BAB 2
LANDASAN TEORI

2.1 Perbandingan dengan Penelitian Terdahulu

Penelitian-penelitian sebelumnya yang dijadikan acuan oleh penelitian ini berkaitan dengan masalah prediksi harga saham dan implementasi jaringan saraf tiruan *LSTM*. Penelitian ini memiliki beberapa persamaan dan perbedaan dengan penelitian sebelumnya, yang dapat dilihat dalam Tabel 2.1

Tabel 2.1 Perbandingan dengan penelitian terdahulu

No.	Peneliti	Judul	Persamaan	Perbedaan
1	Ahmad Fauzi (2019)	*Forecasting* Syariah dengan Menggunakan LSTM	-Memakai *Long Short Term Memory* (*LSTM*) *Network* untuk membuat prediksi harga saham -Memakai data harga saham historis -Memakai ukuran dataset pelatihan berupa 90% dari dataset keseluruhan dan 10% yang tersisa sebagai dataset pengujian.	-Hiper-parameter yang diuji berupa dampak no. Epoch pelatihan *LSTM Network* terhadap performa jaringan, sedangkan penelitian ini mengukur dampak ukuran dataset pelatihan *LSTM Network* terhadap performa jaringan.
2	Adhib Arfan & Lussiana ETP (2019)	Prediksi Harga Saham di Indonesia menggunakan Algoritma *Long Short-Term Memory*	-Memakai *Long Short Term Memory* (*LSTM*) *Network* untuk membuat prediksi harga saham -Memakai data harga saham historis. -Memakai ukuran dataset pelatihan berupa 90% dari dataset keseluruhan dan 10% yang tersisa sebagai dataset pengujian.	-Hiper-parameter yang diuji berupa jumlah *hidden layer*, jumlah *neuron* pada *hidden layer*, no. *epoch,* dan *batch size*, sedangkan penelitian ini mengukur dampak ukuran dataset pelatihan *LSTM Network* terhadap performa jaringan.
3	Prismahardi Aji Riyantoko, Tresn Maulana Fahruddin, Kartika Maulida Hindrayani, Eristya Maya Safitri (2020	Analisis Prediksi Harga Saham Sektor Perbankan Menggunakan Algoritma *Long-Short Terms Memory* (*LSTM*)	-Memakai *Long Short Term Memory* (*LSTM*) *Network* untuk membuat prediksi harga saham -Memakai data harga saham historis -Memakai ukuran dataset pelatihan berupa 80% dari dataset keseluruhan dan 20% yang tersisa sebagai dataset pengujian.	-Hiper-parameter yang diuji berupa dampak no. Epoch pelatihan *LSTM Network* t terhadap performa jaringan, sedangkan penelitian ini mengukur dampak ukuran dataset pelatihan *LSTM Network* terhadap performa jaringan.
4	Roby Julian, Muhammad Rizky Pribadi (2021)	Peramalan Harga Saham Pertambangan Pada Bursa Efek Indonesia (BEI) Menggunakan *Long Short Term Memory* (*LSTM*)	-Memakai *Long Short Term Memory* (*LSTM*) *Network* untuk membuat prediksi harga saham -Memakai data harga saham historis -Memakai ukuran dataset pelatihan berupa 80% dari dataset keseluruhan dan 20% yang tersisa sebagai dataset pengujian.	-Hiper-parameter yang diuji berupa dampak no. Epoch pelatihan *LSTM Network* terhadap performa jaringan, sedangkan penelitian ini mengukur dampak ukuran dataset pelatihan *LSTM Network* terhadap performa jaringan.

Tabel 2.1 Perbandingan dengan penelitian terdahulu (lanjutan)

No.	Peneliti	Judul	Persamaan	Perbedaan
5	Adhib Arfan, Lussiana ETP (2020)	Perbandingan Algoritma *Long Short-Term Memory* dengan *SVR* pada Prediksi Harga Saham di Indonesia	-Memakai *Long Short Term Memory* (*LSTM*) *Network* untuk membuat prediksi harga saham -Memakai data harga saham historis	-Memakai Algoritma *Support Vector Regression* (SVR) untuk membuat prediksi harga saham. -Penelitian sebelumnya membandingkan *LSTM Network* terhadap *SVR*, sedangkan penelitian ini mengukur dampak ukuran dataset pelatihan terhadap performa jaringan *LSTM Network*
6	Widi Hastomo, Adhitio Satyo Bayangkari Karno, Nawang Kalbuana, Ervina Nisfiani, Lussiana ETP (2021)	Optimasi *Deep Learning* untuk Prediksi Saham di Masa Pandemi *Covid-19*	-Memakai *Long Short Term Memory* (*LSTM*) *Network* untuk membuat prediksi harga saham -Memakai data harga saham historis	-Memakai *Gated Recurrent Unit* (*GRU*) untuk membuat prediksi harga saham. -Tujuan yang ingin dicapai adalah kombinasi arsitektur mana antara *LSTM* dan *GRU* yang memberikan hasil prediksi harga saham yang paling akurat, sedangkan penelitian ini mengukur dampak ukuran dataset pelatihan terhadap performa jaringan *LSTM Network*.

2.2. Prediksi Harga Saham

Dari awal terbentuknya pasar harga saham, telah terjadi perdebatan konstan tentang prediktabilitas pengembalian saham. Fama (1970) memperkenalkan *Efficient Market Hypothesis*, yang menyatakan bahwa harga saat ini dari sebuah aset selalu mencerminkan semua informasi sebelumnya yang tersedia untuk aset tersebut secara langsung. Selain itu, terdapat juga *Random-Walk Hypothesis* oleh Malkiel (1973), yang mengklaim bahwa harga saham berubah secara independen dari nilainya di masa lalu. Dengan kata lain, harga saham besok hanya akan tergantung dengan informasi yang akan tersedia di keesokan harinya, bukan harga saham hari ini. Biondo et al (2013) melakukan serangkaian percobaan yang menunjukkan bahwa strategi acak dapat mengungguli beberapa metode perdagangan teknis yang paling klasik, seperti *Moving Average Convergence Divergence* (*MACD*) dan *Relative Strength Index* (*RSI*).

Di sisi lain, terdapat juga banyak penulis yang mengklaim bahwa sebenarnya harga saham dapat diprediksi setidaknya sampai tingkat tertentu (Lo & MacKinlay, 1999). Dan berbagai metode untuk memprediksi dan memodelkan perilaku saham telah menjadi objek studi berbagai bidang studi, seperti ekonomi, statistik, dan ilmu

komputer. Di tahun 2012 sendiri, diperkirakan 85% perdagangan di pasar saham Amerika Serikat dilakukan oleh algoritma (Glantz & Kissel, 2013).

Prediksi pasar saham secara tradisional merupakan salah satu tugas prediksi deret waktu yang paling menantang, karena data saham berisik dan tidak stasioner. Prediksi pengembalian saham secara luas diklasifikasikan ke dalam model linier dan non-linier: model rata-rata bergerak terintegrasi autoregresif linier, model pemulusan eksponensial, dan model heteroskedastisitas bersyarat autoregresif umum (Cheng et al, 2018).

Salah satu metode yang paling popular untuk memodelkan dan memprediksi pasar saham adalah analisis teknis (*technical analysis*).

