# BAB II
# TINJAUAN PUSTAKA

## 2.1. Penelitian Sebelumnya

Pada sub bab ini, dipaparkan mengenai penelitian – penelitian sebelumnya yang berkaitan dengan hasil prediksi serta parameter yang mempengaruhi dalam menggunakan metode LSTM. Pada penelitian yang dilakukan oleh wiranda et al tahun 2019 yang berjudul "Penerapan Long Short Term Memory Pada Data Time Series Untuk Memprediksi Penjualan Produk PT. Metiska Farma" (Wiranda & Sadikin, 2019). Permasalahan yang dialami dalam studi ini adalah mencoba mengimplementasikan LSTM pada dataset penjualan obat pada perusahaan tersebut untuk memprediksi jumlah penjualan beberapa hari kedepan. Dan hasilnya berdasarkan dua parameter percobaan, pada percobaan kelima mencapai kinerja yang lebih baik. Hasil penelitian menunjukkan bahwa komposisi data latih 90% dan data uji 10% cukup baik, *range interval* [-1,1] dan *epoch* sebanyak 1500. Metode tersebut mendapatkan nilai RMSE sebesar 13,762,154.00 dan hasil MAPE adalah 12%.

Penelitian lain yang dilakukan oleh Wildan et al pada tahun 2018 penelitian tersebut berjudul "Analisis dan Implementasi Long Short Term Memory Neural Network untuk Prediksi Harga Bitcoin" (Wildan et al., 2018). Permasalahan yang dialami adalah mencoba untuk menerapkan LSTM dalam memprediksi harga Bitcoin dan bagaimana tingkat akurasinya dalam melakukan prediksi. Hasil penelitian tersebut pada pengujian didapatkan hasil yang terbaik yaitu dengan komposisi data latih 70% dan data uji 30%, parameter 1 pola *time series*, jumlah 25 *neuron hidden*, dan *max epoch* adalah 100 dengan akurasi rata-rata pada data latih 95.36% dan data testing 93.5%.

Pada penelitian lain yang dilakukan oleh khumaidi et al tahun 2020, penelitian tersebut berjudul Pengujian Algoritma *Long Short Term Memory* untuk prediksi kualitas udara dan suhu kota bandung (Khumaidi et al., 2020). Penelitian yang dilakukuan ini menggunakan LSTM sebagai pemodelan untuk memprediksi data *time series*, yaitu kualitas udara di kota Bandung, dari parameter PM10, ISPU, Suhu, dan Kelembaban. Hasil dari penelitian tersebut model LSTM dan

penerapannya pada time series, dengan 4 *hidden layes*, penentuan jumlah *batch size* yaitu 32, penentuan *optimizer* adam, *epoch* 1000 menunjukkan bahwa model menghasilkan keakuratan prediksi yang cukup baik untuk 3 parameter (Suhu, Kelembaban, ISPU). Hal ini ditunjukan dengan nilai RMSE yang kecil.

Penelitian selanjutnya dilakukan oleh riyantoko et al tahun 2020, penelitian tersebut berjudul Analisis Prediksi Harga Saham Sektor Perbankan Menggunakan Algoritma LSTM (Riyantoko et al., 2020). Penelitian tersebut menggunakan LSTM sebagai pemodelan untuk melakukan prediksi harga saham dan mengukur tingkat akurasi dengan beberapa parameter berbeda. Parameter pengujian yang digunakan adalah model optimasi (Adam, SGD, RMSprop) dan *epoch* (25, 50, 75, 100). Hasil yang diperoleh dari penelitian tersebut adalah algoritma LSTM dengan menggunakan model optimasi Adam memiliki tingkat prediksi yang baik dan akurat, hal ini ditunjukkan dengan nilai RMSE yang turun dari setiap jumlah epoch yang berbeda.

## 2.2. Saham

Pada saat ini saham masih menjadi salah satu hal yang paling banyak diminati oleh investor dibursa efek (*stock exchange*). Saham dapat didefinisikan sebagai tanda bukti memiliki perusahaan dimana pemiliknya juga sebagai pemegang saham (Cahya & Kusuma, 2019). Bentuk saham itu sendiri hanya berupa secarik kertas yang memuat kepemilikan saham perseroan. Persentase kepemilikan saham ditentukan oleh seberapa besar dana yang ditanamkan dalam perusahaan (Zurriah, 2021). Dalam 1 hari terdapat beberapa informasi harga dalam suatu saham yaitu *open price, close price, low price, high price, adj price* dan *volume*. Harga *open* artinya harga pembukaan, harga *low* ialah harga terendah, harga *high* adalah harga tertinggi, sedangkan harga *close* merupakan harga penutupan (Pratama et al., 2018).

Saham memiliki karakteristik yaitu imbal hasil yang cukup tinggi dengan resiko yang tinggi pula. Saham memiliki peluang bagi seorang investor untuk mendapatkan nilai return atau hasil yang besar dalam waktu yang cukup singkat. Akan tetapi harga saham itu *fluktuatif*, kadang naik kadang juga turun. Tapi harga saham bergerak *fluktuatif* atau naik turun, sehingga harga saham tidak pasti dan sulit untuk ditebak. Hal ini juga dapat membuat investor mengalami kerugian yang

cukup besar dalam waktu yang singkat. Pergerakan harga saham sendiri dipengaruhi oleh beberapa faktor contohnya adalah kekuatan penawaran dan permintaan. Apabila permintaan pasar lebih besar daripada penawaran maka harga saham akan bergerak naik, begitu juga sebaliknya jika penawaran pasar lebih besar dari permintaan maka harga saham akan bergerak turun. Harga saham juga mengalami perubahan setiap saat dan akan membuat keseimbangan harga yang baru (Seviani & Budiwinarto, 2021). Oleh karena itu metode *forecasting* atau peramalan mungkin dapat membantu memberikan saran atau masukkan kepada pemain saham untuk beli maupun jual.

## 2.3. *Forecasting*

*Forecasting* atau peramalan merupakan sebuah teknik perpaduan antara ilmu dan seni. Teknik ini digunakan untuk melakukan perkiraan apa yang akan terjadi dimasa depan. Teknik ini memerlukan historis data kebelakang untuk diolah dan memproyeksikannya menjadi data yang akan terjadi di masa depan menggunakan model matematika (A. Nasution, 2019). Dapat disimpulkan bahwa dengan menggunakan metode atau teknik ini kita bisa mengetahui kira - kira apa yang akan terjadi di masa yang akan datang.

Pada metode *forecasting* terdapat 2 tipe pendekatan yaitu pendekatan *kualitatif* dan pendekatan *kuantitatif*.

