BAB 3 ANALISIS DAN PERANCANGAN SISTEM

Bab ini memaparkan analisis masalah yang akan diatasi. Analisis masalah dilakukan dengan menjelaskan pendekatan dan alur kerja dari perangkat lunak yang dikembangkan, implementasi dari metode yang akan digunakan, dan hasil yang akan ditampilkan sistem.

3.1 Analisis Masalah

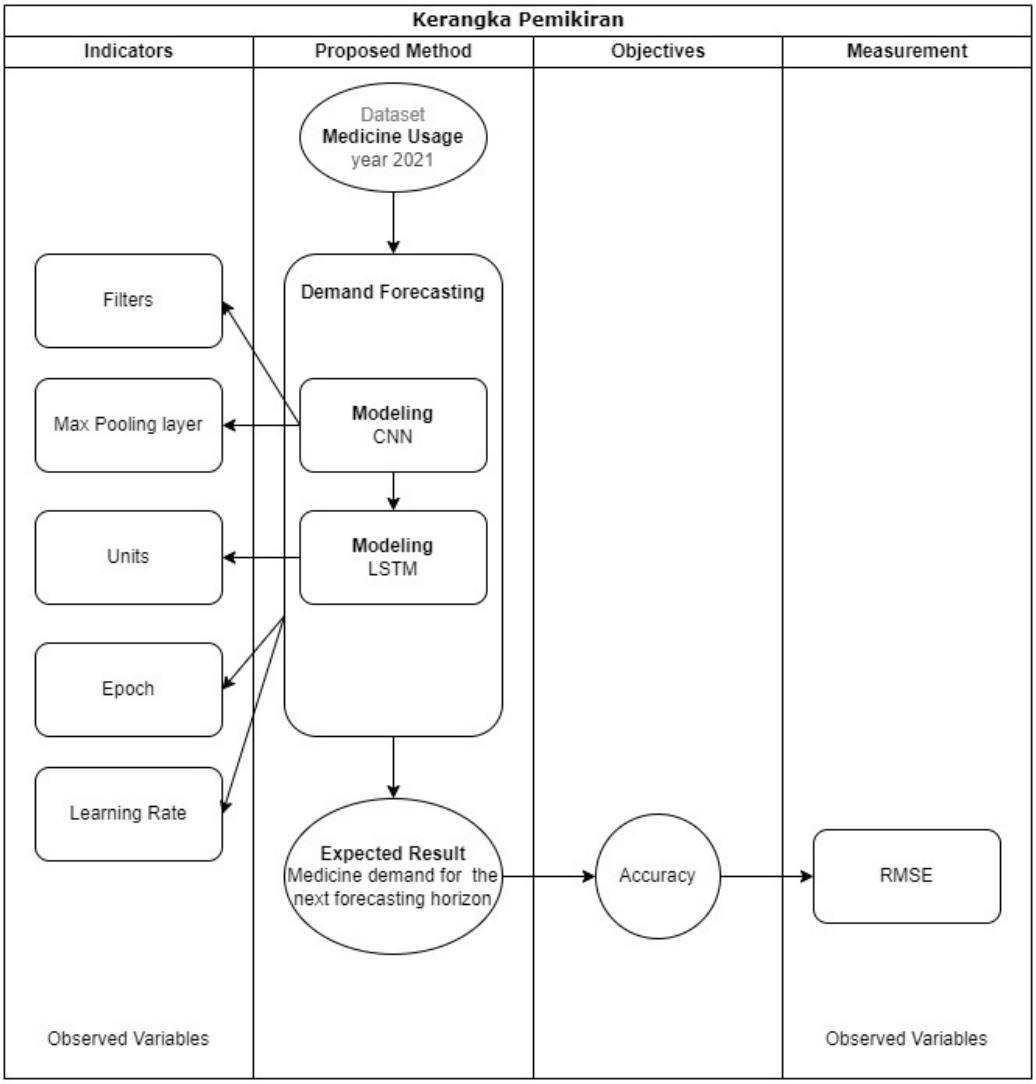
Pada bab 1 telah dijelaskan mengenai prediksi pemakaian obat pada rumah sakit merupakan hal yang penting dan sesuai dengan program JKN dari pemerintah. Pada penelitian ini, penulis menggunakan data pemakaian obat perhari di rumah sakit yang berada di Jakarta, Indonesia. Penerapan *Nonpooling* CNN-LSTM dilakukan untuk melakukan prediksi pemakaian obat di rumah sakit tersebut. Selain itu, penelitian ini juga akan membandingkan dengan model CNN-LSTM dengan lapisan *Max Pooling*.

Hasil perhitungan *Nonpooling* CNN-LSTM dan CNN-LSTM berupa prediksi titik data hari berikutnya selama *n* hari dari data masukan. Pertama data akan diproses melalui lapisan CNN terlebih dahulu untuk melakukan ekstraksi fitur-fitur penting dan mempelajari *short-term seasonality* sebelum diberikan kepada LSTM. Kemudian LSTM akan melakukan proses kembali untuk mempelajari *long-term seasonality* dari data yang nantinya akan diberikan ke *fully-connected layer* agar menghasilkan nilai keluaran sesuai *n* hari yang ingin diprediksi.

Masukan sistem prediksi merupakan data riwayat pemakaian obat di rumah sakit perhari selama 1 tahun kebelakang, sedangkan keluaran dari sistem akan menunjukan berapa pemakaian di hari berikutnya selama *n* hari ke depan. Penelitian ini menentukan *n* hari ke depan adalah 30 hari (1 bulan) sebagai prediksi *short-term* dan 90 hari (3 bulan) sebagai prediksi *long-term*.

3.2 Kerangka Pemikiran

Gambar 3.1 adalah kerangka pemikiran dari metode yang diusulkan untuk melakukan prediksi pemakaian obat rumah sakit.



Gambar 3.1 Kerangka Pemikiran

Berdasarkan Gambar 3.1, penelitian ini akan dimulai dengan membuat pemodelan CNN-LSTM dengan lapisan *Max Pooling* dan tanpa lapisan *Max Pooling* akan menggunakan data pemakaian obat selama 1 tahun pada tahun 2021. Pelatihan ini bertujuan melihat *error* pemodelan CNN-LSTM dengan dan tanpa lapisan *Max Pooling* dalam melakukan prediksi pemakaian untuk melihat kebutuhan obat pada rumah sakit. Berikut ini merupakan penjelasan setiap bagian pada Gambar 3.1.

1. *Indicators* adalah variabel yang akan memengaruhi hasil dari metode utama. Indikator dinilai pada saat pegujian CNN-LSTM dimulai. Indikator yang diobservasi selama penelitian ini antara lain sebagai berikut.

(a) *Filters* merupakan banyak *feature map* yang akan dihasilkan dalam satu kali konvolusi pada *CNN* serta menentukan banyak *kernel* untuk memproses *feature map*. Seperti yang sudah dijelaskan pada bagian 2.1.2.1 , *kernel* ini akan melakukan ekstraksi fitur. Maka, *kernel* yang semakin banyak akan menghasilkan fitur yang semakin banyak. Masing-masing operasi konvolusi menghasilkan *feature map* dua dimensi. Semakin banyak *filter* dalam satu *convolutional layer*, semakin banyak informasi atau fitur yang dapat diperoleh dari citra masukan. Namun, sama seperti jumlah konvolusi, jika jumlah *filter* terlalu banyak, model yang dibangun cenderung tidak mudah mencapai *generalization*. Karena itu, variabel ini diobservasi untuk menentukan berapa banyak *filter* yang tepat dengan kombinasi jumlah *filter* bagi arsitektur *CNN* yang dibangun untuk mendapatkan fitur.

(b) *Max Pooling Layer* merupakan lapisan pada CNN yang berfungsi untuk mengambil nilai maksimum dari hasil convolusi dan akan menghasilkan lapisan baru dengan ukuran yang lebih kecil. Pengambilan nilai maksimum ini berdasarkan ukuran *kernel* yang telah ditentukan sebelumnya. Adanya lapisan ini dapat membuat pekerjaan lebih efisien, namun pada kasus data *time-series* hal ini dapat menyebabkan terjadinya kehilangan informasi. Karena itu, variabel ini diobservasi untuk menentukan pengaruh lapisan *Max Pooling* pada arstitektur CNN-LSTM.