2.3 Analisis Teknis (*Technical Analysis*)

Di bidang keuangan, analisis teknis adalah metodologi analisis untuk meramalkan arah harga melalui studi data pasar masa lalu, terutama harga dan *Volume*. Analisis teknis menggunakan model dan aturan perdagangan berdasarkan transformasi harga dan *Volume*, seperti indeks kekuatan relatif, rata-rata bergerak, regresi, korelasi harga antar-pasar dan intra-pasar, siklus bisnis, siklus pasar saham atau, secara klasik, melalui pengenalan grafik pola.

Analisis teknis bekerja dengan menggunakan beberapa asumsi dasar, yaitu: (1) harga ditentukan secara eksklusif oleh hubungan penawaran-permintaan; (2) harga berubah mengikuti kecenderungan; (3) perubahan penawaran dan permintaan menyebabkan kecenderungan untuk berbalik; (4) perubahan pada penawaran dan permintaan dapat diidentifikasi pada grafik; Dan (5) pola pada grafik cenderung berulang [6]. Dengan kata lain, analisis teknis tidak memperhitungkan faktor eksternal seperti politik, sosial atau ekonomi makro.

Prinsip inti dari analisis teknis adalah bahwa harga pasar mencerminkan semua informasi relevan yang memengaruhi pasar itu. Oleh karena itu, seorang analis teknis melihat sejarah pola perdagangan keamanan atau komoditas daripada penggerak eksternal seperti peristiwa ekonomi, fundamental, dan berita. Diyakini bahwa aksi harga cenderung berulang karena perilaku kolektif dan terpola dari investor. Oleh karena itu, analisis teknis berfokus pada tren dan kondisi harga yang dapat diidentifikasi (Elder, 2008)

2.4 Jaringan Saraf Tiruan (*Artificial Neural Network*)

ANN pada awal mulanya dikembangkan sebagai model matematis kemampuan memproses informasi otak biologis (McCulloch & Pitts, 1988; Rumelhart et al., 1986). Struktur dasar dari *ANN* adalah sebuah jaringan yang terdiri dari beberapa unit pemrosesan, atau *node*, yang terhubung satu sama lain dengan koneksi berbobot. Rancangan ini dimaksudkan untuk meniru sel saraf (*neuron*) yang terdapat didalam otak manusia, dimana *node* mewakili *neuron* itu sendiri dan koneksi berbobot mewakili kekuatan sinapsis antar *neuron*. Jaringan diaktifkan dengan memberikan *input* ke beberapa atau semua *node*, dan aktivasi ini kemudian menyebar ke seluruh jaringan sepanjang koneksi berbobot (Graves, 2012)

ANN terbagi menjadi dua jenis, yaitu jaringan yang koneksinya membentuk siklus (*cyclic*) dan jaringan yang koneksinya tidak membentuk siklus (*acyclic*). *ANN* dengan siklus termasuk jaringan umpan-balik (*feedback*), jaringan rekursif (*recursive*), jaringan berulang (*recurrent*), dan lain-lain. *ANN* tanpa siklus disebut sebagai *Feedforward Neural Network* (*FNN*), dan beberapa jenis yang paling dikenal termasuk *perceptron* (Rosenblatt, 1958), jaringan *radial basis function* (Broomhead & Lowe, 1988), *Kohonen maps* (Kohonen, 1982), dan jaringan Hopfield (Hopfield, 1982).

Model non-linier mencakup *SVM*, algoritme genetika, dan *Deep Learning* yang canggih. Akita dkk (2016) membangun model *LSTM* menggunakan informasi tekstual dan numerik untuk memprediksi harga saham penutupan sepuluh perusahaan. Ding dkk (2015) mengusulkan *Deep CNN* menggunakan *event embeddings* yang menggabungkan pengaruh peristiwa jangka panjang dan peristiwa jangka pendek untuk memprediksi harga saham.

2.5 Pembelajaran Mendalam *(Deep Learning)*

Baru-baru ini, pembelajaran mendalam (*Deep Learning*) telah menjadi bidang penelitian yang luas dan produktif dengan terobosan sukses di beberapa bidang ilmiah dan bisnis, seperti pengenalan suara dan gambar (Krizhevsky dkk., 2012), pengemudi otomatis, bioinformasi genomik (Esteva dkk., 2017; Domeniconi dkk., 2016), dan sebagainya. Efektivitas yang mengesankan dari solusi pembelajaran mendalam bergantung pada jaringan saraf baru dengan kemampuan memori.

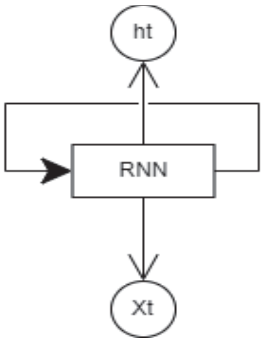
Teknik *Deep Learning* yang canggih termasuk jaringan saraf dalam (*Deep Neural Network / DNN*), jaringan saraf berulang (*Recurrent Neural Network / RNN*), dan jaringan saraf convolutional (*Convolutional Neural Network / CNN*). Jaringan dalam (*Deep Network*) dapat mengeksploitasi dalam arsitekturnya struktur khusus dari fungsi komposisi, sedangkan jaringan dangkal tidak mengetahui hal ini (Mhaskar dkk, 2017).

Khususnya jaringan saraf berulang (*RNN*) adalah teknologi pembelajaran konkret pertama yang mampu mendeteksi ketergantungan jangka panjang di antara contoh-contoh pelatihan, seperti di antara kata-kata berurutan dalam kalimat atau di antara nilai-nilai berurutan dalam deret waktu (Domeniconi dkk., 2017).

2.6 Recurrent Neural Network (RNN)

RNN adalah tipe *ANN* yang ditambah dengan penyertaan hubungan yang menjembatani langkah waktu yang berdekatan. Penyertaan hubungan ini memungkinkan jaringan untuk memproses data yang terurut secara temporal. Pada waktu *t*, node dengan pengulangan hubungan menerima input dari titik data saat ini $\boldsymbol{x}(t)$ dan juga dari simpul tersembunyi nilai $\boldsymbol{h}(t-1)$ dalam keadaan jaringan sebelumnya. Keluaran $\hat{\boldsymbol{y}}(t)$ pada setiap waktu *t* dihitung mengingat nilai *node* tersembunyi *h(t)* pada waktu *t*. Masukan $\boldsymbol{x}(t-1)$ pada waktu *t-1* dapat mempengaruhi output $\hat{y}(t)$ pada waktu *t* dan seterusnya dengan cara pengulangan koneksi (Lipton dkk, 2015).

Dua persamaan menentukan semua perhitungan yang diperlukan untuk perhitungan pada setiap *time step* di *forward pass* dalam sebuah *RNN* sederhana, seperti yang dapat dilihat pada Gambar 2.1.



1. **Gambar 2.1** Sebuah Recurrent Neural Network (RNN) sederhana

$$\boldsymbol{h}^{(t)} = \sigma\left(W^{hx}\boldsymbol{x}^{(t)} + W^{hh}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h\right) \quad \textbf{(2.1)}$$

Keterangan:

$\boldsymbol{h^{(t)}}$: *Hidden state time step t*

W_{hx} : Bobot sampel input

$\boldsymbol{x^{(t)}}$: Sampel input *time step t*

W^{hh} : Bobot *hidden state*

$\boldsymbol{h^{(t-1)}}$: *Hidden state time step t-1*

$\boldsymbol{b_h}$: Nilai bias *hidden state*

$$\hat{\boldsymbol{y}}^{(t)} = softmax(W^{yh}\boldsymbol{h}^{(t)} + \boldsymbol{b}_y) \quad \textbf{(2.2)}$$

Keterangan:

$\hat{\boldsymbol{y}}^{(t)}$: Output *RNN* pada *time step t*

W^{yh} : Bobot lapisan *output*

$\boldsymbol{h}^{(t)}$: *Hidden state* pada *time step t*

$\boldsymbol{b}_y$: Bias lapisan *output*

Arsitektur ini sangat sesuai dengan sifat deret waktu dari harga saham data (Eapen dkk, 2019). Sebuah hierarki *Deep RNN* terbukti berfungsi baik untuk tugas prediksi yang terkait dengan data deret waktu (Herman & Schrauwen, 2013).