1. Metode pendekatan *kualitatif* merupakan metode yang subjektif. Metode ini didasari atas data kualitatif pada masa lampau. Hasil yang didapatkan dalam pendekatan ini sangat tergantung pada pemikiran atau pola pikir orang yang menyusun metode tersebut. Pemikiran ini biasanya didasari oleh hasil penelitian sebelumnya (Krisma et al., 2019).
2. Metode peramalan *kuantitatif* ini dibagi menjadi 2 jenis, yaitu kausal dan *time series*. Untuk *kualitatif*, tipe kasual berhubungan dengan variabel dalam membuat prediksi seperti Analisis Regresi. Sedangkan untuk tipe time series ini menggunakan metode *kuantitatif*, yang mana metode dengan tipe ini dipergunakan untuk menganalisis data history masa lampau yang telah dikumpulkan secara urut berdasarkan tanggal atau waktu menggunakan teknik yang tepat.

Penggunaan dan pemilihan metode yang berbeda menyebabkan hasil akhir yang berbeda. Hasil yang didapat juga bisa digunakan sebagai acuan untuk prediksi kejadian dimasa yang akan datang (Rahmad et al., 2019).

## 2.4. *Time Series*

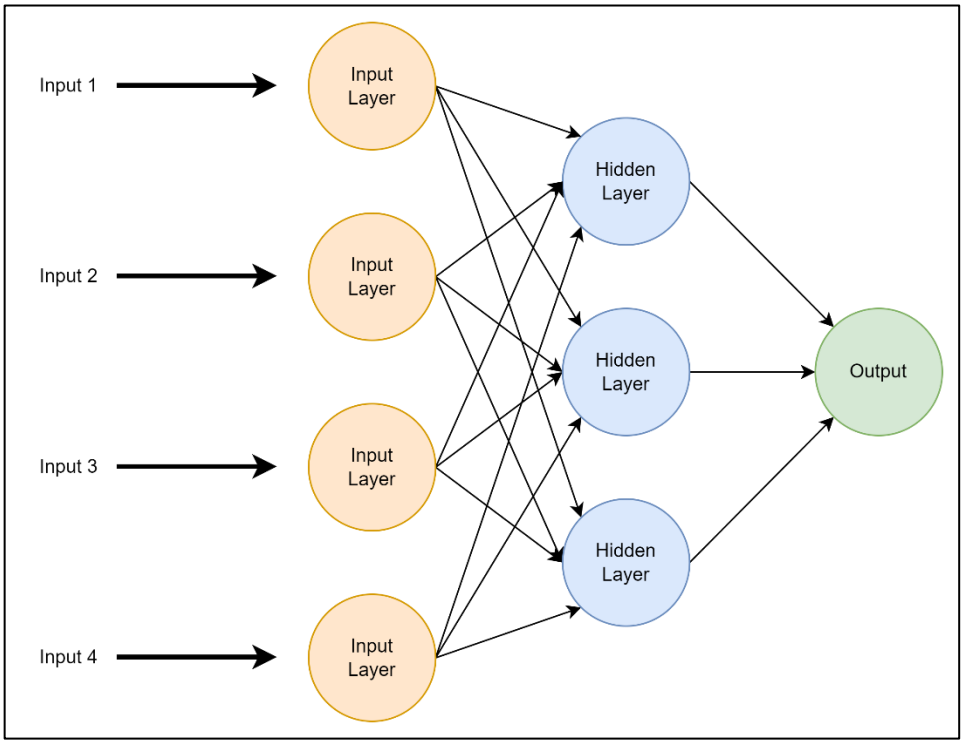
*Time series* merupakan salah satu tipe dalam metode peramalan *kualitatif*. *Time series* merupakan serangkaian pengamatan terhadap suatu peristiwa, atau kejadian yang terjadi dari waktu ke waktu dengan *interval* yang sama. Waktu yang digunakan dapat berupa hari bulan, dan tahun (Pandji et al., 2019). Hasil dari pengamatan yang dikumpulkan secara berkala pada sebuah variabel waktu tertentu menghasilkan relasi antara waktu dan kebutuhan yang dapat digunakan untuk memprediksi kebutuhan di masa yang akan datang yaitu dengan mempelajari dan melihat bagaimana variabel dapat berubah dari waktu kewaktu (Pangestu et al., 2018). Oleh karena itu untuk memilih suatu metode *time series* yang tepat harus mempertimbangkan pola data, agar metode yang digunakan dengan pola tersebut dapat diuji.

## 2.5. *Neural Network*

Jaringan Syaraf Tiruan (JST) atau yang sering dikenal dengan *Neural Network* (NN) merupakan model algoritma yang terinspirasi oleh bagaimana cara saraf *neuron* dalam otak manusia bekerja (Budiyanto & Fatimah, 2019). *Neural Network* dirancang sehingga menyerupai sistem kerja otak manusia dalam menyelesaikan suatu permasalahan dengan melakukan proses belajar.

*Neural Network* bermanfaat untuk pengenalan pola, *signal processing*, pengklasifikasian dan peramalan (Achmalia et al., 2019). Salah satu penerapan *neural network* adalah untuk melakukan prediksi atau peramalan terhadap suatu peristiwa tertentu serta dianggap mampu menyelesaikan masalah yang komplek seperti penalaran otak manusia (Pujianto et al., 2018). *Neural Network* dirancang untuk mengaolah data 2 dimensi dan NN juga merupakan metode dalam *machine learning* yang dikembangkan dari Multi Layer Perceptron (MLP). *Neural Network* termasuk dalam tipe *Deep Neural Network* karena dalamnya tingkat jaringan dan

banyak diimplementasikan dalam data citra (Ridwan et al., 2020). Gambar 2.1 Dibawah merupakan desain arsitektur sederhana Neural Network.




**Gambar 2. 1** *Architecture Neural Network*

Secara umum arsitektur dari *neural network* dipisah atau dibagi menjadi 3 bagian lapisan seperti yang ditunjukkan pada gambar 2.1 diatas. Bagian – bagian ini dikenal sebagai:

### 2.5.1. *Input Layer*

Lapisan ini bertanggung jawab untuk menerima informasi (data), sinyal, fitur, atau pengukuran dari lingkungan *eksternal*. *Input* ini (sampel atau pola) biasanya dinormalisasi dalam nilai batas yang dihasilkan oleh fungsi aktivasi. Normalisasi ini menghasilkan presisi numerik yang lebih baik untuk operasi matematika yang dilakukan oleh jaringan.

