sigmoid dapat dilihat pada persamaan 2.11. Pada *Input gate*, nilai yang dimasukkan ke gerbang ini akan memasuki fungsi aktivasi Sigmoid dan tanh. Fungsi aktivasi tanh menghasilkan nilai antara negatif satu sampai satu. Perhitungan fungsi aktivasi tanh dapat dilihat pada persamaan 2.12. Keluaran dari fungsi aktivasi Sigmoid akan digunakan untuk menentukan nilai dari keluaran fungsi aktivasi tanh yang akan ditambahkan dengan *cell state* sebelumnya untuk menjadi *cell state* pada iterasi sekarang. Pada *output gate*, nilai yang dimasukkan ke gerbang ini akan digunakan pada fungsi aktivasi Sigmoid yang dikalikan dengan keluaran tanh *cell state* untuk menentukan berapa banyak nilai yang akan digunakan sebagai memori atau keluaran pada iterasi sekarang.

$$\sigma = f(x) = \frac{1}{1+e^{-x}} \tag{2.11}$$

$$\tanh = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.12}$$

Selanjutnya terdapat contoh perhitungan LSTM menggunakan data saham B pada Tabel 2.5 untuk memprediksi harga saham pada hari keempat dengan *input weight*, *output weight* serta *bias* yang sudah tersedia pada Tabel 2.7, 2.8 dan 2.6. Nilai dari H_0 dan C_0 adalah nol.

Tabel 2.5. Contoh Saham B

Hari	Harga	Harga Normalisasi
1	100	0.00
2	102	0.50
3	105	1.00

Tabel 2.6. Contoh nilai *bias* untuk perhitungan *Long Short-Term Memory*

Parameter	Bias
b_f (Forget Gate)	0.1
b_i (Input Gate)	0.2
b_c (Candidate Cell)	0.05
b_o (Output Gate)	0.15

Tabel 2.7. Contoh nilai bobot *input* untuk perhitungan *Long Short-Term Memory*

Parameter	Weight
W_f (Forget Gate)	0.2
W_i (Input Gate)	0.5
W_c (Candidate Cell)	0.3
W_o (Output Gate)	0.4

Tabel 2.8. Contoh nilai bobot *output* untuk perhitungan *Long Short-Term Memory*

Parameter	Value
U_f (Forget Gate)	0.25
U_i (Input Gate)	0.45
U_c (Candidate Cell)	0.35
U_o (Output Gate)	0.30

1. Hari Pertama

$$f_1 = \sigma(0.2 \times 0.00 + 0.25 \times 0 + 0.1) = \sigma(0.1) = 0.525$$

$$i_1 = \sigma(0.5 \times 0.00 + 0.45 \times 0 + 0.2) = \sigma(0.2) = 0.550$$

$$\tilde{C}_1 = \tanh(0.3 \times 0.00 + 0.35 \times 0 + 0.05) = \tanh(0.05) = 0.050$$

$$C_1 = 0.525 \times 0 + 0.550 \times 0.050 = 0.028$$

$$h_1 = \sigma(0.4 \times 0.00 + 0.30 \times 0 + 0.15) \times \tanh(0.028) = 0.537 \times 0.028 = 0.015$$

2. Hari Kedua

$$f_2 = \sigma(0.2 \times 0.50 + 0.25 \times 0.015 + 0.1) = \sigma(0.203) = 0.551$$

$$i_2 = \sigma(0.5 \times 0.50 + 0.45 \times 0.015 + 0.2) = \sigma(0.457) = 0.612$$

$$\tilde{C}_2 = \tanh(0.3 \times 0.50 + 0.35 \times 0.015 + 0.05) = \tanh(0.215) = 0.212$$

$$C_2 = 0.551 \times 0.028 + 0.612 \times 0.212 = 0.140$$

$$h_2 = \sigma(0.4 \times 0.50 + 0.30 \times 0.015 + 0.15) \times \tanh(0.140) = 0.583 \times 0.139 = 0.081$$

3. Hari Ketiga

$$f_3 = \sigma(0.2 \times 1.00 + 0.25 \times 0.081 + 0.1) = \sigma(0.320) = 0.579$$

$$i_3 = \sigma(0.5 \times 1.00 + 0.45 \times 0.081 + 0.2) = \sigma(0.700) = 0.668$$

$$\tilde{C}_3 = \tanh(0.3 \times 1.00 + 0.35 \times 0.081 + 0.05) = \tanh(0.375) = 0.358$$

$$C_3 = 0.579 \times 0.140 + 0.668 \times 0.358 = 0.283$$

$$h_3 = \sigma(0.4 \times 1.00 + 0.30 \times 0.081 + 0.15) \times \tanh(0.283) = 0.638 \times 0.276 = 0.176$$

Hasil prediksi hari keempat adalah $h_3 = 0.176$. Nilai ini masih dalam bentuk normalisasi sehingga perlu dilakukan *denormalization* pada data terlebih dahulu sebelum diketahui nilai aslinya. Setelah proses *denormalization*, didapat harga prediksi saham pada hari keempat adalah 101.