(c) *Units* merupakan konfigurasi jumlah *hidden neuron* yang akan dihitung pada arsitektur LSTM. Semakin banyak jumlah *units* akan mempengaruhi kepada banyaknya proses atau informasi yang dapat diproses oleh LSTM. Variabel ini akan diobservasi untuk menentukan jumlah *units* pada arsitektur CNN-LSTM.

(d) *Epoch* merupakan jumlah iterasi selama proses pelatihan berlangsung. Nilai *epoch* yang optimal ditentukan berdasarkan hasil dari proses pelatihan. Variabel ini akan diobservasi untuk menetukan jumlah *epoch* pada arsitektur CNN-LSTM.

(e) *Learning rate* merupakan seberapa besar nilai hasil pelatihan dari satu iterasi akan mempengaruhi hasil iterasi berikutnya. Variabel ini diobservasi untuk menentukan besar *learning rate* pada arsitektur CNN-LSTM

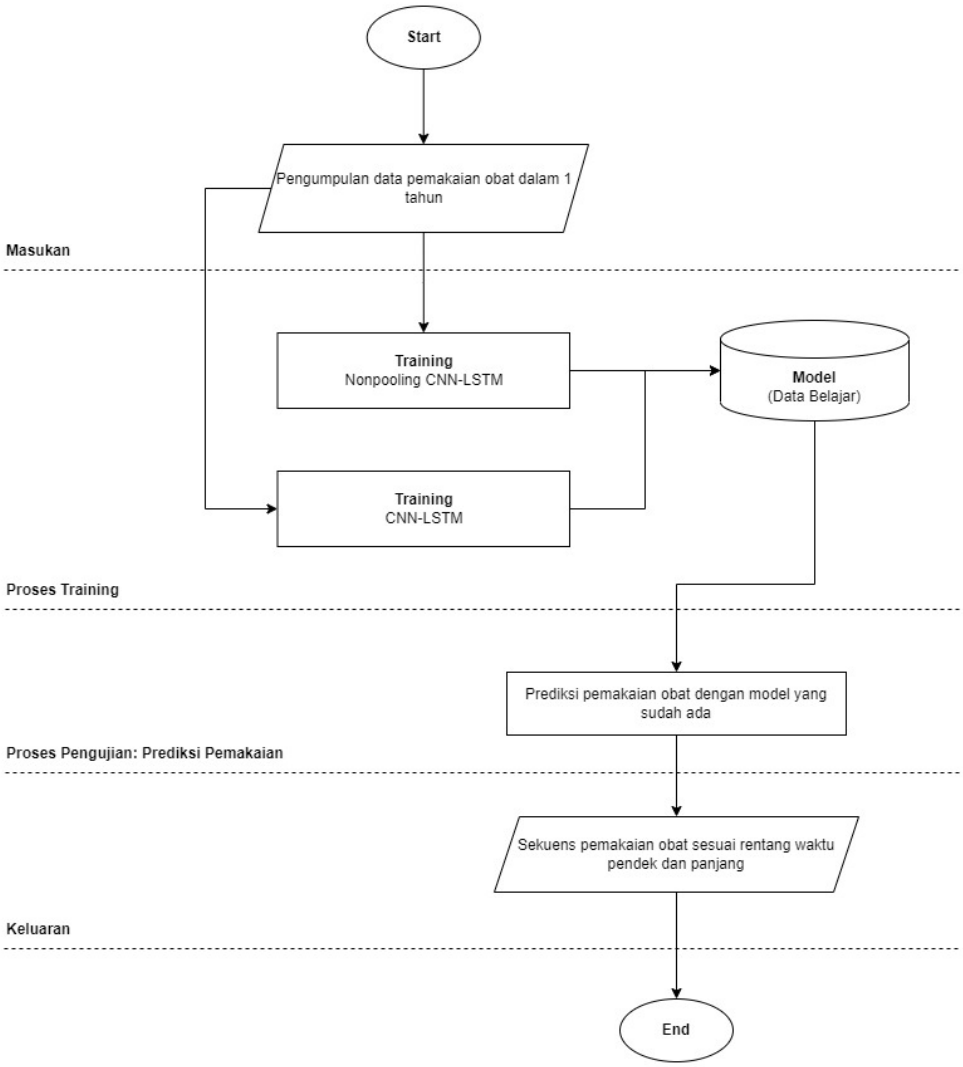
2. *Proposed Method* adalah bagian yang menjelaskan proses penelitian dari awal hingga akhir. Proses ini dimulai dengan membuat model CNN-LSTM dengan dan tanpa lapisan *Max Pooling* lalu diuji untuk memperoleh hasil prediksi

pemakaian obat dalam bentuk sekuens. Sekuens tersebut kemudian diuji untuk melihat nilai *error*.

3. *Objectives* adalah bagian yang menjelaskan target yang akan menjadi acuan pengukuran. Dalam penelitian ini, target tersebut adalah *error* dari estimasi sekuens pemakaian obat oleh model CNN-LSTM dengan dan tanpa lapisan *Max Pooling*.
4. *Measurement* adalah satuan ukur yang digunakan untuk mengukur hal-hal yang ada pada bagian *Objectives*. Dalam penelitian ini, digunakan metrik RMSE sebagai nilai evaluasi nilai *error* pemodelan CNN-LSTM dengan dan tanpa lapisan *Max Pooling*.

3.3 Analisis Urutan Proses Global

Penelitian ini akan menggunakan CNN-LSTM dengan dan tanpa lapisan *Max Pooling* untuk melakukan prediksi pemakaian obat. Setelah dilakukan pelatihan, diharapkan arsitektur yang dibangun dapat melakukan prediksi pemakaian obat dengan tepat. Hasil estimasi pemakaian obat kemudian akan diuji kembali dengan *dataset*. Gambar 3.2 menunjukkan urutan proses global pada penelitian ini dalam bentuk *flowchart*.

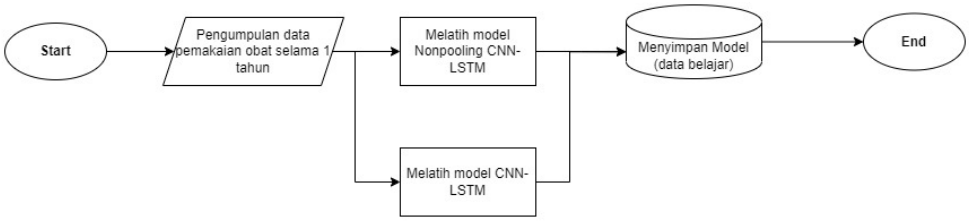


Gambar 3.2 *Flowchart* urutan proses global

Sistem prediksi pemakaian obat terbagi atas 2 proses, yaitu proses pelatihan dan proses pengujian. Proses pelatihan dilakukan untuk mendapatkan model CNN-LSTM dengan dan tanpa lapisan *Max Pooling* yang optimal untuk menghasilkan *error* yang kecil. Proses pengujian dilakukan untuk mencari sekuens pemakaian obat sesuai dengan rentang waktu prediksi pendek dan panjang.

3.3.1 Proses Pelatihan

Pada penelitian ini, proses pelatihan model CNN-LSTM dengan dan tanpa lapisan *Max Pooling* digambarkan pada Gambar 3.3.

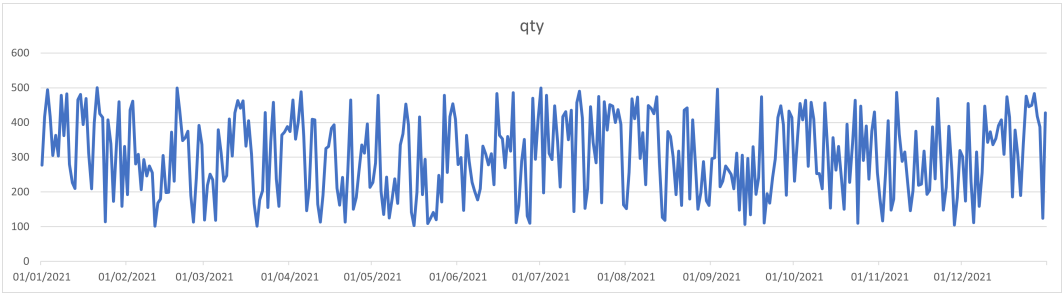


Gambar 3.3 *Flowchart* pelatihan sistem prediksi pemakaian obat

Berikut adalah uraian proses pelatihan dari *flowchart* pada Gambar 3.3 yang dilakukan dalam penelitian ini.