### 2.5.2. *Hidden Layer*

Lapisan ini terdiri dari *neuron* yang bertanggung jawab untuk mengekstraksi pola yang terkait dengan proses atau sistem yang dianalisis. Lapisan ini melakukan sebagian besar pemrosesan internal dari jaringan

### 2.5.3. *Output Layer*

Lapisan ini juga terdiri dari neuron, dan dengan demikian bertanggung jawab untuk memproduksi dan menampilkan keluaran akhir jaringan, yang dihasilkan dari pemrosesan yang dilakukan oleh *neuron* pada lapisan sebelumnya.

Proses *Neural Network* dapat diringkas dalam persamaan sederhana seperti yang disajikan dalam persamaan berikut (Lawal & Idris, 2020) :

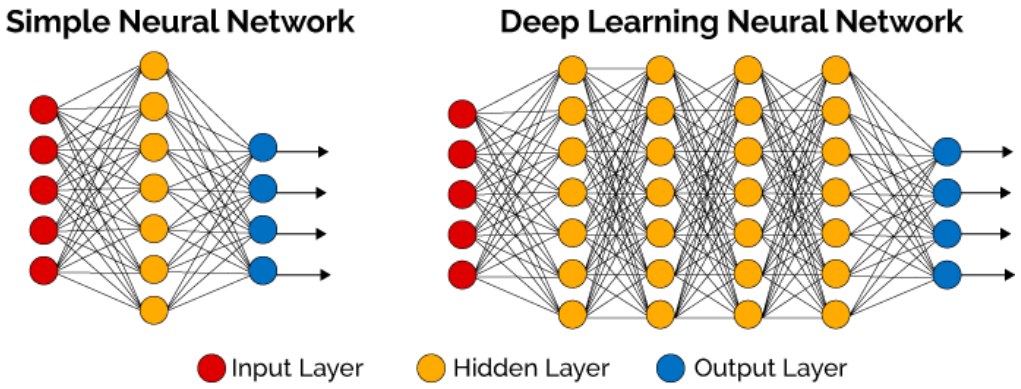
$$Y_j = f\left(\theta_j + \sum_{i=1}^{n} w_{ji} X_i\right) \qquad (2.1)$$

Keterangan:

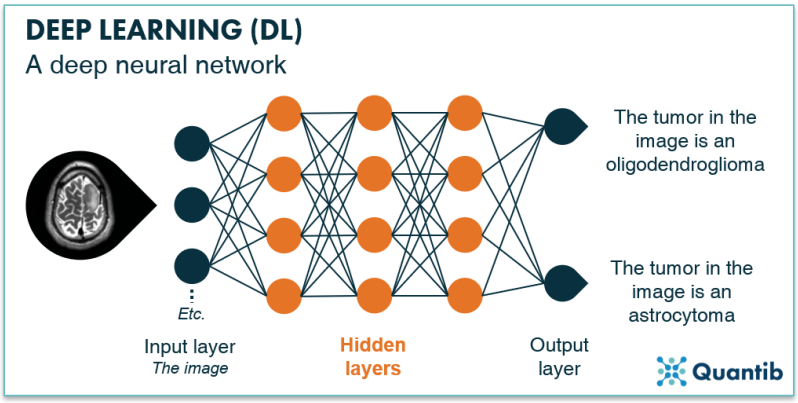
$\theta_j$ : Bias pada lapisan tersembunyi
n : Jumlah *neuron* di lapisan tersembunyi
$w_{ji}$ : Bobot koneksi antara variabel input dan lapisan tersembunyi
$X_i$ : Variabel masukan
$Y_j$ : Variabel keluaran
$f$ : Fungsi aktivasi

## 2.6. *Deep Learning*

*Deep learning* adalah format *neural network* yang mengambil *metadata* sebagai input dan kemudian memproses data melalui sejumlah lapisan untuk menghitung *output*. Sementara jaringan saraf tradisional hanya dapat menangani satu lapisan tersembunyi (Gambar 2.2, kiri), pembelajaran mendalam memproses data input melalui sejumlah besar lapisan tersembunyi dalam strukturnya (Gambar 2.2, kanan). Setiap lapisan terbuat dari node, yang merupakan tempat untuk komputasi berlangsung (Xing & Du, 2019). Untuk desain arsitektur dari *Deep Learning* sendiri dapat dilihat pada gambar 2.3.




**Gambar 2.2** Perbedaan *Neural Network* dengan *Deep Learning* (Tran, 2019)



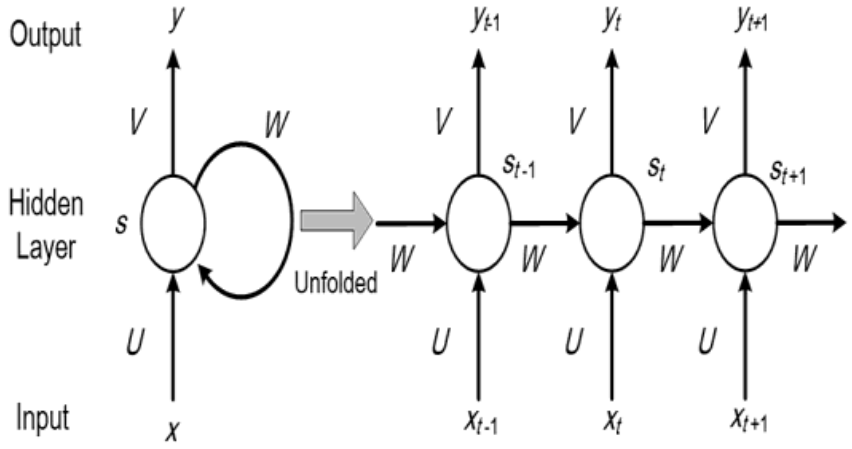

**Gambar 2.3** *Deep Learning architecture*
*(Sumber : https://www.quantib.com/blog/how-does-deep-learning-work-in-radiology)*

*Deep learning*, telah membuat terobosan besar dalam mengklasifikasi gambar menggunakan *Computer Vision*. *Deep learning* sudah banyak digunakan dan sudah mencapai hasil yang selalu baik. *Framework deep learning* yang paling banyak digunakan adalah Keras (Rochmawati et al., 2021). Keras merupakan sebuah *library open-source* yang dibuat untuk menyederhanakan model dari kerangka *Deep Learning*. Keras dapat berjalan pada TensorFlow (Santoso & Ariyanto, 2018). Keras pertama kali dikembangkan bertujuan untuk membantu para peneliti menyelesaikan eksperimen dengan lebih cepat. Pada saat ini keras sudah dikenal dan merupakan *framework* terpopuler dalam berbagai kompetisi *Deep Learning* dunia. Keras mendukung 2 bahasa pemrograman yang digunakan untuk mengolah *data science* yang populer yaitu : R dan Python (Sen et al., 2020). Kelebihan yang diberikan adalah dengan munculnya fitur – fitur seperti membuat code berjalan lancar pada CPU atau GPU, lalu *User – friendly* API yang

membuatnya mudah untuk membuat prototipe sebuah *deep learning*, dan mendukung model RNN.

