Jurnal KomtekInfo

https://jkomtekinfo.org/ojs

2022 Vol. 9 No. 2 Hal: 1-14 p-ISSN: 2356-0010, e-ISSN: 2502-8758

Optimalisasi Deep Learning Method Dengan Long Short-Term Memory Dalam Memproyeksi Curah Hujan

Edo Sulaiman[1], Sumijan[2], Musli Yanto[3]

[1,2,3] Teknik Informatika, Universitas Putra Indonesia "YPTK" Padang, Indonesia

edosulaii@gmail.com[1], sumijan@upiyptk.ac.id[2], musli_yanto@upiyptk.ac.id[3]

Abstract

Nowadays data is support for quick decision-making in various aspects including making predictions regarding accurate rainfall information where the commonly used modeling still has shortcomings such as the use of the number of parameters, mathematical assumptions, and the formulation of equations that tend to be complicated, therefore in the form a system to produce a prediction model that is close to optimal accuracy that is more efficient. deep learning can be applied to predict an event to make decisions such as predicting the rainfall of an area, one of which is the Pariaman Padang. One of the deep learning methods that are suitable for use on sequential data types is Long Short-Term Memory (LSTM). This study applies the deep learning LSTM method with 50 epochs 1 layer, the data used is 9:1, where 90% is training data and 10% is test data, the data range used in the calculation starts from October 16, 2004, to December 14, 2004, where 54 rows of data are used as training data, while the last 4 data lines are used as a comparison of the prediction results of the LSTM method, as well as the measurement of MSE values. The results showed that the MSE value from the evaluation of the model that was trained for 50 epochs got an MSE value of 0.03 for the prediction results of testing data for the next 4 days. The implementation of the LSTM method into the system makes it easier to make comparisons and future predictions compared to doing mathematical calculations manually, this convenience provides benefits so that the process of predicting rainfall in the Padang Pariaman area can be done more easily, quickly, and efficiently.

Keywords: *deep learning, BMKG, climatology, rainfall, long short-term memory*

Abstrak

Dewasa ini data merupakan penunjang pengambilan keputusan secara cepat dalam berbagai aspek termasuk melakukan prediksi mengenai informasi curah hujan yang akurat di mana pemodelan yang biasa di gunakan masih memiliki kekurangan seperti penggunaan jumlah parameter, asumsi matematis, dan rumusan persamaan yang cenderung rumit, maka dari itu di bentuk suatu sistem untuk menghasilkan sebuah model prediksi yang mendekati keakuratan optimal yang lebih efisien. deep learning dapat diterapkan untuk memprediksi suatu peristiwa untuk mengambil keputusan seperti memprediksi curah hujan suatu area salah satunya padang pariaman. Salah satu metode deep learning yang cocok digunakan pada tipe data sekuensial adalah Long Short-Term Memory (LSTM). Penelitian ini menerapkan deep learning metode LSTM dengan 50 epoch 1 layer, data yang di gunakan berbanding 9:1 dimana 90% sebagai data training dan 10% sebagai data uji, rentang data yang digunakan dalam perhitungan di mulai dari tanggal 16 Oktober 2004 sampai 14 Desember 2004 dimana 54 baris data digunakan sebagai data training, sedangkan data 4 baris terakhir digunakan sebagai perbandingan hasil prediksi metode LSTM, serta pengukuran nilai MSE. Hasil penelitian menunjukkan bahwa nilai MSE hasil evaluasi dari model yang di latih selama 50 epoch mendapat nilai MSE sebesar 0.03 untuk hasil prediksi data testing 4 hari kedepan. Implementasi metode LSTM ke dalam sistem mempermudah dalam melakukan perbandingan dan prediksi yang akan datang di bandingkan melakukan perhitungan matematis secara manual, kemudahan tersebut memberikan manfaat agar proses prediksi curah hujan daerah padang pariaman dapat di lakukan lebih mudah, cepat, dan efisien.

Kata kunci: *deep learning*, BMKG, klimatologi, curah hujan, *long short-term memory*.

KomtekInfo is licensed under a Creative Commons Attribution-Share Alike 4.0 International License.

1. Pendahuluan

Dewasa Dewasa ini data merupakan penunjang pengambilan keputusan secara cepat, atau dikenal dengan istilah Data Driven Decision Making (DDDM) di mana kemajuan teknologi berperan besar dalam memanfaatkan data dan informasi tersebut [1]. Dalam berbagai aspek termasuk melakukan prediksi mengenai informasi curah hujan yang akurat di mana pemodelan tersebut masih memiliki kekurangan seperti penggunaan jumlah parameter, asumsi matematis, dan rumusan persamaan yang cenderung rumit, untuk menghasilkan sebuah model prediksi yang mendekati keakuratan optimal harus memiliki banyak paremeter dan variabel input untuk memenuhi sebuah asumsi Prediksi [2].