1. Arsitektur CNN-LSTM dengan dan tanpa lapisan *Max Pooling* yang akan digunakan pada penelitian ini akan mempengaruhi proses pengujian. Data pertama akan diproses pada lapisan CNN untuk melakukan ekstraksi fitur dan *seasonality* jarak pendek yang nantinya akan diberikan secara utuh kepada lapisan LSTM karena tidak menggunakan lapisan *Max Pooling*. Kemudian lapisan LSTM akan mempelajari data kembali untuk nantinya dikeluarkan hasil prediksi berupa nilai pemakaian obat.
2. Sebagai masukan untuk model CNN-LSTM dengan dan tanpa lapisan *Max Pooling* yang dibangun, digunakan *dataset* larik 2 dimensi dimana dimensi pertama adalah waktu dan dimensi kedua adalah jumlah pemakaian obat. Gambar 3.4 adalah visualisasi dari data pemakaian obat selama 1 tahun di tahun 2021.

   Proses pelatihan model CNN-LSTM dengan dan tanpa lapisan *Max Pooling* terdiri atas 2 proses pada masing-masing model, yaitu pada lapisan CNN dengan dan tanpa lapisan *Max Pooling* dan pada lapisan LSTM. Pada bagian berikutnya akan dijelaskan tahap pelatihan dengan CNN dan LSTM.



Gambar 3.4 Gambaran *dataset* dengan keterangan sumbu *x* dan *y*

3.3.1.1 CNN-LSTM

Berikut ini adalah algoritme dalam proses CNN untuk menghasilkan fitur yang nantinya akan diberikan kepada lapisan LSTM. Proses ini akan dilakukan secara

berulang berdasarkan jumlah iterasi yang telah ditentukan. Algoritme 3.1 merupakan penjelasan secara umum bagaimana proses CNN-LSTM dilakukan.

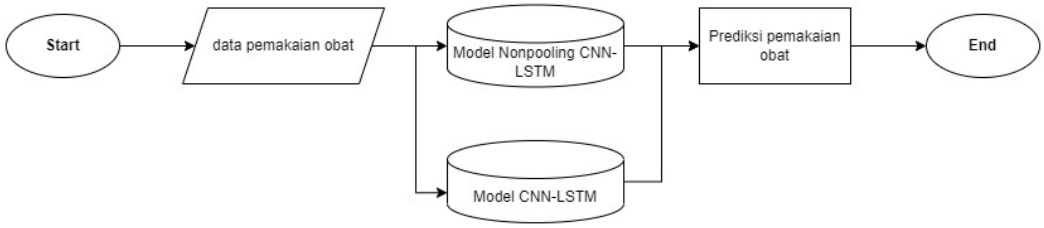
ALGORITME 3.1 CNN-LSTM

1: Masukan merupakan larik 2 dimensi dimana dimensi pertama adalah waktu dan dimensi kedua adalah jumlah, maka dari itu yang digunakan untuk pelatihan hanya merupakan larik dengan ukuran 365×1.
2: Inisialisasi nilai *kernel*, *weight*, *bias*. Nilai *kernel* diinisialisasi untuk melakukan konvolusi pada *convolutional layer* dan pada lapisan *dense* untuk mengeluarkan nilai keluaran, sedangkan nilai *weight* dan *bias* untuk melakukan perhitungan pada lapisan LSTM. Nilai *kernel* dan *weight* diinisialisasikan dengan nilai acak terdistribusi normal sesuai dengan Persamaan 2.4. Sedangkan nilai *bias* pada tahap pertama akan menggunakan nilai *default* yaitu 0.
3: Lakukan proses konvolusi pada setiap *convolutional layer*.
4: Konvolusi akan dilakukan sebanyak 2 kali sesuai dengan arsitektur pada 2.11. Setiap sehabis dilakukan operasi konvolusi, akan dilakukan operasi pada lapisan *Max Pooling* yang akan mengurangi dimensi dari masukan. Jumlah lapisan *Max Pooling* akan disesuaikan yaitu 2 lapisan, 1 lapisan dan tanpa lapisan *Max Pooling* atau dapat juga disebut dengan *Nonpooling* CNN-LSTM.
5: Proses akan dilanjutkan dengan memberikan nilai keluaran dari CNN kepada lapisan LSTM.
6: Proses pada lapisan LSTM akan menggunakan fungsi aktivasi *tanh* dan ReLU seperti yang ada pada 2.1.4. Pada lapisan ini akan dihitung nilai dari *input gate*, *forget gate*, *cell state* dan *output gate*.
7: Hitung hasil keluaran lapisan *dense* atau *fully connected layer* untuk menjadi nilai keluaran berupa prediksi pemakaian obat.
8: Simpan model untuk digunakan pada saat pengujian.

Pada langkah ke-4 di Algoritme 3.1, total lapisan Max Pooling akan dibagi menjadi 3 jenis arsitektur, yaitu arsitektur dengan 2 lapisan *Max Pooling* berarti setiap konvolusi selesai dilakukan akan dilakukan operasi *Max Pooling*, arsitektur dengan 1 lapisan *Max Pooling* berarti setelah 2 kali konvolusi akan dilakukan 1 kali operasi *Max Pooling* dan arsitektur tanpa lapisan *Max Pooling* atau *Nonpooling* CNN-LSTM berarti pada lapisan CNN hanya akan dilakukan operasi konvolusi saja tanpa melakukan operasi *Max Pooling*.

3.3.2 Proses Pengujian

Pada penelitian ini, proses pengujian model digambarkan pada Gambar 3.3.



Gambar 3.5 *Flowchart* pengujian sistem prediksi pemakaian obat

Berikut ini adalah uraian proses pengujian dari *flowchart* pada Gambar 3.5 yang dilakukan dalam penelitian ini.

1. Masukan data pemakaian obat dengan ukuran 365×1 yang akan dilakukan untuk prediksi sesuai dengan *forecasting window* pendek maupun panjang.
2. Ambil model dari CNN-LSTM dengan dan tanpa lapisan *Max Pooling* yang sudah dihasilkan pada proses pelatihan.
3. Gunakan model tersebut untuk melakukan prediksi pemakaian obat dari data uji. Model ini digunakan untuk melakukan proses konvolusi tanpa *Max Pooling* pada lapisan *Nonpooling* CNN serta proses konvolusi dengan *Max Pooling* pada CNN untuk mempelajari *seasonality* jangka pendek yang kemudian akan diberikan kepada lapisan LSTM yang nantinya akan menghasilkan hasil prediksi.

3.4 Analisis Manual

Pada bagian ini, dilakukan analisis tahapan proses yang akan dilakukan dalam sistem. Analisis dilakukan dengan contoh dan ilustrasi perhitungan manual. Analisis manual ini akan menggunakan data 5 hari untuk memprediksi 1 hari ke depan.

3.4.1 *Dataset*

Dataset yang digunakan dalam penelitian ini adalah data pemakaian obat per hari selama 1 tahun pada tahun 2021 di sebuah rumah sakit yang berada di Jakarta, Indonesia. *Dataset* didapatkan dari sebuah perusahaan Sistem Informasi Menejemen Rumah Sakit yang berada di Jakarta, Indonesia dan diberikan secara anonimus baik untuk nama obat dan nama rumah sakit. Data pemakaian obat ini berbentuk dalam berkas .csv (*comma seperated value*) yang berisikan 365 baris tanggal selama tahun 2021 dan jumlah pemakaian obat sesuai tanggalnya. Visualisasi keseluruhan data terdapat pada Gambar 3.4.

3.4.2 *Nonpooling* CNN-LSTM

Berikut ini akan dijelaskan perhitungan untuk proses-proses yang dilalui dalam pembangunan model *Nonpooling* CNN-LSTM berdasarkan algoritme yang sudah dijabarkan pada bagian 3.3.1. Arsitektur ini terdiri atas 3 lapisan, yaitu lapisan *Nonpooling* CNN, lapisan LSTM dan lapisan *dense* atau *fully connected layer*.