Salah satu kelemahan dari *RNN* adalah masalah *vanishing/exploding gradient* (Bengio dkk, 1994; Hochreiter dkk, 2001). Masalah *exploding gradient* mengacu pada peningkatan besar dalam norma gradien selama pelatihan. Peristiwa ini disebabkan dari penggunaan jangka waktu yang lebih besar untuk input sampel, yang dapat tumbuh secara eksponensial lebih dari jangka waktu yang lebih pendek. Masalah *vanishing gradient* mengacu kepada perilaku yang bertolak belakang, yaitu penurunan besar dalam norma gradien menuju nilai nol, sheingga model tidak lagi dapat mempelajari korelasi antara peristiwa-peristiwa yang terpisah jauh secara temporal (Pascanu dkk, 2013)

2.7. Long Short Term Memory (LSTM) Network

Salah satu solusi untuk masalah *exploding/vanishing gradient* dalam *RNN* adalah dengan penambahan *constant error carousel* dalam unit memori *RNN*, serta gerbang multiplikatif yang menentukan apakah suatu informasi relevan atau tidak terhadap keluaran jaringan secara keseluruhan. Modifikasi *RNN* ini disebut *Long Short Term Memory* (*LSTM*) *Network*, dan pertama kali digagaskan oleh Hochreiter and Schmidhuber pada tahun 1997.

Istilah *Long Short Term Memory* berasal dari intuisi berikut. *RNN* sederhana memiliki memori jangka panjang dalam bentuk bobot-bobot. Bobot-bobot berubah perlahan selama proses pelatihan, menyandikan pengetahuan umum dari data input kedalam jaringan. *RNN* juga memiliki memori jangka pendek dalam bentuk aktivasi-aktivasi singkat, yang berpindah dari setiap *node* ke *node* selanjutnya secara berurutan. Model *LSTM* memperkenalkan jenis penyimpanan memori perantara melalui sel memori (*memory cell*), yang mampu mempertahankan keadaannya dari waktu ke waktu, dan unit-unit gerbang (*gate*) non-linier yang mengatur aliran informasi masuk dan keluar dari sel memori (Greff dkk, 2015).

Secara formal, perhitungan-perhitungan didalam sebuah model *LSTM* berlangsung mengikuti persamaan-persamaan berikut, yang dilakukan untuk setiap *time step*:

$$\begin{gathered} \boldsymbol{g}^{(t)} = \phi(W^{gx} \cdot \boldsymbol{x}^{(t)} + W^{gh} \cdot \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_g) \\ x_c{}^{(t)} = \left[\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t-1)}\right] \\ W_g{}^{(t)} = [W^{gx}, W^{gh}] \\ \boldsymbol{g}^{(t)} = \phi(x_c{}^{(t)} \cdot W_g{}^{(t)} + \boldsymbol{b}_g) \end{gathered} \tag{2.3}$$

Keterangan:

$\boldsymbol{g}^{(t)}$: Nilai *candidate gate LSTM* pada *time step* (*t*)

$\cdot$: *Dot product*

ϕ : Fungsi *hyperbolic tangent* (tanh)

W^{gx} : Bobot *candidate gate* untuk sampel input *x*

$\boldsymbol{x}^{(t)}$: Sampel input *x* pada *time step* (*t*)

W^{gh} : Bobot *hidden state* untuk *candidate gate*

$\boldsymbol{h}^{(t-1)}$: *Hidden state* pada *time step* (*t-1*)

$x_c{}^{(t)}$: Matriks input hasil konkatenasi $\boldsymbol{x}^{(t)}$ dan $\boldsymbol{h}^{(t-1)}$

$W_g{}^{(t)}$: Matriks bobot hasil konkatenasi W^{gx} dan W^{gh}

$\boldsymbol{b}_g$: Bias *candidate gate*

$$\begin{gathered} \boldsymbol{i}^{(t)} = \sigma(W^{ix} \cdot \boldsymbol{x}^{(t)} + W^{ih} \cdot \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i) \\ x_c{}^{(t)} = \left[\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t-1)}\right] \\ W_i{}^{(t)} = \left[W^{ix}, W^{ih}\right] \\ \boldsymbol{i}^{(t)} = \sigma(x_c{}^{(t)} \cdot W_i{}^{(t)} + \boldsymbol{b}_i) \end{gathered} \tag{2.4}$$

Keterangan:

$\boldsymbol{i}^{(t)}$: Nilai *input layer LSTM* pada *time step* (*t*)

$\cdot$: *Dot product*

σ : Fungsi *logistic* (sigmoid)

W^{ix} : Bobot *input layer* untuk sampel input *x*

$\boldsymbol{x}^{(t)}$: Sampel input *x* pada *time step* (*t*)

W^{ih} : Bobot *input layer* untuk *hidden state*

$\boldsymbol{h}^{(t-1)}$: *Hidden state* pada *time step* (*t-1*)

$x_c{}^{(t)}$: Matriks input hasil konkatenasi $\boldsymbol{x}^{(t)}$ dan $\boldsymbol{h}^{(t-1)}$

$W_i{}^{(t)}$: Matriks bobot hasil konkatenasi W^{ix} dan W^{ih}

$\boldsymbol{b}_i$: Bias *input layer gate*

$$\boldsymbol{f}^{(t)} = \sigma(W^{fx} \cdot \boldsymbol{x}^{(t)} + W^{fh} \cdot \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f)$$
$$x_c{}^{(t)} = \left[\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t-1)}\right]$$
$$W_f{}^{(t)} = [W^{fx}, W^{fh}]$$
$$\boldsymbol{f}^{(t)} = \sigma(x_c{}^{(t)} \cdot W_f{}^{(t)} + \boldsymbol{b}_{f)} \quad \textbf{(2.5)}$$

Keterangan:

$\boldsymbol{f}^{(t)}$: Nilai *forget gate LSTM* pada *time step* (*t*)

$\cdot$: *Dot product*

σ : Fungsi *logistic* (sigmoid)

W^{fx} : Bobot *forget gate* untuk sampel input *x*

$\boldsymbol{x}^{(t)}$: Sampel input *x* pada *time step* (*t*)

W^{fh} : Bobot *forget gate* untuk *hidden state*

$\boldsymbol{h}^{(t-1)}$: *Hidden state* pada *time step* (*t-1*)

$x_c{}^{(t)}$: Matriks input hasil konkatenasi $\boldsymbol{x}^{(t)}$ dan $\boldsymbol{h}^{(t-1)}$