## 2.7. *Recurrent Neural Network*

*Recurrent Neural Network* (RNN) adalah salah satu jenis arsitektur jaringan saraf tiruan (JST) yang dalam prosesnya dipanggil berulang kali untuk memproses input data sekuensial (Firmansyah et al., 2020). RNN termasuk dalam salah satu kategori *Deep Learning* karena data yang diproses harus melalui banyak lapisan. RNN telah mengalami kemjuan yang cukup pesat dan telah merevolusi bidang – bidang seperti NLP (Liu et al., 2016). Dalam setiap proses, *output* yang dihasilkan tidak hanya merupakan fungsi dari satu sampel itu saja, tetapi *output* yang dihasilkan juga berdasarkan state internal yang merupakan hasil dari pemrosesan sampel – sampel sebelumnya (Yogatama et al., 2017). Pemodelan RNN dapat menyelesaikan berbagai tugas kategorisasi kalimat dan dapat melakukan klasifikasi (Xia et al., 2018). Pada intinya RNN adalah jaringan syaraf tiruan yang menggunakan rekurensi dengan memanfaatkan data masa lalu. RNN memiliki arsitektur yang dapat digunakan untuk data berbentuk sekuensial seperti yang dapat dilihat pada Gambar 2




**Gambar 2.4** Arsitektur RNN (Firmansyah et al., 2020)

Pada gambar 2.4 di atas, pada bagian arsitektur sebelah kiri menunjukkan bahwa RNN berada pada posisi yang tidak terbuka ke jaringan penuh. Sedangkan untuk arsitektur yang berada disebelah kanan menunjukkan bagian RNN yang telah

dibuka menjadi *full network*. Contohnya jika data yang dimasukkan berupa 1 kalimat dengan 3 kata, maka akan terdapat 3 *layer Neural Network*, satu *layer* untuk setiap kata. Berikut ini ialah keterangan simbol formula dari gambar di atas:

1. $x_t$ adalah *input* pada time step. Misalnya, $x_1$ dapat menjadi *one-hot vector* yang sesuai dengan kata kedua dari sebuah kalimat yang sedang diproses.
2. $s_t$ adalah *hidden state* pada setiap time step t. *Hidden state* bisa disebut sebagai "*memory*" dari sebuah *network* yang berfungsi menyimpan hasil perhitungan dan rekaman yang telah dilakukan. $s_t$ dihitung berdasarkan *hidden state* sebelumnya dan berdasarkan *input* pada *current state* (keadaan saat ini)

   $$s_t = f(Ux_t + Ws_{t-1}) \quad (2.2)$$

   Fungsi f biasanya adalah *non-linear* seperti tanh atau ReLU. $s_{t-1}$ digunakan untuk menghitung hidden state yang pertama, biasanya pada inisialisasi selalu diawali dengan 0 (nol).
3. $y_t$ adalah *output* pada step t. Misalnya, jika kita ingin memprediksi "kata selanjutnya" pada sebuah kalimat, maka yt merupakan vektor probabilitas di seluruh kosakata dalam database yang kita miliki.

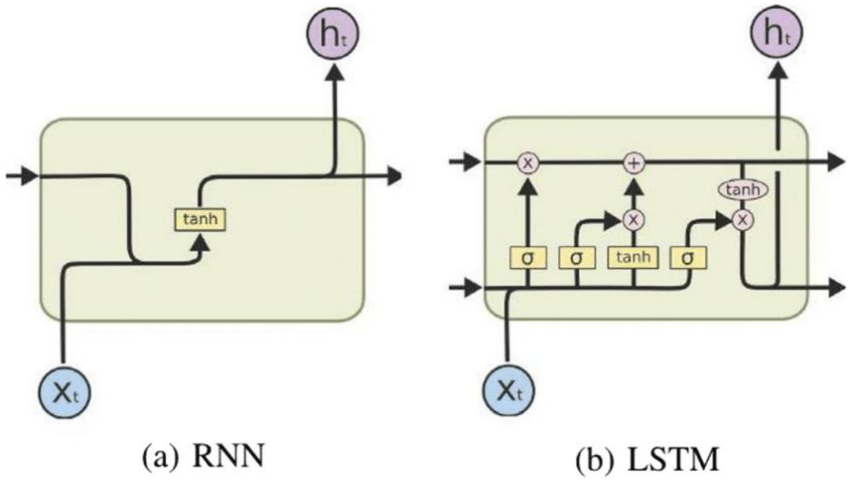
   $$y_t = softmax(Vs_t). \quad (2.3)$$

Satu permasalahan yang terdapat dalam arsitektur RNN adalah masalah hilangnya gradien (*vanishing problem*) pada proses *backward pass*. Cara yang cukup populer untuk mengatasi permasalahan ini ialah menggunakan unit RNN yaitu LSTM. LSTM diusulkan sebagai solusi untuk mengatasi terjadinya *vanishing gradient* pada RNN saat memproses data *sequential* yang panjang. Permasalahan *vanishing gradient* ini mengakibatkan RNN gagal dalam menangkap *long term dependencies*, sehingga mengurangi akurasi dari suatu prediksi pada RNN.

## 2.8. *Long Short Term Memory*

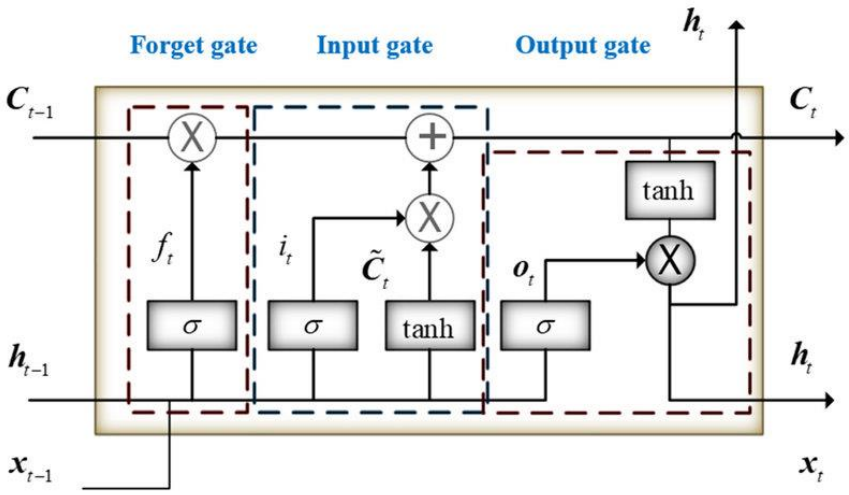
*Long Short Term Memory* (LSTM) merupakan salah satu jenis dari RNN dimana dilakukan modifikasi pada RNN dengan menambahkan *memory cell* yang dapat menyimpan informasi untuk jangka waktu yang lama (Aprian et al., 2020). Metode ini diajukan oleh Sepp Hochreiter dan Jurgen Schimidhuber pada tahun 1927. RNN tidak dapat untuk belajar menghubungkan informasi karena memori

lama yang tersimpan akan semakin banyak dan tidak berguna dengan seiring berjalan waktu karena tertimpa atau tergantikan dengan memori yang baru, permasalahan ini ditemukan oleh Bengio, et al. (1994) (Arfan & ETP, 2019). Gambar 2.5 merupakan perbedaan arsitektur antara RNN dengan LSTM.