3.4.2.1 Perhitungan *Nonpooling* CNN

Masukan berupa larik 1 dimensi dengan ukuran 365×1 akan dimasukan ke lapisan *Nonpooling* CNN terlebih dahulu. Contoh nilai larik terdapat pada tabel 3.1 memiliki ukuran 5×1 dan akan digunakan untuk melakukan operasi konvolusi antara matriks dengan *kernel*. Pada penelitian ini, operasi konvolusi akan berjalan secara 1 arah ke samping dikarenakan data yang digunakan merupakan data *time-series* seperti yang telah dijelaskan pada 2.1.2.1

Tabel 3.1 Contoh larik

277	417	494	417	305

Kernel yang digunakan dalam penelitian ini berukuran 30×1, di mana 1 merupakan dimensi *height* dari larik masukan dan 30 merupakan dimensi *width* dari larik masukan agar proses konvolusi dapat melihat 30 nilai sekaligus. Dengan mengoperasikan nilai *kernel* kepada larik masukan, akan didapatkan sebuah *feature map* satu dimensi. Proses konvolusi bergerak secara satu dimensi, di mana pergerakan *kernel* adalah ke arah samping kanan. Nilai *kernel* pada contoh perhitungan ini berukuran 2×1. Nilai awal *kernel* dalam contoh perhitungan manual dan yang digunakan dalam penelitian didapatkan dengan inisialisasi awal acak terdistribusi normal sesuai dengan Persamaan 2.4, dengan nilai *mean* = 0 dan standar deviasi = 0,5. Contoh *kernel* tersebut dapat dilihat pada tabel 3.2.

Tabel 3.2 Contoh nilai *kernel* 2×1

0.1044	0.030146

Perhitungan untuk sel [0, 0], sel [1,0], dan sel-sel selanjutnya memiliki pola yang sama yang dapat dilihat pada Persamaan 2.5. Perhitungan dilakukan hingga mencapai sel [x, y] yang berada pada ujung kanan bawah larik masukan. Hasil perkalian konvolusi $w \times x$ disebut sebagai *feature map*. Proses perkalian dilakukan menggunakan *stride* (pergerakan indeks larik) bernilai 1. Sebelum dilakukan perhitungan, larik masukan akan diberikan *padding* terlebih dahulu. Hal ini

bertujuan agar larik hasil konvolusi memiliki panjang yang sama dengan larik masukan. Nilai *padding* yang akan digunakan adalah nilai 0. Maka dari itu larik yang akan dilakukan operasi konvolusi menjadi seperti dibawah ini.

Tabel 3.3 Contoh larik

277	417	494	417	305	0

Hasil perhitungan operasi konvolusi yang dilakukan merupakan konvolusi 1 dimensi seperti ilustrasi di Gambar 2.4 yang hasil perhitungannya sesuai dengan Persamaan 2.5 dan Persamaan 2.1. Nilai *bias* yang digunakan adalah 0. Hasil konvolusi antara larik dan *kernel* pada contoh kasus sel [0, 0] adalah sebagai berikut.

$$\begin{aligned} v_k &= (w_{k_{0,0}} x_{0,0} + b_{0,0}) + (w_{k_{1,0}} x_{1,0} + b_{1,0}) \\ &= (277 * 0.1044 + 0) + (417 * 0.030146 + 0) \\ &= 41.489682 \end{aligned}$$

Setelah konvolusi selesai dilakukan, didapatkan *feature map* berukuran 5×1 seperti pada Tabel 3.4. Ukuran *feature map* dapat berubah bergantung pada *kernel* yang digunakan dan *stride* yang dipakai pada lapisan konvolusi. Jumlah *filter* akan berpengaruh kepada jumlah *feature map* yang dihasilkan. Jika *filter* yang digunakan berjumlah 32 pada satu lapisan konvolusi, maka akan dihasilkan 32 *feature map*. Jumlah *filter* yang digunakan bergantung kepada arsitektur yang dipakai. Tabel 3.4 merupakan *feature map* hasil konvolusi yang menggunakan 1 *filter*.

Tabel 3.4 Contoh *feature map* hasil konvolusi

41.489682	58.426924	64.144482	52.72933	42.784998

Karena lapisan pertama merupakan lapisan *Nonpooling* CNN, maka *feature map* yang telah dihasilkan tidak akan dilakukan perhitungan *Max Pooling*. Maka dari itu nilai yang ada dalam *feature map* tersebut akan dimasukan ke dalam lapisan

LSTM yang akan dijelaskan lebih lanjut pada bagian 3.4.2.2.

3.4.2.2 Perhitungan LSTM

Setelah dilakukan perhitungan pada lapisan *Nonpooling* CNN dan menghasilkan *feature map*, langkah berikutnya adalah melakukan perhitungan pada lapisan LSTM. Seperti yang telah dibahas sebelumnya pada Bagian 2.1.4, lapisan ini akan menghitung *input gate*, *forget gate*, *cell state* dan *output gate*. Langkah pertama adalah melakukan inisialisasi nilai bobot dan bias untuk *input gate*, *forget gate*, *cell* dan *output gate*. Inisialisasi ini menggunakan *gausial distribution* dengan *mean* = 0 dan *std* = 0,5. Sesuai pada Bagian 2.1.4, masing-masing *gate* memiliki 2 bobot dan 1 bias, maka dari itu terdapat 8 bobot dan 4 bias yang harus diinisialisasikan terlebih dahulu. Berikut adalah inisialisasi dari masing-masing *gate*. Perhitungan akan menggunakan fungsi aktivasi ReLU dan *tanh* seperti yang sudah dibahas pada bagian 2.1.3. Fungsi aktivasi ReLU akan digunakan diterapkan pada perhitungan *gate* sedangkan fungsi aktivasi *tanh* akan diterapkan pada perhitungan *candidate state* dan nilai keluaran.

1. Inisialsisasi nilai bobot dan *bias* pada *input gate*.

$$W_u = \begin{bmatrix} -0.64220 & -0.13910 & 0.42470 & -0.74550 & -0.89280 \\ 0.15150 & 0.21010 & 0.58110 & -0.020940 & -0.32080 \\ 0.25550 & 0.020920 & -0.20380 & -0.31210 & 0.28700 \\ 0.80000 & -0.56660 & -0.13470 & -0.83260 & -0.52940 \\ 0.53240 & 0.077130 & -0.14730 & 0.83840 & -0.19260 \end{bmatrix}$$

$$R_u = \begin{bmatrix} -0.33270 & -1.2440 & 0.26770 & -0.20980 & 0.074580 \\ 0.47430 & -0.0030220 & 0.62060 & 0.21340 & 0.026960 \\ 0.66820 & 0.19730 & -0.35240 & 0.34950 & 0.0025130 \\ 0.26230 & 0.63160 & 0.40020 & -0.19910 & 0.89960 \\ -0.84100 & 0.30370 & 0.69530 & -0.36870 & 0.25060 \end{bmatrix}$$

$$b_u = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

2. Inisialsisasi nilai bobot dan *bias* pada *candidate state*.

$$W_{\tilde{h}} = \begin{bmatrix} -0.042160 & -0.40840 & -0.32040 & -0.29200 & -0.74820 \\ 0.32400 & 0.58910 & 0.25840 & 0.51780 & 0.60620 \\ -0.22980 & -0.23720 & -0.62320 & 0.49310 & 0.24230 \\ 0.50480 & -1.0130 & -0.71360 & 0.33950 & -0.61370 \\ -0.39210 & 0.12640 & -0.022060 & 0.80220 & 0.036120 \end{bmatrix}$$

$$R_{\tilde{h}} = \begin{bmatrix} -0.40520 & 0.16320 & 0.21780 & -0.0071840 & 0.41630 \\ 0.70990 & 1.5030 & 0.34580 & -0.0048140 & -0.71000 \\ -0.65940 & 0.17790 & 0.36630 & 0.12870 & 0.39480 \\ 0.60180 & -0.13430 & -0.39560 & -0.67070 & -0.97020 \\ 0.0018580 & 0.60030 & -0.37040 & -0.49820 & -0.73800 \end{bmatrix}$$

$$b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

3. Inisialsisasi nilai bobot dan *bias* pada *forget gate*.

$$W_f = \begin{bmatrix} -0.33390 & -0.055780 & 0.060340 & 0.17870 & -0.19800 \\ 0.35340 & -0.56940 & -0.41500 & 0.45970 & -0.51810 \\ -0.21700 & -0.56380 & 1.0630 & 0.22850 & -0.16080 \\ -0.18710 & -0.35190 & 0.17220 & -0.55190 & 0.37890 \\ -1.0690 & 0.10680 & 0.12700 & 0.008233 & -0.39140 \end{bmatrix}$$