$W_f{}^{(t)}$: Matriks bobot hasil konkatenasi W^{fx} dan W^{fh}

$\boldsymbol{b}_f$: Bias *forget gate*

$$\boldsymbol{o}^{(t)} = \sigma(W^{ox} \cdot \boldsymbol{x}^{(t)} + W^{oh} \cdot \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o)$$
$$x_c{}^{(t)} = \left[\boldsymbol{x}^{(t)}, \boldsymbol{h}^{(t-1)}\right]$$
$$W_o{}^{(t)} = [W^{ox}, W^{oh}]$$
$$\boldsymbol{o}^{(t)} = \sigma(x_c{}^{(t)} \cdot W_o{}^{(t)} + \boldsymbol{b}_{o)} \quad \textbf{(2.6)}$$

Keterangan:

$\boldsymbol{o}^{(t)}$: Nilai *output gate LSTM* pada *time step* (*t*)

$\cdot$: *Dot product*

σ : Fungsi *logistic* (sigmoid)

W^{ox} : Bobot *output gate* sampel input *x*

$\boldsymbol{x}^{(t)}$: Sampel input *x* pada *time step* (*t*)

W^{oh} : Bobot *output gate* untuk *hidden state*

$\boldsymbol{h}^{(t-1)}$: *Hidden state* pada *time step* (*t-1*)

$x_c{}^{(t)}$: Matriks input hasil konkatenasi $\boldsymbol{x}^{(t)}$ dan $\boldsymbol{h}^{(t-1)}$

$W_o{}^{(t)}$: Matriks bobot hasil konaktenasi W^{ox} dan W^{oh}

$\boldsymbol{b}_o$: Bias *output gate*

$$\boldsymbol{s}^{(t)} = \boldsymbol{g}^{(t)} \odot \boldsymbol{i}^{(t)} + \boldsymbol{s}^{(t-1)} \odot \boldsymbol{f}^{(t)} \quad \textbf{(2.7)}$$

Keterangan:

$\boldsymbol{s}^{(t)}$: Nilai *cell state* pada *time step* (*t*)

$\odot$: Perkalian *element-by-element* (*Hadamard product*)

$\boldsymbol{g}^{(t)}$: Nilai *forget gate LSTM* pada *time step* (*t*)

$\boldsymbol{i}^{(t)}$: Nilai *input layer LSTM* pada *time step* (*t*)

$\boldsymbol{s}^{(t-1)}$: Nilai *cell state* pada *time step* (*t-1*)

$\boldsymbol{f}^{(t)}$: Nilai *forget gate LSTM* pada *time step* (*t*)

$$\boldsymbol{h}^{(t)} = \phi\left(\boldsymbol{s}^{(t)}\right) \odot \boldsymbol{o}^{(t)} \quad \textbf{(2.8)}$$

Keterangan:

$\boldsymbol{h}^{(t)}$: Nilai *hidden state* pada *time step* (*t*)

$\odot$: Perkalian *element-by-element* (*Hadamard product*)

ϕ : Fungsi *hyperbolic tangent* (tanh)

$\boldsymbol{s}^{(t)}$: Nilai *cell state* pada *time step* (*t*)

$\boldsymbol{o}^{(t)}$: Nilai *output gate LSTM* pada *time step* (*t*)

LSTM telah dipakai di banyak bidang untuk menyelesaikan berbagai macam masalah, seperti pengenalan tulisan tangan (Graves dkk, 2008; Pham dkk, 2014; Doetsch dkk, 2014) serta penulisannya (Graves, 2013), pemodelan bahasa (Zaremba dkk, 2014) dan penerjemahan (Luong dkk, 2014), pemodelan akustik ucapan (Sak dkk, 2014), sintesis ucapan (Fan dkk, 2014), analisis data audio (Marchi dkk, 2014) dan video (Donahue dkk, 2014), dan seterusnya.

LSTM juga digunakan dalam penelitian subjek prediksi pasar saham. Bao dkk (2017) menggunakan kombinasi *Wavelet Transforms* (*WT*), *Stacked Autoencoders*

(*SAEs*) dan *Long Short Term Memory* (*LSTM*) untuk membuat peramalan harga saham.

2.8 *Backpropagation Through Time* (*BPTT*)

Untuk *backpropagation*, dimisalkan sebuah fungsi *loss* *l*(*t*) dimana nilainya ingin dikecilkan pada setiap *time step t* yang bergantung pada nilai *hidden state* (*h*(*t*)) dan pada nilai target (*y(t)*) pada waktu sekarang melalui sebuah fungsi *loss f*, yang didapat dilihat pada Persamaan 2.9*:*

$$l^{(t)} = f\left(h^{(t)}, y^{(t)}\right) \quad \textbf{(2.9)}$$

Keterangan:

$l^{(t)}$: Fungsi *loss*

$h^{(t)}$: Nilai hasil prediksi

$y^{(t)}$: Nilai target

Dimana f dapat berupa fungsi *loss* apapun yang dapat diturunkan, seperti misalnya *Euclidean loss*, yang dapat dilihat pada Persamaan 2.10:

$$l^{(t)} = f\left(h^{(t)}, y^{(t)}\right) = \left(h^{(t)}, y^{(t)}\right)^2 \quad \textbf{(2.10)}$$

Keterangan:

$l^{(t)}$: Fungsi *loss*

$h^{(t)}$: Nilai hasil prediksi

$y^{(t)}$: Nilai target

Untuk jaringan rekursif seperti *LSTM*, penambahan dimensi waktu dalam perhitungan berarti nilai *loss* yang perlu dikurangi adalah jumlah nilai *loss* untuk semua *time step*, yang direpresentasikan dengan simbol *L*, untuk sebuah urutan nilai-nilai dengan panjang *T*, yang dapat dilihat pada Persamaan 2.11:

$$L = \sum_{t=1}^{T} l^{(t)} \quad \textbf{(2.11)}$$

Keterangan:

L : Nilai *loss* untuk sebuah urutan nilai-nilai dengan panjang *T*

$l^{(t)}$: Fungsi *loss*

Nilai prediksi didapatkan dari proses *forward pass* jaringan yang digunakan, dimana parameter-parameter jaringan yang dapat mempengaruhi hasil prediksi berupa bobot jaringan dan bias jaringan tersebut, yang direpresentasikan dengan

(2.12)

menggunakan simbol θ. Untuk pendekatan pengubahan bobot dan bias . Ini dapat dilihat pada Persamaan 2.12:

$$\frac{dL}{dh_i^{(t)}} = \sum_{t=1}^{T}\sum_{i=1}^{M}\frac{dL}{dh_i^{(t)}}\frac{dh_i^{(t)}}{d\theta}$$

Keterangan:

$\frac{dL}{dh_i^{(t)}}$: Turunan nilai *loss* untuk sebuah urutan nilai-nilai dengan panjang *T* terhadap *hidden state LSTM Node* pada *time step t*

$\frac{dh_i^{(t)}}{d\theta}$: Turunan *hidden state LSTM Node* pada *time step t* terhadap parameter *LSTM Node* (Cth: Bobot, bias)

Dimana $h_i^{(t)}$ adalah skalar yang sesuai dengan nilai *hidden state* sel memori ke-*i* dan *M* adalah jumlah total *memory cell*. Karena *LSTM Network* merambatkan informasi kedepan dalam waktu, mengubah nilai $h_i^{(t)}$ tidak akan mempengaruhi nilai *loss* sebelum dari waktu *t*, yang memungkinkan penulisan Persamaan 2.13:

$$\frac{dL}{dh_i^{(t)}} = \sum_{s=1}^{T}\frac{dl^{(s)}}{dh_i^{(t)}} = \sum_{s=t}^{T}\frac{dl^{(s)}}{dh_i^{(t)}} \qquad \textbf{(2.13)}$$