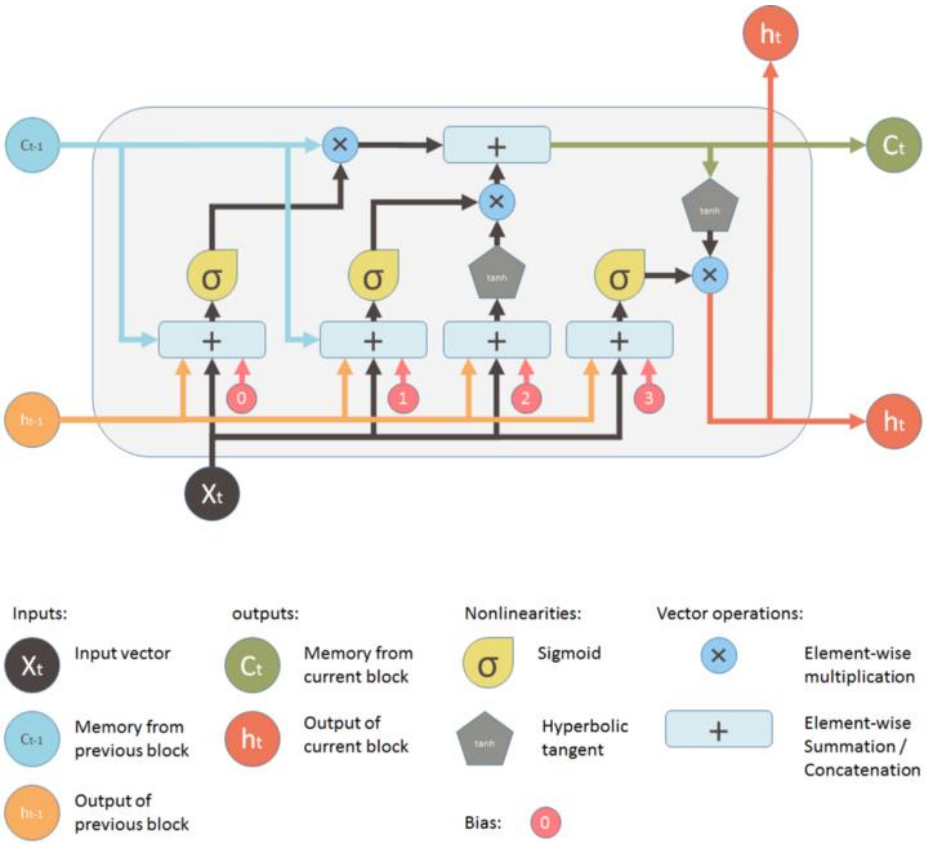

**Gambar 2. 5** Perbedaan arsitektur RNN dan LSTM
*(Sumber : https://www.researchgate.net/figure/The-general-schema-of-a-RNN-unit-versus-a-LSTM-one-adapted-from-Olah-2015_fig1_338905482)*

LSTM dapat mempelajari data mana yang dibutuhkan untuk diproses dan data mana yang tidak karena memiliki sistem gates di dalam nya. LSTM sendiri cocok digunakan pada teknologi pemrosesan video maupun teks, dan data time series (Wildan et al., 2018). Sebuah penelitian *A New Method for Semantic Consistency Verification of Aviation Radiotelephony Communication Based on LSTM-RNN* mengatakan bahwa LSTM berhasil diterapkan pada berbagai tugas sekuensial dan bahasa pemodelan (Wiranda & Sadikin, 2019). Arsitektur LSTM seperti gambar 2.6 dibawah memiliki *gate* seperti filter yang berfungsi untuk menghapus maupun menambahkan informasi.




**Gambar 2. 6** Arsitektur *Long Short Term Memory* (LSTM) (Jiang et al., 2019)

Pada gambar 2.6 diatas tergambar bentuk dari LSTM, pada gambar tersebut terdapat *input gate, forget gate* dan *output gate*. Dan pada gambar 2.7 merupakan bagian dari masing – masing filter dari LSTM. Untuk memperjelas dalam pemahaman mengenai LSTM dapat dilihat pada gambar dibawah ini.

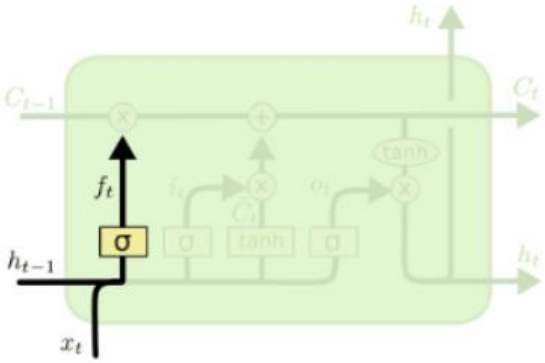



**Gambar 2.7** Bagian - bagian LSTM
*(Sumber : https://becominghuman.ai/q-a-system-deep-learning-2-2-c0ad60800e3?gi=b508e10a1283)*

Alur tahapan informasi yang melewati LSTM adalah sebagai berikut :

### 2.8.1. *Forget Gate*

Gerbang ini memutuskan apakah suatu informasi harus dibuang atau tidak dari *cell state*. Hasil output dari gerbang ini adalah angka antara 0 dan 1. Angka 1 menggambarkan bahwa informasi harus disimpan, sedangkan angka 0 sebaliknya, yaitu informasi sudah tidak dibutuhkan sehingga boleh dihapus.