$$R_f = \begin{bmatrix} 0.0056150 & -0.44130 & 0.88370 & 0.65300 & 0.17920 \\ 0.45240 & 0.93330 & 0.41220 & 0.48090 & -0.28720 \\ 0.18380 & -0.20220 & -0.64060 & -0.64900 & -0.26450 \\ -0.65270 & -0.22710 & -0.17060 & -0.086930 & -0.71810 \\ 0.57990 & 0.33270 & -0.41020 & -0.69790 & -0.11650 \end{bmatrix}$$

$$b_f = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

4. Inisialsisasi nilai bobot dan *bias* pada *output gate*.

$$W_o = \begin{bmatrix} -0.48530 & -0.042330 & -0.49050 & -0.22460 & -0.18820 \\ 0.063690 & -0.23460 & 0.16280 & 0.53000 & -0.15680 \\ 1.0220 & 0.74460 & 0.012930 & 0.40940 & -0.21790 \\ -0.045900 & -0.087540 & 0.16220 & 1.2770 & -0.87750 \\ 0.011170 & -0.83700 & -0.19930 & -0.86200 & -0.82720 \end{bmatrix}$$

$$R_o = \begin{bmatrix} 0.52350 & -0.69080 & 0.47550 & 0.31400 & -0.28620 \\ 0.56460 & 0.63700 & 0.22930 & 0.15670 & -0.20780 \\ 0.13610 & -0.35800 & -0.16660 & 0.16340 & 0.074710 \\ 0.19950 & 0.66100 & -0.020360 & -0.38690 & -0.59090 \\ 0.023970 & 0.31670 & 0.72550 & 0.50990 & -0.015850 \end{bmatrix}$$

$$b_o = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

5. Inisialsisasi nilai pada *cell state* dan *candidate state* untuk t ke-0.

$$y_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad c_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Setelah melakukan inisialisasi, tahap berikutnya adalah melakukan perhitungan untuk masing-masing *gate*. Tahapan di bawah ini merupakan contoh perhitungan

untuk masing-masing *gate* hingga menjadi nilai akhir.

1. Perhitungan *input gate* sesuai dengan Persamaan 2.9.

$$
\begin{aligned}
f_u[1] = &\begin{bmatrix} -0.64220 & -0.13910 & 0.42470 & -0.74550 & -0.89280 \\ 0.15150 & 0.21010 & 0.58110 & -0.020940 & -0.32080 \\ 0.25550 & 0.020920 & -0.20380 & -0.31210 & 0.28700 \\ 0.80000 & -0.56660 & -0.13470 & -0.83260 & -0.52940 \\ 0.53240 & 0.07710 & -0.14730 & 0.83840 & -0.19260 \end{bmatrix} \odot \\
&\begin{bmatrix} 41.489682 \\ 58.426924 \\ 64.144482 \\ 52.72933 \\ 42.784998 \end{bmatrix} + \\
&\begin{bmatrix} -0.33270 & -1.2440 & 0.26770 & -0.20980 & 0.074580 \\ 0.47430 & -0.0030220 & 0.62060 & 0.21340 & 0.026960 \\ 0.66820 & 0.19730 & -0.35240 & 0.34950 & 0.0025130 \\ 0.26230 & 0.63160 & 0.40020 & -0.19910 & 0.89960 \\ -0.84100 & 0.30370 & 0.69530 & -0.36870 & 0.25060 \end{bmatrix} \odot \\
&\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\end{aligned}
$$

Dari perhitungan di atas, diperoleh matriks untuk *input gate* seperti berikut:

$$f_u[1] = f\left(\begin{bmatrix} -85.03785 \\ 41.00596 \\ -5.42726 \\ -75.10602 \\ 53.11321 \end{bmatrix}\right)$$

Berikut adalah hasil perhitungan *input gate* yang telah diterapkan fungsi aktivasi ReLU:

$$f_u[1] = \begin{bmatrix} 0 \\ 41.00596 \\ 0 \\ 0 \\ 53.11321 \end{bmatrix}$$

2. Perhitungan *candidate state* sesuai dengan Persamaan 2.10.

$$f_{\bar{h}}[1] = \begin{bmatrix} -0.042160 & -0.40840 & -0.32040 & -0.29200 & -0.74820 \\ 0.32400 & 0.58910 & 0.25840 & 0.51780 & 0.60620 \\ -0.22980 & -0.23720 & -0.62320 & 0.49310 & 0.24230 \\ 0.50480 & -1.0130 & -0.71360 & 0.33950 & -0.61370 \\ -0.39210 & 0.12640 & -0.022060 & 0.80220 & 0.036120 \end{bmatrix} \odot \begin{bmatrix} 41.489682 \\ 58.426924 \\ 64.144482 \\ 52.72933 \\ 42.784998 \end{bmatrix} + \begin{bmatrix} -0.40520 & 0.16320 & 0.21780 & -0.0071840 & 0.41630 \\ 0.70990 & 1.5030 & 0.34580 & -0.0048140 & -0.71000 \\ -0.65940 & 0.17790 & 0.36630 & 0.12870 & 0.39480 \\ 0.60180 & -0.13430 & -0.39560 & -0.67070 & -0.97020 \\ 0.0018580 & 0.60030 & -0.37040 & -0.49820 & -0.73800 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dari perhitungan di atas, diperoleh matriks untuk *candidate state* seperti berikut:

$$f_{\tilde{h}}[1] = g_1 \left(\begin{bmatrix} -93.57135 \\ 117.67640 \\ -27.00039 \\ -92.37153 \\ 33.54689 \end{bmatrix} \right)$$

Berikut adalah hasil perhitungan *candidate state* yang telah diterapkan fungsi aktivasi *tanh*:

$$f_{\tilde{h}}[1] = \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

3. Perhitungan *forget gate* sesuai dengan Persamaan 2.11.

$$f_f[1] = \begin{bmatrix} -0.33390 & -0.055780 & 0.060340 & 0.17870 & -0.19800 \\ 0.35340 & -0.56940 & -0.41500 & 0.45970 & -0.51810 \\ -0.21700 & -0.56380 & 1.0630 & 0.22850 & -0.16080 \\ -0.18710 & -0.35190 & 0.17220 & -0.55190 & 0.37890 \\ -1.0690 & 0.10680 & 0.12700 & 0.008233 & -0.39140 \end{bmatrix} \odot \begin{bmatrix} 41.489682 \\ 58.426924 \\ 64.144482 \\ 52.72933 \\ 42.784998 \end{bmatrix} +$$

$$\begin{bmatrix} 0.0056150 & -0.44130 & 0.88370 & 0.65300 & 0.17920 \\ 0.45240 & 0.93330 & 0.41220 & 0.48090 & -0.28720 \\ 0.18380 & -0.20220 & -0.64060 & -0.64900 & -0.26450 \\ -0.65270 & -0.22710 & -0.17060 & -0.086930 & -0.71810 \\ 0.57990 & 0.33270 & -0.41020 & -0.69790 & -0.11650 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dari perhitungan di atas, diperoleh matriks untuk *forget gate* seperti berikut:

$$f_f[1] = f\left(\begin{bmatrix} -12.29067 \\ -43.15303 \\ 31.41004 \\ -30.16755 \\ -46.27805 \end{bmatrix}\right)$$