Keterangan:

$\frac{dL}{dh_i^{(t)}}$: Turunan nilai *loss* untuk sebuah urutan nilai-nilai dengan panjang *T* terhadap *hidden state LSTM Node* pada *time step t*

$\frac{dl^{(s)}}{dh_i^{(t)}}$: Turunan fungsi *loss* pada *cell state s* terhadap *hidden state LSTM Node* pada *time step t*

Untuk memudahkan notasi perumusan, sebuah variabel *L(t)* akan digunakan untuk merepresentasikan nilai *loss* kumulatif dari *time step t* selanjutnya, seperti yang dapat dilihat pada Persamaan 2.14:

$$L^{(t)} = \sum_{s=t}^{s=T} l^{(s)} \qquad \textbf{(2.14)}$$

Keterangan:

$L^{(t)}$: Nilai *loss* kumulatif dari *time step t* selanjutnya

$l^{(s)}$: Fungsi *loss*

Dengan perumusuan ini, nilai dari *L(1)* adalah nilai *loss* untuk semua nilai input, dari awal sampai akhir. Hal ini memungkinkan penulisan Persamaan 2.15:

$$\frac{dL}{dh_i(t)} = \sum_{s=t}^{T} \frac{dl(s)}{dh_i(t)} = \frac{dL(t)}{dh_i(t)} \quad \textbf{(2.15)}$$

Keterangan:

$\frac{dL}{dh_i(t)}$: Turunan nilai *loss* kumulatif dari *time step t* selanjutnya terhadap *hidden state LSTM Node* pada *time step t*

$\frac{dl(s)}{dh_i(t)}$: Turunan fungsi *loss* pada *cell state s* terhadap *hidden state LSTM Node* pada *time step t*

$\frac{dL(t)}{dh_i(t)}$: Turunan nilai *loss* kumulatif dari *time step t* selanjutnya pada *time step t* terhadap *hidden state LSTM Node* pada *time step t*

Tujuan utama penurunan ini adalah untuk menghitung gradien parameter-parameter *LSTM Network* yang mempengaruhi hasil prediksi dan dapat dilatih, seperti nilai bobot, nilai bias, dst. Simbol θ digunakan untuk merepresentasikan parameter-parameter tersebut, dan dengan tambahan ini, perhitungan gradien dapat ditulis ulang menjadi Persamaan 2.16:

$$\frac{dL}{d\theta} = \sum_{t=1}^{T} \sum_{i=1}^{M} \frac{dL(t)}{dh_i(t)} \frac{dh_i(t)}{d\theta} \quad \textbf{(2.16)}$$

Keterangan:

$\frac{dL}{d\theta}$: Turunan nilai *loss* kumulatif dari *time step t* selanjutnya terhadap parameter skalar *LSTM Network* (Cth: Bobot, bias)

$\frac{dL(t)}{dh_i(t)}$: Turunan nilai *loss* kumulatif dari *time step t* selanjutnya pada *time step t* terhadap *hidden state LSTM Node* pada *time step t*

$\frac{dh_i(t)}{d\theta}$: Turunan *hidden state LSTM Node* pada *time step t* terhadap parameter skalar *LSTM Network* (Cth: Bobot, bias)

Variabel *L(t)* memungkinkan ekspresi rekursi pada Persamaan 2.17 untuk dipaparkan:

$$L(t) = \begin{cases} l(t) + L(t+1) & if\ t < T \\ l(t) & if\ t = T \end{cases} \quad \textbf{(2.17)}$$

Keterangan:

$L^{(t)}$: Nilai *loss* kumulatif dari *time step t* selanjutnya pada *time step t*

$l(t)$: Fungsi *loss* pada *time step t*

t : Nilai *time step*

T : Panjang sebuah urutan nilai-nilai

Maka, untuk aktivasi *hidden state* oleh sebuah *LSTM Node* pada waktu *t*, didapatkan Persamaan 2.18:

$$\frac{dL(t)}{dh(t)} = \frac{dl(t)}{dh(t)} + \frac{dL(t+1)}{dh(t)} \quad \textbf{(2.18)}$$

Keterangan:

$\frac{dL(t)}{dh(t)}$: Turunan nilai *loss* kumulatif dari *time step t* selanjutnya pada *time step t* terhadap *hidden state LSTM Node* pada *time step t*

$\frac{dl(t)}{dh(t)}$: Turunan fungsi *loss* pada *time step t* terhadap *hidden state LSTM Node* pada *time step t*

Dimana $\frac{dl(t)}{dh(t)}$ merupakan turunan *element-wise* dari *loss l(t)* terhadap aktivasi (*h(t)*) pada waktu *t*. Sedangkan untuk $\frac{dL(t+1)}{dh(t)}$, nilai ini merupakan bagian dari sifat berulang (*recurrent*) *LSTM Network*. Perhitungan nilai $\frac{dL(t+1)}{dh(t)}$ hanya dapat didapatkan dengan menggunakan nilai turunan *LSTM Node* yang posisinya berada di masa depan dari *LSTM Node* yang sedang dikomputasikan. Karena pada akhirnya $\frac{dL(t)}{dh(t)}$ perlu dihitung untuk semua nilai *t* = 1, ..., *T*, maka untuk *LSTM Node* terakhir dalam *LSTM Network*, nilai *loss* dihitung dengan Persamaan 2.19:

$$\frac{dL^{(t)}}{dh^{(t)}} = \frac{dl^{(t)}}{dh^{(t)}} \quad \textbf{(2.19)}$$

Keterangan:

$\frac{dL(t)}{dh(t)}$: Turunan nilai *loss* kumulatif dari *time step t* selanjutnya pada *time step t* terhadap *hidden state LSTM Node* pada *time step t*

$\frac{dl(t)}{dh(t)}$: Turunan fungsi *loss* pada *time step t* terhadap *hidden state LSTM Node* pada *time step t*

Dan seterusnya nilai ini digunakan untuk perhitungan-perhitungan selanjutnya mundur melalui jaringan (Jimenez, 2014). Berikut adalah algoritma

(2.20)

backpropagation through time (*BPTT*) untuk satu *time step*, dengan menggunakan fungsi *loss Mean Squared Error* (*MSE*) sebagai fungsi *loss f*:

$$\frac{dL^{(t)}}{dh^{(t)}} = f'\left(\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2\right) = \frac{1}{n}\sum_{i=1}^{n} -2\left(Y_i - \widehat{Y}_i\right)$$

$$\frac{dL^{(t)}}{dh^{(t)}} = \frac{dL^{(t)}}{dh^{(t)}} + \frac{dL^{(t+1)}}{dh^{(t+1)}}$$

$$\frac{dL^{(t)}}{do^{(t)}} = \frac{dL^{(t)}}{dh^{(t)}}\frac{dh^{(t)}}{do^{(t)}} = \frac{dL^{(t)}}{dh^{(t)}} \odot \phi\left(\boldsymbol{s}^{(t)}\right)$$

$$\frac{dL^{(t)}}{ds^{(t)}} = \frac{dL^{(t)}}{dh^{(t)}}\frac{dh^{(t)}}{ds^{(t)}} + \frac{dL^{(t+1)}}{ds^{(t)}} = \frac{dL^{(t+1)}}{ds^{(t)}} + \frac{dL^{(t)}}{dh^{(t)}} \odot \boldsymbol{o}^{(t)} \odot \left(1 - \left(\phi\left(\boldsymbol{s}^{(t)}\right)\right)^2\right.$$