Mengatasi perihal tersebut, dikembangkanlah sebuah Kecerdasan Buatan (Artificial Intelligence) yang memiliki kemampuan untuk melakukan pembelajaran untuk menganalisis berbagai macam asumsi dan aspek yang berpengaruh untuk menarik kesimpulan [2].

Machine Learning adalah bagian dari AI di mana mesin digunakan untuk belajar dari pengalaman masa lalu [1].

Diterima: xx-xx-20xx | Revisi: xx-xx-20xx | Diterbitkan: xx-xx-20xx | doi: 10.35134/komtekinfo.v9i2.1

Algoritma *Machine Learning* digunakan dalam berbagai aplikasi, seperti dalam kedokteran, pengenalan *email*, pengenalan suara, dan visi komputer, di mana sulit atau tidak mungkin untuk mengembangkan algoritma konvensional untuk melakukan tugas yang diperlukan [3].

Beberapa implementasi *Machine Learning* menggunakan data dan *Neural Network (Jaringan Saraf)* dengan cara yang meniru kerja otak biologis manusia [4]. *Deep Learning* dapat dipahami sebagai bentuk *Neural Network layer* berganda yang merupakan bagian dari *Machine Learning* yang dapat digunakan dalam tugas termasuk *computer vision, speech recognition, natural language processing, Machine translation, bioinformatics, drug design, medical image analysis, material inspection* dan *board game programs*, di mana mereka telah menghasilkan hasil yang sebanding dan dalam beberapa kasus melebihi kinerja para pakar [3].

Misalnya, sebuah komputasi yang menggunakan *Deep Learning*, mampu memahami konsep seperti garis, bentuk, tekstur, dan juga pengaruhnya dengan melihat data-data citra tanpa bantuan tambahan dari manusia [5]. *Machine Learning* senantiasa bekerja menggunakan 1 *layer* di mana *Deep Learning* bekerja lebih dari 1 *layer*. untuk batasan *layer* dari *Deep Learning* itu sendiri sebagai *Neural Network* biasanya memiliki 3 *layer* atau lebih, makin bayak *layer* yang digunakan akan memengaruhi lama waktu yang terpakai untuk komputer mengalkulasi [6].

Layer pada *Deep Learning* dapat di gambarkan seperti lapisan *neuron* pada otak manusia *layer* itu nantinya akan menggambarkan jarak atau *vektor* menggunakan Fungsi matematika yaitu fungsi *sigmoid (σ)* [7]. Alasan fungsi *sigmoid* digunakan karena dalam fungsi ini membutuhkan perhitungan yang relatif mudah dan cepat. Selain itu, fungsi *sigmoid* dapat diartikan sebagai nilai peluang karena nilainya antara 0 dan 1 [7].

Salah satu pendekatan *Deep Learning* yang mampu secara otomatis mempelajari fitur yang dideskripsikan dalam bentuk *vektor* adalah *Recurrent Neural Networks (RNN)* [8]. Pada *RNN* sendiri teknik *Learning* bekerja dengan menyimpan *layer* dari *output* kembali sebagai *input* pada *hidden layer* berikutnya hingga memprediksi hasil akhir [9]. Kelemahan *RNN* adalah tidak mampu lagi untuk belajar menghubungkan informasi ketika ada kesenjangan yang terus tumbuh, memori yang tersimpan akan semakin tidak relevan seiring waktu berjalan karena tertimpa dengan memori baru [10], di sebabkan kelemahan dari *RNN* sendiri tidak dapat mempelajari informasi yang terlalu jauh atau *Long-Term Dependencies*, yang cukup jauh pada masukannya [11].

Long Short-Term Memory (LSTM) merupakan sebuah pengembangan metode dari arsitektur *Recurrent Neural Network (RNN)*, Banyak peneliti yang mengembangkan metode *LSTM* di berbagai bidang seperti dalam bidang prediksi deret wakatu atau *forecasting* dikarenakan metode *LSTM* mampu mengatasi kekurangan tersebut karena metode ini dapat mengatur memori pada setiap masukannya dengan menggunakan *memory cells* dan *gate units* pada setiap *neuron*s yang berfungsi sebagai pengatur memori [10]. Contoh penggunaan *Deep Learning* untuk data *timeseries* yang banyak dihasilkan dari pengamatan cuaca adalah *LSTM*, *LSTM* sendiri diciptakan oleh *Hochreiter* dan *Schmidhuber* pada tahun 1997 [2].