Berikut adalah hasil perhitungan *forget gate* yang telah diterapkan fungsi aktivasi ReLU:

$$f_f[1] = \begin{bmatrix} 0 \\ 0 \\ 31.41004 \\ 0 \\ 0 \end{bmatrix}$$

4. Perhitungan *cell state* sesuai dengan Persamaan 2.12.

$$h[1] = \begin{bmatrix} 0 \\ 41.00596 \\ 0 \\ 0 \\ 53.11321 \end{bmatrix} \odot \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 31.41004 \\ 0 \\ 0 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dari perhitungan di atas, diperoleh matriks untuk *cell state* seperti berikut:

$$h[1] = \begin{bmatrix} 0 \\ -41.01 \\ 0 \\ 0 \\ -53.11 \end{bmatrix}$$

5. Perhitungan *output gate* sesuai dengan Persamaan 2.13.

$$f_o[1] = \begin{bmatrix} -0.48530 & -0.042330 & -0.49050 & -0.22460 & -0.18820 \\ 0.063690 & -0.23460 & 0.16280 & 0.53000 & -0.15680 \\ 1.0220 & 0.74460 & 0.012930 & 0.40940 & -0.21790 \\ -0.045900 & -0.087540 & 0.16220 & 1.2770 & -0.87750 \\ 0.011170 & -0.83700 & -0.19930 & -0.86200 & -0.82720 \end{bmatrix} \odot \begin{bmatrix} 41.489682 \\ 58.426924 \\ 64.144482 \\ 52.72933 \\ 42.784998 \end{bmatrix} +$$

$$\begin{bmatrix} 0.52350 & -0.69080 & 0.47550 & 0.31400 & -0.28620 \\ 0.56460 & 0.63700 & 0.22930 & 0.15670 & -0.20780 \\ 0.13610 & -0.35800 & -0.16660 & 0.16340 & 0.074710 \\ 0.19950 & 0.66100 & -0.020360 & -0.38690 & -0.59090 \\ 0.023970 & 0.31670 & 0.72550 & 0.50990 & -0.015850 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dari perhitungan di atas, diperoleh matriks untuk *output gate* seperti berikut:

$$f_o[1] = f\left(\begin{bmatrix} -73.96616 \\ 20.61610 \\ 101.44409 \\ 33.17668 \\ -142.06832 \end{bmatrix}\right)$$

Berikut adalah hasil perhitungan *output gate* yang telah diterapkan fungsi aktivasi ReLU:

$$f_o[1] = \begin{bmatrix} 0 \\ 20.6161 \\ 101.44409 \\ 33.17668 \\ 0 \end{bmatrix}$$

Hasil dari perhitungan untuk *output gate* dan *cell state* akan digunakan untuk mendapatkan nilai keluaran sesuai dengan Persamaan 2.14.

$$y[1] = \begin{bmatrix} 0 \\ 20.6161 \\ 101.44409 \\ 33.17668 \\ 0 \end{bmatrix} \odot g_2\left(\begin{bmatrix} 0 \\ -41.01 \\ 0 \\ 0 \\ -53.11 \end{bmatrix}\right)$$

Dari perhitungan di atas, diperoleh matriks untuk nilai keluaran seperti berikut:

$$y[1] = \begin{bmatrix} 0 \\ 20.6161 \\ 101.44409 \\ 33.17668 \\ 0 \end{bmatrix}$$

3.4.2.3 Perhitungan *Dense*

Setelah dilakukan perhitungan pada lapisan LSTM, langkah berikutnya adalah melakukan perhitungan pada lapisan *dense* atau *fully connected layer*. Hal ini bertujuan untuk menghubungkan seluruh *neuron* yang ada untuk menghasilkan nilai keluaran.

Pada lapisan *dense*, akan dilakukan perhitungan sesuai dengan Persamaan 2.6. Ukuran *kernel* pada contoh perhitungan manual ini adalah 5×1 yang dihasilkan dari nilai acak terdistribusi normal sesuai dengan Persamaan 2.4. Contoh nilai *kernel* untuk lapisan *dense* dapat dilihat pada tabel 3.5 berikut. Contoh nilai *kernel* yang digunakan adalah sebagai berikut.

Tabel 3.5 Contoh *kernel* pada lapisan *dense*

0.298860	0.22722	-0.869150	1.13740	-0.542370

Perhitungan pada lapisan ini dapat dilihat sebagai berikut. Hasil keluaran dari lapisan LSTM akan dilakukan perhitungan *dot product* dengan *kernel*. Fungsi aktivasi yang digunakan pada lapisan *dense* ini adalah ReLU seperti pada Bagian 2.1.3.1. Hasil dari lapisan *dense* berikut merupakan prediksi pemakaian obat di hari keenam berdasarkan data 5 hari yang dipelajari.

$$
\begin{aligned}
y &= f\left(W^T x + b\right) \\
&= f(\\
&\quad (0.298860 * 0) + (0.227220 * 20.6161) + \\
&\quad (-0.869150 * 101.44409) + (1.13740 * 33.17668) + \\
&\quad (-0.542370 * 0) \\
&\quad) \\
&= f(91.5011695) \\
&= 91.5011695
\end{aligned}
$$

3.4.3 CNN-LSTM

Berikut ini akan dijelaskan perhitungan untuk proses-proses yang dilalui dalam pembangunan model CNN-LSTM berdasarkan algoritme yang sudah dijabarkan pada bagian 3.3.1. Arsitektur ini terdiri atas 3 lapisan, yaitu lapisan CNN, lapisan LSTM dan lapisan *dense* atau *fully connected layer*. Pada bagian ini, akan digunakan konfigurasi yang sama seperti pada Bagian 3.4.2 namun yang membedakan adalah adanya perhitungan pada lapisan Max Pooling pada CNN.

3.4.3.1 Perhitungan CNN

Masukan berupa larik 1 dimensi dengan ukuran 365×1 akan dimasukan ke lapisan *Nonpooling* CNN terlebih dahulu. Contoh nilai larik terdapat pada tabel 3.6 memiliki ukuran 5×1 dan akan digunakan untuk melakukan operasi konvolusi antara matriks dengan *kernel*. Pada penelitian ini, operasi konvolusi akan berjalan secara 1 arah ke samping dikarenakan data yang digunakan merupakan data *time-series* seperti yang telah dijelaskan pada 2.1.2.1

Tabel 3.6 Contoh larik

277	417	494	417	305

Kernel yang digunakan dalam penelitian ini berukuran 30×1, di mana 1 merupakan dimensi *height* dari larik masukan dan 30 merupakan dimensi *width* dari larik masukan agar proses konvolusi dapat melihat 30 nilai sekaligus. Dengan mengoperasikan nilai *kernel* kepada larik masukan, akan didapatkan sebuah *feature map* satu dimensi. Proses konvolusi bergerak secara satu dimensi, di mana pergerakan *kernel* adalah ke arah samping kanan. Nilai *kernel* pada contoh

perhitungan ini berukuran 2×1. Nilai awal *kernel* dalam contoh perhitungan manual dan yang digunakan dalam penelitian didapatkan dengan inisialisasi awal acak terdistribusi normal sesuai dengan Persamaan 2.4, dengan nilai *mean* = 0 dan standar deviasi = 0,5. Contoh *kernel* pada tabel 3.7 merupakan konfigurasi *kernel* yang sama dengan pada tabel 3.2.

Tabel 3.7 Contoh nilai *kernel* 2×1

0.1044	0.030146

Perhitungan untuk sel [0, 0], sel [1,0], dan sel-sel selanjutnya memiliki pola yang sama yang dapat dilihat pada Persamaan 2.5. Perhitungan dilakukan hingga mencapai sel [x, y] yang berada pada ujung kanan bawah larik masukan. Hasil perkalian konvolusi $w \times x$ disebut sebagai *feature map*. Proses perkalian dilakukan menggunakan *stride* (pergerakan indeks larik) bernilai 1. Sebelum dilakukan perhitungan, larik masukan akan diberikan *padding* terlebih dahulu. Hal ini bertujuan agar larik hasil konvolusi memiliki panjang yang sama dengan larik masukan. Nilai *padding* yang akan digunakan adalah nilai 0. Maka dari itu larik yang akan dilakukan operasi konvolusi menjadi seperti dibawah ini.

Tabel 3.8 Contoh larik

277	417	494	417	305	0

Hasil perhitungan operasi konvolusi yang dilakukan merupakan konvolusi 1 dimensi seperti ilustrasi di Gambar 2.4 yang hasil perhitungannya sesuai dengan Persamaan 2.5 dan Persamaan 2.1. Nilai *bias* yang digunakan adalah 0. Hasil konvolusi antara larik dan *kernel* pada contoh kasus sel [0, 0] adalah sebagai berikut.

$$\begin{aligned} v_k &= (w_{k_{0,0}} x_{0,0} + b_{0,0}) + (w_{k_{1,0}} x_{1,0} + b_{1,0}) \\ &= (277 * 0.1044 + 0) + (417 * 0.030146 + 0) \\ &= 41.489682 \end{aligned}$$

Setelah konvolusi selesai dilakukan, didapatkan *feature map* berukuran 5×1 seperti pada Tabel 3.9. Ukuran *feature map* dapat berubah bergantung pada *kernel*

yang digunakan dan *stride* yang dipakai pada lapisan konvolusi. Jumlah *filter* akan berpengaruh kepada jumlah *feature map* yang dihasilkan. Jika *filter* yang digunakan berjumlah 32 pada satu lapisan konvolusi, maka akan dihasilkan 32 *feature map*. Jumlah *filter* yang digunakan bergantung kepada arsitektur yang dipakai. Tabel 3.9 merupakan *feature map* hasil konvolusi yang menggunakan 1 *filter*.