$$\frac{dL^{(t)}}{di^{(t)}} = \frac{dL^{(t)}}{ds^{(t)}}\frac{ds^{(t)}}{di^{(t)}} = \frac{dL^{(t)}}{ds^{(t)}} \odot \boldsymbol{g}^{(t)}$$

$$\frac{dL^{(t)}}{dg^{(t)}} = \frac{dL^{(t)}}{ds^{(t)}}\frac{ds^{(t)}}{dg^{(t)}} = \frac{dL^{(t)}}{ds^{(t)}} \odot \boldsymbol{i}^{(t)}$$

$$\frac{dL^{(t)}}{df^{(t)}} = \frac{dL^{(t)}}{ds^{(t)}}\frac{ds^{(t)}}{df^{(t)}} = \frac{dL^{(t)}}{ds^{(t)}} \odot \boldsymbol{s}^{(t-1)}$$

$$\frac{dL^{(t)}}{ds^{(t-1)}} = \frac{dL^{(t)}}{ds^{(t)}}\frac{ds^{(t)}}{ds^{(t-1)}} = \frac{dL^{(t)}}{ds^{(t)}} \odot \boldsymbol{f}^{(t)}$$

$$\frac{dL^{(t)}}{dwo^{(t)}} = \frac{dL^{(t)}}{do^{(t)}}\frac{do^{(t)}}{dwo^{(t)}} = \left(\frac{dL^{(t)}}{do^{(t)}} \odot \boldsymbol{o}^{(t)} \odot \left(1 - \boldsymbol{o}^{(t)}\right)\right) \otimes {x_c}^{(t)}$$

$$\frac{dL^{(t)}}{dbo^{(t)}} = \frac{dL^{(t)}}{do^{(t)}}\frac{do^{(t)}}{dbo^{(t)}} = \frac{dL^{(t)}}{do^{(t)}} \odot \boldsymbol{o}^{(t)} \odot \left(1 - \boldsymbol{o}^{(t)}\right)$$

$$\frac{dL^{(t)}}{dwf^{(t)}} = \frac{dL^{(t)}}{df^{(t)}}\frac{df^{(t)}}{dwf^{(t)}} = \left(\frac{dL^{(t)}}{df^{(t)}} \odot \boldsymbol{f}^{(t)} \odot \left(1 - \boldsymbol{f}^{(t)}\right)\right) \otimes {x_c}^{(t)}$$

$$\frac{dL^{(t)}}{dbf^{(t)}} = \frac{dL^{(t)}}{df^{(t)}}\frac{df^{(t)}}{dbf^{(t)}} = \frac{dL^{(t)}}{df^{(t)}} \odot \boldsymbol{f}^{(t)} \odot \left(1 - \boldsymbol{f}^{(t)}\right)$$

$$\frac{dL^{(t)}}{dwi^{(t)}} = \frac{dL^{(t)}}{di^{(t)}}\frac{di^{(t)}}{dwi^{(t)}} = \left(\frac{dL^{(t)}}{di^{(t)}} \odot \boldsymbol{i}^{(t)} \odot \left(1 - \boldsymbol{i}^{(t)}\right)\right) \otimes {x_c}^{(t)}$$

$$\frac{dL^{(t)}}{dbi^{(t)}} = \frac{dL^{(t)}}{di^{(t)}}\frac{di^{(t)}}{dbi^{(t)}} = \frac{dL^{(t)}}{di^{(t)}} \odot \boldsymbol{i}^{(t)} \odot \left(1 - \boldsymbol{i}^{(t)}\right)$$

$$\frac{dL^{(t)}}{dwg^{(t)}} = \frac{dL^{(t)}}{dg^{(t)}}\frac{dg^{(t)}}{dwg^{(t)}} = \left(\frac{dL^{(t)}}{dg^{(t)}} \odot \left(1 - (\boldsymbol{g}^{(t)})^{\mathbf{2}}\right)\right) \otimes {x_c}^{(t)}$$

(2.34)

(2.35)

$$\frac{dL^{(t)}}{dbg^{(t)}} = \frac{dL^{(t)}}{dg^{(t)}} \frac{dg^{(t)}}{dbg^{(t)}} = \frac{dL^{(t)}}{dg^{(t)}} \odot \left(1 - (\boldsymbol{g}^{(t)})^{\mathbf{2}}\right) \quad \textbf{(2.36)}$$

$$\frac{dL^{(t)}}{dxc^{(t)}} = \frac{dL^{(t)}}{dxc^{(t)}} + \left(\left(W_o{}^{(t)}\right)^T \odot \left(\frac{dL^{(t)}}{do^{(t)}} \odot \boldsymbol{o}^{(t)} \odot \left(1 - \boldsymbol{o}^{(t)}\right) \right) \right) \quad \textbf{(2.37)}$$

$$\frac{dL^{(t)}}{dxc^{(t)}} = \frac{dL^{(t)}}{dxc^{(t)}} + \left(\left(W_f{}^{(t)}\right)^T \odot \left(\frac{dL^{(t)}}{df^{(t)}} \odot \boldsymbol{f}^{(t)} \odot \left(1 - \boldsymbol{f}^{(t)}\right) \right) \right) \quad \textbf{(2.38)}$$

$$\frac{dL^{(t)}}{dxc^{(t)}} = \frac{dL^{(t)}}{dxc^{(t)}} + \left(\left(W_i{}^{(t)}\right)^T \odot \left(\frac{dL^{(t)}}{di^{(t)}} \odot \boldsymbol{i}^{(t)} \odot \left(1 - \boldsymbol{i}^{(t)}\right) \right) \right) \quad \textbf{(2.39)}$$

$$\frac{dL^{(t)}}{dxc^{(t)}} = \frac{dL^{(t)}}{dxc^{(t)}} + \left(\left(W_g{}^{(t)}\right)^T \odot \left(\frac{dL^{(t)}}{dg^{(t)}} \odot \left(1 - (\boldsymbol{g}^{(t)})^{\mathbf{2}}\right) \right) \right) \quad \textbf{(2.40)}$$

$$\frac{dL^{(t)}}{dh^{(t-1)}} = \frac{dL^{(t)}}{ds^{(t-1)}} \quad \textbf{(2.41)}$$

$$\frac{dL^{(t-1)}}{dh^{(t-1)}} = \frac{dL^{(t)}}{dxc^{(t)}} \quad \textbf{(2.42)}$$

Keterangan:

Y_i : Nilai observasi

$\widehat{Y}_i$: Nilai hasil prediksi

n : Jumlah observasi

$\boldsymbol{s}^{(t)}$: *Cell state LSTM Node* pada waktu *t*

$\boldsymbol{o}^{(t)}$: *Output gate LSTM Node* pada waktu *t*

$\boldsymbol{f}^{(t)}$: *Forget gate LSTM Node* pada waktu *t*

$\boldsymbol{i}^{(t)}$: *Input layer LSTM Node* pada waktu *t*

ϕ : Fungsi *hyperbolic tangent* (tanh)

$\odot$: Perkalian *element-by-element* (*Hadamard product*)

$\otimes$: Perkalian *outer product*

$\frac{dL^{(t)}}{dh^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap *hidden state LSTM Node* pada waktu *t*

$\frac{dL^{(t+1)}}{dh^{(t+1)}}$: Turunan fungsi *loss f* pada waktu *t+1* terhadap *hidden state LSTM Node* pada waktu *t+1*