**Gambar 2.8** *Forget Gate LSTM*
*(sumber : https://indoml.com/2018/04/13/pengenalan-long-short-term-memory-lstm-dan-gated-recurrent-unit-gru-rnn-bagian-2)*

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (2.4)$$

Keterangan :

$f_t$ : Forget gate

$\sigma$ : Fungsi sigmoid

$W_f$ : Bobot untuk nilai input pada waktu ke t

$h_{t-1}$ : Output dari waktu ke t-1

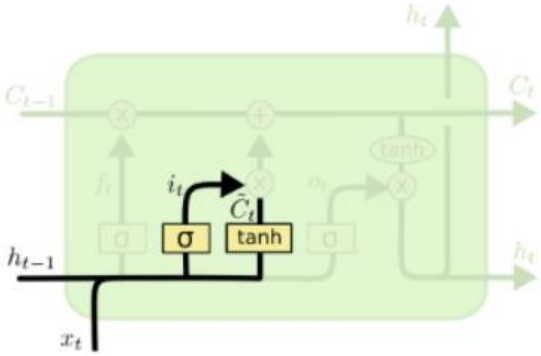
$x_t$ : Input pada waktu ke t

$b_f$ : Bias pada forget gate

### 2.8.2 *Input Gate*

Secara kondisional, gerbang ini digunakan untuk menentukan sebuah masukan baru yang akan ditambahkan ke dalam memori *cell state* saat itu atau tidak (Zaman et al., 2019). Fungsi aktivasi sigmoid pertama, memutuskan nilai

mana yang akan diupdate. Kemudian fungsi tanh, membuat vektor dari kandidat nilai yang baru, Ct, yang akan ditambahkan ke state. Kedua nilai dari fungsi aktivasi ini kemudian akan dikombinasikan menjadi nilai tambahan.




**Gambar 2.9** *Input Gate LSTM*
*(sumber : https://indoml.com/2018/04/13/pengenalan-long-short-term-memory-lstm-dan-gated-recurrent-unit-gru-rnn-bagian-2)*

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.5)$$

$$\widetilde{C_t} = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2.6)$$

Keterangan :

$i_t$ : Nilai input gate

$\sigma$ : Fungsi Sigmoid

$W_i$ : Bobot untuk nilai input pada waktu ke t

$h_{t-1}$ : Nilai output dari waktu ke t-1

$x_t$ : Nilai input pada waktu ke t

$b_i$ : Bias pada input gate

$C_t$ : Kandidat cell state

$tanh$ : Fungsi Hyperbolic tangent

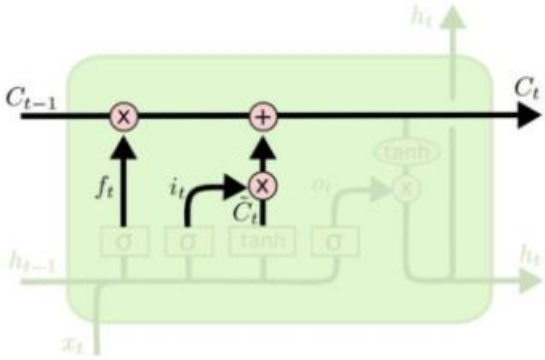
$W_c$ : Bobot untuk nilai input pada cell ke c

$b_c$ : Bias pada cell ke c

### 2.8.3 *Cell State*

Jalur ini berfungsi sebagai memori atau ingatan pada layer LSTM. Pada akhir tahap ini, nilai pada *cell state* diupdate dari Ct-1 menjadi Ct. Nilai dari *state* yang lama, dikalikan dengan nilai dari *forger gete*($f_t$) yang mana bisa merupakan nilai

anatar 0 sampai 1. Kemudian ditambahkan nilai dari hasil *input gate* yaitu $i_t$ * $C_t$. Pada akhirnya didapatkan nilai baru dari *cell state* ($C_t$)




**Gambar 2.10** *Cell State*
*(sumber : https://indoml.com/2018/04/13/pengenalan-long-short-term-memory-lstm-dan-gated-recurrent-unit-gru-rnn-bagian-2)*

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (2.7)$$

Keterangan :

$C_t$ : Nilai memori *cell state*

$f_t$ : Nilai *forget gate*

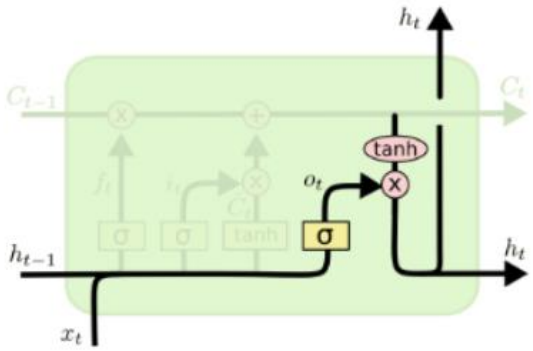
$C_{t-1}$ : Nilai memori *cell state* pada cell sebelumnya

$i_t$ : Nilai dari *input gate*

$\tilde{C}_t$ : Nilai kandidat memory *cell state*

### 2.8.4 *Output Gate*

*Gate* ini memutuskan apa yang akan dihasilkan berdasarkan nilai dari *input* dan *cell state*. Pada tahap akhir ini, *output* berdasarkan pada *cell state* saat ini, tapi akan melalui tahap filter terlebih dahulu. Pertama *layer sigmoid* yang memutuskan nilai mana yang akan dipertahankan menuju *output*. Kemudian *cell state* akan melewati fungsi tanh (yang mengubah nilai menjadi rentang antara -1 dan 1) dan mengalikan dengan output dari gerbang sigmoid (Moghar & Hamiche, 2020).




**Gambar 2.11** *Output Gate LSTM*
*(sumber : https://indoml.com/2018/04/13/pengenalan-long-short-term-memory-lstm-dan-gated-recurrent-unit-gru-rnn-bagian-2)*

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \quad (2.8)$$

$$h_t = o_t * tanh\, C_t \quad (2.9)$$

Keterangan:

$o_t$ : *Output gate*

$\sigma$ : Fungsi sigmoid

$W_o$ : Bobot untuk nilai *input* pada waktu ke t

$h_{t-1}$ : Nilai *output* dari waktu ke t-1

$x_t$ : Nilai *input* pada waktu ke t

$b_o$ : Bias pada *output gate*

$h_t$ : Hasil *output* final

$o_t$ : Nilai *output gate*

$tanh$ : Fungsi *hyperbolic tangent*

$C_t$ : Nilai memori *cell state* yang baru

## 2.9. Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang digunakan pada jaringan saraf untuk mengaktifkan atau tidak mengaktifkan *neuron*. Karakteristik yang harus dimiliki oleh fungsi aktivasi jaringan perambatan balik antara lain harus kontinyu, terdiferensialkan, dan tidak menurun secara *monotonis* (*monotonically non-decreasing*). Lebih lanjut, untuk efisiensi komputasi, turunan fungsi tersebut mudah didapatkan dan nilai turunannya dapat dinyatakan dengan fungsi aktivasi itu sendiri. Berikut ini merupakan beberapa fungsi aktivasi yang sering digunakan:

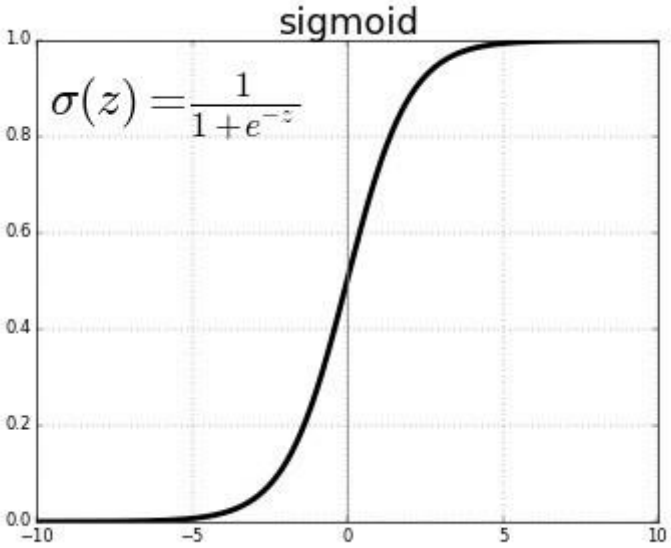
### 2.9.1. Fungsi *Sigmoid*

Ini adalah fungsi aktivasi yang paling banyak digunakan karena merupakan fungsi *non-linier*. Fungsi sigmoid mengubah nilai dalam rentang 0 hingga 1. Fungsi sigmoid dirumuskan sebagai berikut:

$$f(x) = \frac{1}{e^{-x}} \tag{2.10}$$

Fungsi sigmoid terdiferensialkan secara terus menerus dan fungsi berbentuk seperti huruf S. Gambar 2.7 merupakan grafik fungsi Sigmoid. Turunan dari fungsi tersebut adalah:

$$f'(x) = 1 - sigmoid(x) \tag{2.11}$$

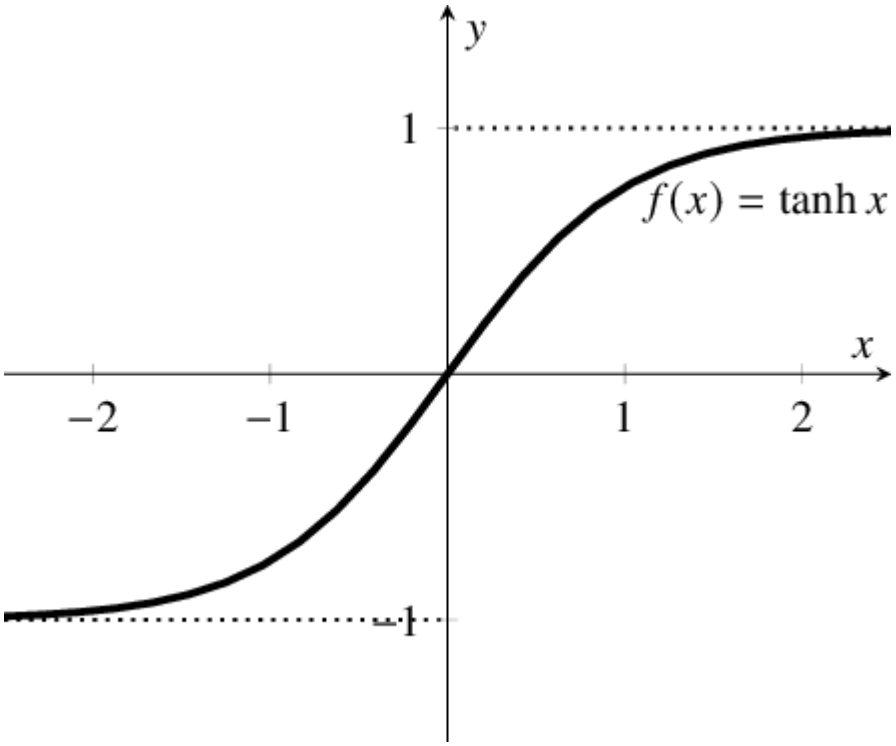



**Gambar 2.12** *Fungsi Sigmoid*
*(Sumber : https://ichi.pro/id/fungsi-aktivasi-219226710201562)*

### 2.9.2. Fungsi Tanh

Ini adalah fungsi Tangen Hiperbolik. Fungsi Tanh mirip dengan fungsi sigmoid tetapi simetris dengan sekitar titik asal. Ini menghasilkan tanda keluaran yang berbeda dari lapisan sebelumnya yang akan diumpankan sebagai input ke lapisan berikutnya. Gambar 2.13 merupakan grafik fungsi Tanh. Fungsi tanh dirumuskan sebagai berikut:

$$f(x) = 2sigmoid(2x) - 1 \qquad (2.12)$$

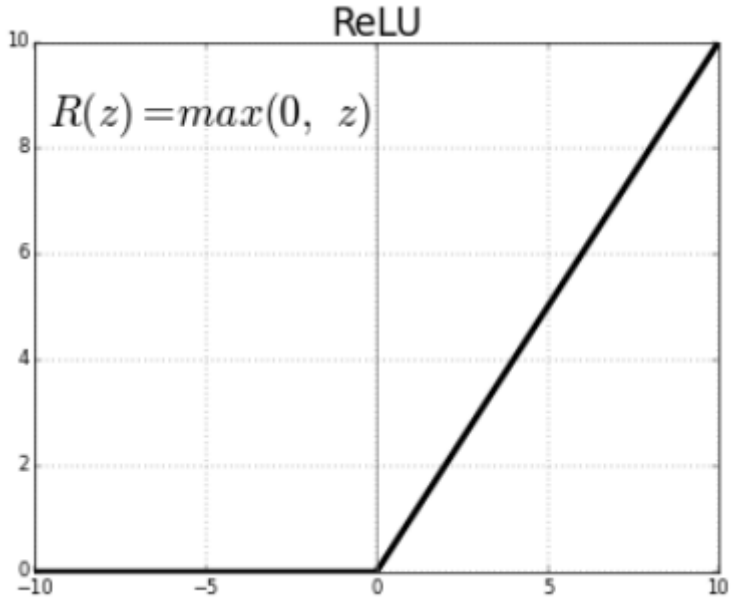



**Gambar 2.13** *Fungsi Tanh* (Jacobs et al., 2018)

### 2.9.3. Fungsi ReLU

ReLU adalah singkatan dari *rectified liner unit* dan merupakan fungsi aktivasi *non-linier* yang banyak digunakan dalam jaringan saraf. Keunggulan menggunakan fungsi ReLU adalah bahwa semua neuron tidak diaktifkan secara bersamaan. Ini bierarti bahwa neuron akan dinonaktifkan hanya ketika *output* dari transformasi linier adalah nol. Gambar 2.9 merupakan grafik fungsi ReLU. Fungsi ReLU dirumuskan sebagai berikut:

$$f(x) = \max(0, x) \qquad (2.13)$$




**Gambar 2.14** *Fungsi ReLU*
*(Sumber : https://ichi.pro/id/fungsi-aktivasi-219226710201562)*

## 2.10. *Normalisasi Data*

Normalisasi data adalah langkah yang dibutuhkan dalam *pre-processing* data, langkah ini membuat beberapa variabel memiliki rentang nilai yang sama sehingga dapat membuat analisis statistik menjadi lebih mudah (D. A. Nasution et al., 2019). Berikut merupakan beberapa metode yang digunakan untuk proses normalisasi:

### 2.10.1. *Min – Max Normalization*

*Min – Max Normalization* merupakan metode normalisasi dengan melakukan transformasi linier terhadap data asli yang kompleks dengan tidak menghilangkan isi, sehingga lebih mudah diolah. Dilakukan denan cara standarisasi data denan menempatkan data dalam range 0 hingga 1, dengan nilai terkecil sebagai 0, dan nilai terbesar sebagai 1 (Hanifa et al., 2017). *Min-max normalization* dapat dirumuskan sebagai berikut:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad (2.14)$$

Keterangan :

$x$ : Data asli

$x'$ : Hasil normalisasi

$x_{max}$ : Nilai tertinggi dalam data

$x_{min}$ : Nilai terendah dalam data

### 2.10.2. *Z – Score*

Metode *Z – Score* merupakan metode normalisasi yang berdasarkan *mean* atau nilai rata – rata dan standard *deviation* (deviasi standart) dari setiap data input (Imron & Prasetiyo, 2020). *Z – Score* dapat dirumuskan sebagai berikut :

$$x_{new} = \frac{x - mean}{std} \tag{2.15}$$

Keterangan:

$x_{new}$ : Hasil normalisasi

$x$ : Data asli

$mean$ : Nilai rata – rata data

$std$ : Nilai dari standard deviasi

Standard deviasi dapat dihitung menggunakan rumus sebagai berikut :

$$SD_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{2.16}$$

Keterangan :

$SD_x$ : Hasil standard deviation

$n$ : Jumlah data

$\bar{x}$ : Nilai rata – rata data

$x_i$ : Nilai dari data ke i

### 2.10.3. *Decimal Scaling Normalization*

Metode *Decimal Scaling* merupakan metide normalisasi dengan menggerakkan nilai desimal dari data ke arah yang diinginkan (Patel, 2019). Untuk normalisasi menggunakan metode ini bisa menggunakan rumus berikut :

$$x_{new} = \frac{x}{10^i} \tag{2.17}$$

Keterangan:

$x_{new}$ : Data hasil normalisasi

$i$ : Nilai *scaling* yang diinginkan

## 2.11. Pengukuran Nilai Error

Setelah mendapatkan hasil prediksi dari proses yang telah dijalankan maka langkah selanjutnya adalah melakukan pengecekan atau pengukuran nilai error dari hasil prediksi tersebut. Seperti yang diketahui bahwa harga yang terjadi atau terbentuk dimasa depan tidak ada yang tahu pasti akan tetapi dengan menggunakan metode forecasting atau peramalan maka masa depan dapat terprediksi mendekati realita. Walaupun tidak sama persis dengan realita, akan tetapi ini hasil ini bisa dijadikan salah satu pertimbangan untuk membuat keputusan dimasa depan. Hasil peramalan sampai saat ini belum ada yang bisa dipastikan benar seluruhnya, pasti ada perbedaan antara nilai actual dengan nilai prediksi. Berikut ini beberapa jenis evaluasi yang sering digunakan:

### 2.11.1. *Root Mean Square Error (RMSE)*

RMSE (*Root Mean Square Error*) adalah metode atau cara umum yang sering dipergunakan untuk mengukur kesalahan model dari prediksi data yang bersifat *quantitatif* (Karno, 2020). RSME menghitung kesalahan berdasarkan nilai standart deviasi. Hasil akhir diberikan dalam standart deviasi dari besarnya kesalahan.

Rumus RMSE:

$$\sqrt{\frac{\sum_{t=1}^{n}(A_t - F_t)^2}{n}} \qquad (2.18)$$

Keterangan:

At = Nilai data Aktual

Ft = Nilai hasil peramalan

N= banyaknya data

∑ = *Summation* (Jumlahkan keseluruhan nilai)

### 2.11.2. *Mean Absolute Percentage Error (MAPE)*

*Mean Absolute Percentage Error* (MAPE) adalah nilai rata – rata perbedaan absolut yang ada diantara nilai dari prediksi dan nilai realisasi yang disebutkan sebagai hasil persenan dari nilai realisasi. Penggunaan Mean Absolute Percentage Error (MAPE) pada evaluasi dari hasil peramalan dapat melihat tingkat akurasi terhadap angka peramalan dan angka realisasi (Nabillah & Ranggadara, 2020). Semakin rendah nilai presentase error pada MAPE menunjukkan semakin akurat hasil peramalan tersebut.

**Tabel 2.1** *Range* nilai *MAPE*

| **Range MAPE** | **Arti Nilai** |
|---|---|
| **< 10%** | Kemampuan model peramalan sangat baik |
| **10 – 20%** | Kemampuan model peramalan baik |
| **20 – 50%** | Kemampuan model peramalan layak |
| **> 50%** | Kemampuan model peramalan buruk |

Dari tabel tersebut kita bisa mengetahui mengenai rentang nilai yang menunjukkan arti nilai persentase error pada MAPE, dimana nilai MAPE masih dianggap bisa digunakan apabila tidak melebihi 50%, jika nilai MAPE sudah di atas 50% maka model tersebut tidak bisa atau tidak layak untuk dijadikan acuan. Nilai MAPE dapat dihitung dengan menggunakan persamaan berikut:

$$\frac{\sum_{t=1}^{n}\left|\left(\frac{A_t - F_t}{A_t}\right)100\right|}{n} \qquad (2.19)$$

Keterangan :

At = Aktual permintaan ke t

Ft = hasil peramalan ke t

N = besarnya data peramalan

Dimana terdapat simbol absolut pada rumus MAPE menunjukkan bahwa nilai negatif hasil perhitungan akan tetap bernilai positif.