Tabel 3.9 Contoh *feature map* hasil konvolusi

41.489682	58.426924	64.144482	52.72933	42.784998

Setelah melakukan konvolusi, langkah berikutnya adalah melakukan operasi *Max Pooling* seperti yang ada pada Bagian 2.1.2.2 Untuk contoh analisis manual ini, parameter *spatial extent* yang akan digunakan adalah 2 dengan *stride* adalah 1. Operasi akan dimulai dari sel [0,0] dan [1,0] kemudian bergerak 1 sel ke [1,0] dan [2,0] dan seterusnya hingga mencapai keseluruhan data. Setiap operasinya akan mencari nilai maksimum dari 2 nilai yang dibandingkan. Tabel 3.10 merupakan hasil dari operasi pada lapisan *Max Pooling*.

Tabel 3.10 Contoh *feature map* hasil konvolusi

58.426924	64.144482	64.144482	52.72933

Terlihat pada tabel 3.10, jumlah panjang data berkurang menjadi 4 data. Langkah berikutnya adalah memberikan nilai hasil *Max Pooling* ke dalam lapisan LSTM yang akan dijelaskan lebih lanjut pada bagian 3.4.3.2.

3.4.3.2 Perhitungan LSTM

Setelah dilakukan perhitungan pada lapisan CNN dan menghasilkan *feature map*, langkah berikutnya adalah melakukan perhitungan pada lapisan LSTM. Seperti yang telah dibahas sebelumnya pada Bagian 2.1.4, lapisan ini akan menghitung *input gate*, *forget gate*, *cell state* dan *output gate*. Langkah pertama adalah melakukan inisialisasi nilai bobot dan bias untuk *input gate*, *forget gate*, *cell* dan *output gate*. Inisialisasi ini menggunakan *gausial distribution* dengan *mean* = 0 dan *std* = 0,5. Sesuai pada Bagian 2.1.4, masing-masing *gate* memiliki 2 bobot dan 1 bias, maka dari itu terdapat 8 bobot dan 4 bias yang harus diinisialisasikan terlebih dahulu. Berikut adalah inisialisasi dari masing-masing *gate*. Perhitungan akan menggunakan fungsi aktivasi ReLU dan *tanh* seperti yang sudah dibahas

pada bagian 2.1.3. Fungsi aktivasi ReLU akan digunakan diterapkan pada perhitungan *gate* sedangkan fungsi aktivasi *tanh* akan diterapkan pada perhitungan *candidate state* dan nilai keluaran.

1. Inisialsisasi nilai bobot dan *bias* pada *input gate*.

$$W_u = \begin{bmatrix} -0.64220 & -0.13910 & 0.42470 & -0.74550 \\ 0.15150 & 0.21010 & 0.58110 & -0.020940 \\ 0.25550 & 0.020920 & -0.20380 & -0.31210 \\ 0.80000 & -0.56660 & -0.13470 & -0.83260 \end{bmatrix}$$

$$R_u = \begin{bmatrix} -0.33270 & -1.2440 & 0.26770 & -0.20980 \\ 0.47430 & -0.0030220 & 0.62060 & 0.21340 \\ 0.66820 & 0.19730 & -0.35240 & 0.34950 \\ 0.26230 & 0.63160 & 0.40020 & -0.19910 \end{bmatrix}$$

$$b_u = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

2. Inisialsisasi nilai bobot dan *bias* pada *candidate state*.

$$W_{\bar{h}} = \begin{bmatrix} -0.042160 & -0.40840 & -0.32040 & -0.29200 \\ 0.32400 & 0.58910 & 0.25840 & 0.51780 \\ -0.22980 & -0.23720 & -0.62320 & 0.49310 \\ 0.50480 & -1.0130 & -0.71360 & 0.33950 \end{bmatrix}$$

$$R_{\bar{h}} = \begin{bmatrix} -0.40520 & 0.16320 & 0.21780 & -0.0071840 \\ 0.70990 & 1.5030 & 0.34580 & -0.0048140 \\ -0.65940 & 0.17790 & 0.36630 & 0.12870 \\ 0.60180 & -0.13430 & -0.39560 & -0.67070 \end{bmatrix}$$

$$b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

3. Inisialsisasi nilai bobot dan *bias* pada *forget gate*.

$$W_f = \begin{bmatrix} -0.33390 & -0.055780 & 0.060340 & 0.17870 \\ 0.35340 & -0.56940 & -0.41500 & 0.45970 \\ -0.21700 & -0.56380 & 1.0630 & 0.22850 \\ -0.18710 & -0.35190 & 0.17220 & -0.55190 \end{bmatrix}$$

$$R_f = \begin{bmatrix} 0.0056150 & -0.44130 & 0.88370 & 0.65300 \\ 0.45240 & 0.93330 & 0.41220 & 0.48090 \\ 0.18380 & -0.20220 & -0.64060 & -0.64900 \\ -0.65270 & -0.22710 & -0.17060 & -0.086930 \end{bmatrix}$$

$$b_f = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

4. Inisialsisasi nilai bobot dan *bias* pada *output gate*.

$$W_o = \begin{bmatrix} -0.48530 & -0.042330 & -0.49050 & -0.22460 \\ 0.063690 & -0.23460 & 0.16280 & 0.53000 \\ 1.0220 & 0.74460 & 0.012930 & 0.40940 \\ -0.045900 & -0.087540 & 0.16220 & 1.2770 \end{bmatrix}$$

$$R_o = \begin{bmatrix} 0.52350 & -0.69080 & 0.47550 & 0.31400 \\ 0.56460 & 0.63700 & 0.22930 & 0.15670 \\ 0.13610 & -0.35800 & -0.16660 & 0.16340 \\ 0.19950 & 0.66100 & -0.020360 & -0.38690 \end{bmatrix}$$

$$b_o = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

5. Inisialsisasi nilai pada *cell state* dan *candidate state* untuk t ke-0.

$$y_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad c_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Setelah melakukan inisialisasi, tahap berikutnya adalah melakukan perhitungan untuk masing-masing *gate*. Tahapan di bawah ini merupakan contoh perhitungan untuk masing-masing *gate* hingga menjadi nilai akhir.

1. Perhitungan *input gate* sesuai dengan Persamaan 2.9.

$$f_u[1] = \begin{bmatrix} -0.64220 & -0.13910 & 0.42470 & -0.74550 \\ 0.15150 & 0.21010 & 0.58110 & -0.020940 \\ 0.25550 & 0.020920 & -0.20380 & -0.31210 \\ 0.80000 & -0.56660 & -0.13470 & -0.83260 \end{bmatrix} \odot \begin{bmatrix} 58.426924 \\ 64.144482 \\ 64.144482 \\ 52.72933 \end{bmatrix} + \begin{bmatrix} -0.33270 & -1.2440 & 0.26770 & -0.20980 \\ 0.47430 & -0.0030220 & 0.62060 & 0.21340 \\ 0.66820 & 0.19730 & -0.35240 & 0.34950 \\ 0.26230 & 0.63160 & 0.40020 & -0.19910 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dari perhitungan di atas, diperoleh matriks untuk *input gate* seperti berikut:

$$f_u[1] = f\left(\begin{bmatrix} -58.51182205 \\ 58.49864097 \\ -13.25948768 \\ -42.14542618 \end{bmatrix}\right)$$