$\frac{dL^{(t)}}{do^{(t)}}$: Turungan fungsi *loss f* pada awktu *t* terhadap *output gate LSTM Node* pada waktu *t*

$\frac{dh^{(t)}}{do^{(t)}}$: Turunan *hidden state LSTM Node* pada waktu *t* terhadap *output gate LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{ds^{(t)}}$: Turunan *fungsi loss f* pada waktu *t* terhadap *cell state LSTM Node* pada waktu *t*

$\frac{dh^{(t)}}{ds^{(t)}}$: Turunan *hidden state LSTM Node* pada waktu *t* terhadap *cell state LSTM Node* pada waktu *t*

$\frac{dL^{(t+1)}}{ds^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap *cell state LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{di^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap *input layer LSTM Node* pada waktu *t*

$\frac{ds^{(t)}}{di^{(t)}}$: Turungan fungsi *cell state LSTM Node* pada waktu *t* terhadap *input layer LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{dg^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap *candidate gate LSTM Node* pada waktu *t*

$\frac{ds^{(t)}}{dg^{(t)}}$: Turunan *cell state LSTM Node* pada waktu *t* terhadap *candidate gate LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{df^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap *forget gate LSTM Node* pada waktu *t*

$\frac{ds^{(t)}}{df^{(t)}}$: Turunan *cell state LSTM Node* pada waktu *t* terhadap *forget gate LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{ds^{(t-1)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap *cell state LSTM Node* pada waktu *t-1*

$\frac{ds^{(t)}}{ds^{(t-1)}}$: Turunan *cell state LSTM Node* pada waktu *t* terhadap *cell state LSTM Node* pada waktu *t-1*

$\frac{dL^{(t)}}{dwo^{(t)}}$: Turungan fungsi *loss f* pada waktu *t* terhadap bobot *output gate LSTM Node* pada waktu *t*

$\frac{do^{(t)}}{dwo^{(t)}}$: Turunan *output gate LSTM Node* pada waktu *t* terhadap bobot *output gate LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{dbo^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap bias *output gate LSTM Node* pada waktu *t*

$\frac{do^{(t)}}{dbo^{(t)}}$: Turunan *output gate LSTM Node* pada waktu *t* terhadap bias *output gate LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{dwf^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap bobot *forget gate LSTM Node* pada waktu *t*

$\frac{df^{(t)}}{dwf^{(t)}}$: Turunan *forget gate LSTM Node* pada waktu *t* terhadap bobot *forget gate LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{dbf^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap bias *forget gate LSTM Node* pada waktu *t*

$\frac{df^{(t)}}{dbf^{(t)}}$: Turunan *forget gate LSTM Node* pada waktu *t* terhadap bias *forget gate LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{dwi^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap bobot *input layer LSTM Node* pada waktu *t*

$\frac{di^{(t)}}{dwi^{(t)}}$: Turunan *input layer LSTM Node* pada waktu *t* terhadap bobot *input layer LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{dbi^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap bias *input layer LSTM Node* pada waktu *t*

$\frac{di^{(t)}}{dbi^{(t)}}$: Turunan *input layer LSTM Node* pada waktu *t* terhadap bias *input layer LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{dwg^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap bobot *candidate gate LSTM Node* pada waktu *t*

$\frac{dg^{(t)}}{dwg^{(t)}}$: Turunan *candidate gate LSTM Node* pada waktu *t* terhadap bobot *candidate gate LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{dbg^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap bias *candidate gate LSTM Node* pada waktu *t*

$\frac{dg^{(t)}}{dbg^{(t)}}$: Turunan *candidate gate LSTM Node* pada waktu *t* terhadap bias *candidate gate LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{dxc^{(t)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap input gabungan *LSTM Node* pada waktu *t*

$\frac{dL^{(t)}}{dh^{(t-1)}}$: Turunan fungsi *loss f* pada waktu *t* terhadap *hidden state LSTM Node* pada waktu *t-1*

$\frac{dL^{(t-1)}}{dh^{(t-1)}}$: Turunan fungsi *loss f* pada waktu *t-1* terhadap *hidden state LSTM Node* pada waktu *t-1*

2.9 *Gradient Descent*

Gradient Descent merupakan sebuah cara untuk meminimalkan sebuah fungsi tujuan $J(\theta)$ yang dibatasi oleh parameter model $\theta \in \mathbb{R}^d$, dimana θ dapat berupa bobot atau bias model, dengan memperbarui parameter dalam arah yang berlawanan dari gradien fungsi tujuan $J(\theta)$ sehubungan dengan parameter. Tingkat pembelajaran η menentukan ukuran langkah-langkah yang diambil untuk mencapai sebuah minima (lokal). Dengan kata lain, kita mengikuti arah kemiringan permukaan yang dibuat oleh fungsi objektif menuruni "bukit" sampai kita mencapai "lembah" (Ruder, 2017).

Persamaan untuk algoritma *Gradient Descent* yang dipakai dalam penelitian ini dapat dilihat pada Persamaan 2.42:

$$\theta_{baru} = \theta_{lama} - \eta \cdot \nabla_\theta J(\theta) \quad \textbf{(2.43)}$$

Keterangan:

θ : Parameter model (Cth: Bobot, bias)

η : Nilai *learning rate*

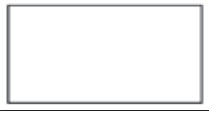
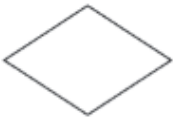
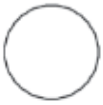
$\nabla_\theta J(\theta)$: Gradien parameter terhadap fungsi objektif

2.10 Bagan Alir (*Flowchart*)

Bagan alir (*flowchart*) atau diagram alir adalah suatu bentuk grafik atau diagram dari algoritma di mana simbol-simbol standar mewakili tampilan operasi yang perlu dan memperlihatkan urutan pelaksanaannya. Penggunaan bagan alir bertujuan untuk mempermudah penjelasan langkah-langkah dan urutan pengolahan data dari sistem yang akan dibuat.

Flowchart menggunakan beberapa simbol-simbol yang berbeda bentuk dan ukuran sebagai representasi suatu benda atau proses yang berlaku. Simbol-simbol tersebut telah distandardisasi oleh *American National Standards Institute* (*ANSI*). Daftar simbol *Flowchart* dokumen dapat dilihat pada Tabel 2.2.

Tabel 2.2 Daftar Simbol Flowchart

No.	Simbol	Nama	Keterangan
1.		Terminal	Menunjukan awal dan akhir dari bagan alir dokumen
2.		Proses	Perhitungan atau pemberian nilai ke sebuah variable
4.		*Input/Output* (I/O)	Merupakan kegiatan untuk memasukan data ke program atau keluaran dari program
5.		Keputusan	Menunjukan tahapan pembuatan keputusan
6.		*Predefined Process*	Kumpulan pernyataan/kegitan yang berfungsi untuk menyelesaikan satu tugas. Digunakan secara ekstensif ketika program dipecah menjadi modul-modul
7.		*Connector*	Berfungsi untuk menghubungkan *flowchart* ke *flowchart* lain
8.		*Flowlines* dan *Arrowheads*	Berguna untuk menghubungkan simbol-simbol dan merupakan indikasi urutan operasi
9.		Anotasi	Dapat digunakan untuk memberi komentar penjelasan

Sumber: Dixit, 2005

2.11 *Data Flow Diagram* (DFD)

Data Flow Diagram (*DFD*) merupakan gambaran suatu sistem yang telah ada atau sistem baru yang dikembangkan dengan menggunakan logika tanpa mempertimbangkan lingkungan fisik dimana sistem tersebut berada. Dengan adanya *DFD*, pengguna sistem dapat dengan lebih mudah memahami cara kerja sistem yang sedang berjalan.