2.8 Root Mean Squared Error

Root Mean Squared Error atau RMSE merupakan akar kuadrat dari *Mean Squared Error* (MSE) [25]. MSE dihitung dari rata-rata dari kuadrat selisih antara nilai prediksi dan nilai sebenarnya. Semakin rendah nilai RMSE berarti performa model semakin baik.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.13}$$

Keterangan:

- n: jumlah data
- y_i: nilai asli
- $\hat{y}_i$: nilai prediksi

2.9 Mean Absolute Percentage Error

Mean Absolute Percentage Error atau MAPE merupakan perhitungan persentase dari *Mean Absolute Error*. MAPE menghitung rata-rata dari kesalahan absolut data dan menunjukkan nilainya dalam bentuk persentase [8]. Performa

model dapat disimpulkan berdasarkan indikator dari MAPE [8]. Indikator tersebut yaitu model mempunyai performa sangat akurat jika MAPE kurang dari 10, performa model akurat saat nilai MAPE berada di rentang 10 sampai dengan 20, model cukup akurat saat nilai MAPE berada di antara 20 sampai dengan 50, dan jika nilai MAPE berada di atas 50 maka model dapat dianggap mempunyai performa yang kurang baik.

Tabel 2.9. Interpretasi *Mean Absolute Percentage Error*

MAPE	Interpretasi
< 10	Sangat akurat
10-20	Akurat
20-50	Cukup akurat
> 50	Kurang akurat

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{2.14}$$

Keterangan:

- n: jumlah data
- y_i: nilai asli
- $\hat{y}_i$: nilai prediksi

2.10 Min-max Normalization

Min-max normalization adalah teknik normalisasi yang digunakan untuk data yang bertipe numerik [26]. Teknik ini diperlukan ketika banyak data memiliki rentang angka yang sangat berbeda. *Min-max normalization* mengubah data menjadi rentang angka nol sampai satu.

$$X_{\text{Normalize}} = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}} \tag{2.15}$$

2.11 Grid Search

Grid search merupakan metode optimasi *hyperparameter* pada model *deep learning*. Metode ini bekerja dengan membuat percobaan penggunaan parameter untuk mencari parameter yang paling optimal pada model berjalan secara otomatis [27]. *Grid search* mengeluarkan hasil yang berisi parameter terbaik yang dapat digunakan pada suatu model. Metode ini mempunyai kelebihan dapat menemukan kombinasi parameter terbaik karena melakukan iterasi pengujian ke seluruh kombinasi yang ada dibanding dengan metode lain seperti *Random Search* yang hanya menguji kombinasi secara acak sehingga tidak mendapat kombinasi terbaik secara menyeluruh.

2.12 Holdout Validation

Metode *holdout validation* merupakan salah satu metode validasi model yang dapat digunakan untuk model *machine learning*. Metode ini sangat efisien saat menggunakan data yang banyak karena dapat menghindari pelatihan model yang dilakukan berulang seperti ketika menggunakan K-fold CV.

Metode *holdout validation* membagi dataset menjadi *training*, *validation*, dan *testing*. Data *training* digunakan saat pelatihan model dengan memperbarui *weight* dan *bias* pada model. Data *validation* berguna untuk *tuning hyperparameter* saat proses *Grid Search*. Data *testing* untuk menguji model setelah pelatihan model untuk mengetahui kinerja model terhadap data yang belum pernah dilihat sebelumnya. Saat pembagian dataset, data *testing* harus cukup dalam menggambarkan penyebaran data namun juga lebih kecil dari ukuran data training [28]. Rasio pembagian yang bisa dipakai adalah 80 persen *training* dengan 20 persen *testing*, 70 persen *training* dengan 30 persen *testing* [28]. Jika ingin menggunakan data validasi maka dataset dapat dibagi menjadi 80-10-10 yang berarti 80 persen *training*, 10 persen *validation*, dan 10 persen *testing* [29]. Pembagian 10 sampai 20 persen *testing* biasanya sering dianggap cukup dalam sistem pembelajaran mesin [30].