Berikut adalah hasil perhitungan *input gate* yang telah diterapkan fungsi aktivasi ReLU:

$$f_u[1] = \begin{bmatrix} 0 \\ 58.49864097 \\ 0 \\ 0 \end{bmatrix}$$

2. Perhitungan *candidate state* sesuai dengan Persamaan 2.10.

$$f_{\tilde{h}}[1] = \begin{bmatrix} -0.042160 & -0.40840 & -0.32040 & -0.29200 \\ 0.32400 & 0.58910 & 0.25840 & 0.51780 \\ -0.22980 & -0.23720 & -0.62320 & 0.49310 \\ 0.50480 & -1.0130 & -0.71360 & 0.33950 \end{bmatrix} \odot \begin{bmatrix} 58.426924 \\ 64.144482 \\ 64.144482 \\ 52.72933 \end{bmatrix} + \begin{bmatrix} -0.40520 & 0.16320 & 0.21780 & -0.0071840 \\ 0.70990 & 1.5030 & 0.34580 & -0.0048140 \\ -0.65940 & 0.17790 & 0.36630 & 0.12870 \\ 0.60180 & -0.13430 & -0.39560 & -0.67070 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dari perhitungan di atas, diperoleh matriks untuk *candidate state* seperti berikut:

$$f_{\tilde{h}}[1] = g_1 \left(\begin{bmatrix} -64.60874196 \\ 100.5960189 \\ -42.61558683 \\ -63.35634385 \end{bmatrix} \right)$$

Berikut adalah hasil perhitungan *candidate state* yang telah diterapkan fungsi aktivasi *tanh*:

$$f_{\tilde{h}}[1] = \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

3. Perhitungan *forget gate* sesuai dengan Persamaan 2.11.

$$f_f[1] = \begin{bmatrix} -0.33390 & -0.055780 & 0.060340 & 0.17870 \\ 0.35340 & -0.56940 & -0.41500 & 0.45970 \\ -0.21700 & -0.56380 & 1.0630 & 0.22850 \\ -0.18710 & -0.35190 & 0.17220 & -0.55190 \end{bmatrix} \odot \begin{bmatrix} 58.426924 \\ 64.144482 \\ 64.144482 \\ 52.72933 \end{bmatrix} + \begin{bmatrix} 0.0056150 & -0.44130 & 0.88370 & 0.65300 \\ 0.45240 & 0.93330 & 0.41220 & 0.48090 \\ 0.18380 & -0.20220 & -0.64060 & -0.64900 \\ -0.65270 & -0.22710 & -0.17060 & -0.086930 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dari perhitungan di atas, diperoleh matriks untuk *forget gate* seperti berikut:

$$f_f[1] = f\left(\begin{bmatrix} -9.793519815 \\ -18.25608014 \\ 31.39093481 \\ -51.55975812 \end{bmatrix}\right)$$

Berikut adalah hasil perhitungan *forget gate* yang telah diterapkan fungsi aktivasi ReLU:

$$f_f[1] = \begin{bmatrix} 0 \\ 0 \\ 31.39093481 \\ 0 \end{bmatrix}$$

4. Perhitungan *cell state* sesuai dengan Persamaan 2.12.

$$h[1] = \begin{bmatrix} 0 \\ 58.49864097 \\ 0 \\ 0 \end{bmatrix} \odot \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 31.39093481 \\ 0 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dari perhitungan di atas, diperoleh matriks untuk *cell state* seperti berikut:

$$h[1] = \begin{bmatrix} 0 \\ -117.00 \\ 0 \\ 0 \end{bmatrix}$$

5. Perhitungan *output gate* sesuai dengan Persamaan 2.13.

$$f_o[1] = \begin{bmatrix} -0.48530 & -0.042330 & -0.49050 & -0.22460 \\ 0.063690 & -0.23460 & 0.16280 & 0.53000 \\ 1.0220 & 0.74460 & 0.012930 & 0.40940 \\ -0.045900 & -0.087540 & 0.16220 & 1.2770 \end{bmatrix} \odot \begin{bmatrix} 58.426924 \\ 64.144482 \\ 64.144482 \\ 52.72933 \end{bmatrix} + \begin{bmatrix} 0.52350 & -0.69080 & 0.47550 & 0.31400 \\ 0.56460 & 0.63700 & 0.22930 & 0.15670 \\ 0.13610 & -0.35800 & -0.16660 & 0.16340 \\ 0.19950 & 0.66100 & -0.020360 & -0.38690 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dari perhitungan di atas, diperoleh matriks untuk *output gate* seperti berikut:

$$f_o[1] = f\left(\begin{bmatrix} -65.91403031 \\ 27.32478805 \\ 108.3239185 \\ 70.72052006 \end{bmatrix}\right)$$

Berikut adalah hasil perhitungan *output gate* yang telah diterapkan fungsi aktivasi ReLU:

$$f_o[1] = \begin{bmatrix} 0 \\ 27.32478805 \\ 108.3239185 \\ 70.72052006 \end{bmatrix}$$

Hasil dari perhitungan untuk *output gate* dan *cell state* akan digunakan untuk mendapatkan nilai keluaran sesuai dengan Persamaan 2.14.

$$y[1] = \begin{bmatrix} 0 \\ 27.32478805 \\ 108.3239185 \\ 70.72052006 \end{bmatrix} \odot g_2 \left(\begin{bmatrix} 0 \\ -117.00 \\ 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan di atas, diperoleh matriks untuk nilai keluaran seperti berikut:

$$y[1] = \begin{bmatrix} 0 \\ -54.65 \\ -216.65 \\ -141.44 \end{bmatrix}$$

3.4.3.3 Perhitungan *Dense*

Setelah dilakukan perhitungan pada lapisan LSTM, langkah berikutnya adalah melakukan perhitungan pada lapisan *dense* atau *fully connected layer*. Hal ini bertujuan untuk menghubungkan seluruh *neuron* yang ada untuk menghasilkan nilai keluaran.

Pada lapisan *dense*, akan dilakukan perhitungan sesuai dengan Persamaan 2.6. Ukuran *kernel* pada contoh perhitungan manual ini adalah 4×1 yang dihasilkan dari nilai acak terdistribusi normal sesuai dengan Persamaan 2.4. Contoh nilai *kernel* untuk lapisan *dense* dapat dilihat pada tabel 3.11 berikut. Contoh nilai

kernel yang digunakan adalah sebagai berikut.

Tabel 3.11 Contoh *kernel* pada lapisan *dense*

0.298860	0.22722	-0.869150	1.13740

Perhitungan pada lapisan ini dapat dilihat sebagai berikut. Hasil keluaran dari lapisan LSTM akan dilakukan perhitungan *dot product* dengan *kernel*. Fungsi aktivasi yang digunakan pada lapisan *dense* ini adalah ReLU seperti pada Bagian 2.1.3.1. Hasil dari lapisan *dense* berikut merupakan prediksi pemakaian obat di hari keenam berdasarkan data 5 hari yang dipelajari.

$$
\begin{aligned}
y &= f\left(W^T x + b\right) \\
&= f(\\
&\quad (0.298860 * 0) + (0.227220 * -54.65) + \\
&\quad (-0.869150 * -216.65) + (1.13740 * -141.44) \\
&\quad) \\
&= f(15.00695176) \\
&= 15.00695176
\end{aligned}
$$

3.4.4 *Root Mean Squared Error* (RMSE)

Bagian ini merupakan perhitungan *error* dari hasil prediksi model CNN-LSTM yang menggunakan ataupun tanpa lapisan *Max Pooling*. Perhitungan *error* yang digunakan adalah *Root Mean Squared Error* (RMSE) dengan cara menghitung rata-rata selisih antara hasil prediksi dengan data asli, dikuadratkan lalu diakarkan. Pada Bagian 3.4.1, data hari keenam adalah 363, maka hasil prediksi akan dibandingkan dengan nilai tersebut. Berikut ini adalah perhitungan RMSE untuk model *Nonpooling CNN-LSTM*.

$$
\begin{aligned}
RMSE_{Nonpool} &= \sqrt{\Sigma_{i=1}^{n}\left(\frac{y_i - \bar{y}_i}{n}\right)^2} \\
&= \sqrt{\left(\frac{363 - 91.5011695}{1}\right)^2} \\
&= 271.4988305
\end{aligned}
$$

Sedangkan perhitugan di bawah ini merupakan perhitungan RMSE untuk model CNN-LSTM.

$$
\begin{aligned}
RMSE_{CNN-LSTM} &= \sqrt{\Sigma_{i=1}^{n}\left(\frac{y_i - \bar{y}_i}{n}\right)^2} \\
&= \sqrt{\left(\frac{363 - 15.00695176}{1}\right)^2} \\
&= 347.99304824
\end{aligned}
$$