Di dalam *DFD* terdapat 3 level, yaitu:

a) Diagram Level Nol (0), menggambarkan satu lingkaran besar yang dapat mewakili seluruh proses yang terdapat di dalam suatu sistem. Merupakan tingkatan tertinggi dalam *DFD*, biasanya diberi nomor nol. Semua entitas eksternal ditunjukkan pada diagram level ini berikut aliran-aliran data utama yang menuju dan keluar dari sistem. Diagram ini sama sekali tidak memuat penyimpanan data.
b) Diagram Level Satu (1), menggambarkan satu lingkaran besar yang mewakili lingkaran-lingkaran kecil yang ada di dalamnya. Merupakan pemecahan dari diagram level nol menjadi komponen-komponennya.
c) Diagram Level Dua (2), merupakan diagram yang menguraikan proses-proses yang ada dalam diagram level satu.

Simbol-simbol pada DFD dapat dilihat pada Tabel 2.3.

Tabel 2.3. Daftar Simbol DFD

Gane/Sarson	Yourdon/De Marco	Keterangan
Entitas Eksternal	Entitas Eksternal	Entitas eksternal, dapat berupa orang/unit terkait yang berinteraksi dengan sistem tetapi diluar sistem.
Proses	Proses	Orang, unit yang mempergunakan atau melakukan transformasi data. Komponen fisik tidak diidentifikasi.
Aliran Data	Aliran Data	Aliran data dengan arah khusus dari sumber ke tujuan.
Data Store	Data Store	Penyimpanan data atau tempat data ditunjuk oleh proses.

Sumber: Muslihudin, 2016 s

2.12 Normalisasi Data

Normalisasi Min-Max merupakan proses untuk mengatur data kedalam bentuk yang tepat digunakan dalam algoritma yang digunakan. Salah satu cara untuk dapat digunakan untuk normalisasi data adalah dengan Persamaan normalisasi *min-max*. Persamaan normalisasi *min-max* mengubah kisaran data sehingga nilai minimum data menjadi nilai nol dan nilai maksimum data menjadi nilai satu, sehingga skala data berada diantara nol dan satu (0, 1). Persamaan yang digunakan untuk normalisasi *min-max* dapat dilihat pada Persamaan 2.43:

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad \textbf{(2.43)}$$

Keterangan:

X_{sc} : Data setelah dinormalisasi

X : Data asli

X_{min} : Nilai minimum yang terdapat di dalam data

X_{max} : Nilai maksimum yang terdapat di dalam data

2.13 *Mean Square Error (MSE)*

Mean Square Error (*MSE*) adalah salah metrik yang sering digunakan untuk mengevaluasi performa suatu model regresi. *MSE* dihitung dengan menjumlahkan kuadrat perbedaan nilai hasil prediksi *LSTM Network* terhadap nilai target, atau *error* dari jaringan, dan kemudian membagi nilai *error* tersebut terhadap jumlah periode peramalan. Persamaan *MSE* dapat dilihat pada Persamaan 2.44:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2 \quad \textbf{(2.44)}$$

Keterangan:

MSE : Nilai rata-rata kuadrat *error* (perbedaan nilai hasil prediksi *LSTM Network* terhadap nilai target)

Y_i : Nilai target

$\widehat{Y}_i$: Nilai hasil prediksi *LSTM Network*

2.14 *Root Mean Square Error* (*RMSE*)

Root Mean Square Error (*RMSE*) adalah metrik yang sering digunakan untuk mengevaluasi performa suatu model regresi. Perhitungan *RMSE* hampir sama dengan *MSE*, dengan satu langkah tambahan, yaitu pengakaran nilai *MSE*. Persamaan *RMSE* dapat dilihat pada Persamaan 2.45:

(2.45)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2}$$

Keterangan:

$RMSE$: Nilai akar rata-rata kuadrat *error* (perbedaan nilai hasil prediksi *LSTM Network* terhadap nilai target)

Y_i : Nilai target

$\widehat{Y}_i$: Nilai hasil prediksi *LSTM Network*

2.15 *NumPy*

NumPy adalah proyek *open-source* yang bertujuan untuk memungkinkan komputasi numerik dengan bahasa pemrograman *Python*. NumPy diciptakan pada tahun 2005, diatas fondasi perpustakaan perangkat lunak *Numeric* dan *Numarray*. *NumPy* dikembangkan secara terbuka di *GitHub*, melalui konsensus *NumPy* dan komunitas ilmiah *Python* (Numpy.org, 2022)

2.16 *Pandas*

Pengembangan *pandas* bermula di tahun 2008 oleh *AQR Capital Management*. Pada akhir tahun 2009, proyek tersebut berubah menjadi *open-source*, dan sampai sekarang dikembangkan secara aktif oleh sebuah komunitas individu yang berpikiran sama di seluruh dunia yang menyumbangkan waktu dan energi berharga mereka untuk memungkinkan *pandas* tetap *open-source* (pandas.pydata.org, 2022)

2.17 *Scikit-learn*

Scikit-learn adalah modul Python yang mengintegrasikan berbagai algoritma pembelajaran mesin canggih untuk masalah skala menengah yang diawasi dan tanpa pengawasan. Paket ini berfokus pada membawa pembelajaran mesin ke non-spesialis menggunakan bahasa tingkat tinggi tujuan umum. Penekanan diberikan pada kemudahan penggunaan, kinerja, dokumentasi, dan konsistensi API. Ini memiliki ketergantungan minimal dan didistribusikan di bawah lisensi BSD yang disederhanakan, mendorong penggunaannya dalam pengaturan akademik dan komersial (Pendregosa et al, 2011).

2.18 *Pillow*

Pillow adalah sebuah cabang dari dari modul Python Imaging Library (PIL) yang dibuat pada saat PIL dihentikan perkembangannya di tahun 2011.

Perpustakaan ini menyediakan dukungan format file yang luas, representasi internal yang efisien, dan kemampuan pemrosesan gambar yang cukup kuat. Perpustakaan gambar inti dirancang untuk akses cepat ke data yang disimpan dalam beberapa format piksel dasar. *Pillow* memberikan dasar yang kuat untuk alat pemrosesan gambar umum (Clark, 2022).

2.19 *Matplotlib*

Matplotlib adalah paket grafis 2D yang digunakan untuk Python untuk pengembangan aplikasi, skrip interaktif, dan pembuatan gambar berkualitas publikasi di seluruh antarmuka pengguna dan sistem operasi. (Hunter, 2007).

2.20 *Tkinter/Tk*

Tk adalah perangkat antarmuka pengguna yang memudahkan untuk membangun antarmuka pengguna grafis desktop. Dibandingkan dengan kebanyakan toolkit antarmuka pengguna. Tk unik karena dirancang sejak awal untuk dipasangkan dengan bahasa pemrograman dinamis tingkat tinggi (seperti Python, Tcl, Ruby, dan Perl) dan sebagai lawan dari bahasa tingkat rendah seperti C atau C++ (Roseman, 2022).