Penelitian *LSTM* terdahulu yang juga di lakukan oleh Supriyadi, mengenai metode *Deep Learning LSTM* untuk memprediksi *parameter* cuaca, seperti suhu udara, kelembaban, kecepatan angin, dan tekanan udara. Metode ini bekerja dengan memanfaatkan fungsi matematika seperti fungsi *tanh* dan *sigmoid* yang berada dalam *layer LSTM*. Adapun jumlah *layer* yang digunakan sebanyak 200 buah. Sedangkan jumlah datanya dibagi dua menjadi *training* data dan test data dengan rasio 9:1. Pada bulan Januari 2019. Diperoleh *MSE parameter* suhu udara, kelembaban, kecepatan angin, dan tekanan udara nilainya semakin baik ketika menggunakan *Deep Learning LSTM* dengan update dibandingkan *LSTM* tanpa update. Diperoleh hasil prediksi suhu udara, kelembaban, kecepatan angin, dan tekanan udara 1 hari ke depan memiliki *MSE* yang baik. Dari *parameter* cuaca tersebut hanya *parameter* suhu dan kelembaban udara yang mengalami pertambahan *MSE* seiring bertambahnya waktu. Sedangkan *parameter* kecepatan angin dan tekanan udara mengalami penurunan di hari ketiga dan meningkat secara kontinu hingga 1 bulan ke depan [2].

Berdasarkan rincian penjelasan sebelumnya, sangat dimungkinkan untuk menggunakan *Deep Learning* dengan metode *LSTM* dikarenakan mendukung kegiatan proyeksi curah hujan. Karena data pengamatan meteorologi umumnya berupa *vektor* dan *timeseries*

Adapun Tujuan dari penelitian ini adalah :

1) Menerapkan Deep Learning menggunakan metode Long Short-Term Memory untuk melakukan proses prediksi curah hujan dalam menghasilkan alternatif dalam pengambilan sebuah keputusan.
2) Menerapkan pendekatan Deep Learning menggunakan metode Long Short-Term Memory untuk melakukan prediksi curah hujan.
3) Melakukan pengujian Deep Learning dengan metode Long Short-Term Memory di implementasikan ke dalam sebuah sistem yang dibangun untuk memprediksi curah hujan di daerah padang Pariaman.

2. Metodologi Penelitian

Metodologi Penelitian merupakan proses melakukan pendekatan terhadap objek penelitian. Permasalahan pada proyeksi curah hujan khusus daerah padang Pariaman masih belum pernah di lakukan sama sekali. Penelitian ini nantinya memerikan alternatif lain dalam melakukan prediksi curah hujan di daerah padang Pariaman dengan menggunakan metode Long Short-Term Memory.

2.1. Pengumpulan Data

Data dalam Penelitian ini data di dapat langsung dari *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman*, data yang di gunakan merupakan data harian dari tanggal *1 January 1985* sampai *31 Desember 2021* data tersebut terdiri dari beberapa fitur / *variabel* seperti yang terlihat pada **Error! Reference source not found.** berikut :

Tabel 1 Fitur Data Klimatologi BMKG

Kode	Keterangan	Satuan
Tn	Temperatur minimum	$°C$
Tx	Temperatur maksimum	$°C$
$Tavg$	Temperatur rata-rata	$°C$
RH_avg	Kelembapan rata-rata	%
RR	Curah hujan	mm
ss	Lamanya penyinaran matahari	jam
ff_x	Kecepatan angin maksimum	m/s
ddd_x	Arah angin saat kecepatan maksimum	°
ff_avg	Kecepatan angin rata-rata	m/s
ddd_car	Arah angin terbanyak	°

Sesuai Data Klimatologi yang di tampilkan pada **Error! Reference source not found.** di mana setiap fitur *variabel* pada data tersebut merupakan *variabel* independen. Tidak semua fitur data pada **Error! Reference source not found.** di gunakan pada Penelitian ini, dan peneliti hanya menyertakan fitur RR.

Data training digunakan untuk proses pelatihan model dengan metode LSTM sehingga terbentuk suatu model yang diuji performansinya terhadap data testing. Pembagian data yang digunakan yaitu 90% data training dan 10% data testing. Jumlah data training lebih besar dikarenakan agar mesin pembelajaran lebih terlatih untuk mempelajari model. Sehingga model yang dihasilkan dapat memberikan peramalan data testing yang lebih optimal.

2.2. *Preprocessing* Data

Dalam Penelitian ini di lakukan preprocessing data dengan data yang di gunakan memiliki beberapa nilai yang hilang missing values dan juga data bernilai NaN dari data yang di teliti. Nilai-nilai yang hilang / NaN tersebut ini muncul dari banyak faktor yang berada di luar kendali staf Stasiun Klimatologi Kelas II Sicincin Padang Pariaman.

2.2.1. Interpolate NaN

Interpolate NaN merupakan Teknik preprocessing data yang NaN atau missing values dengan dengan melakukan interpolasi linear pada data dengan nilai yang hilang / NaN, dan data tidak hilang melainkan data di isi dari nilai rentang nilai sebelum dan sesudahnya. Sebagai contoh peneliti mencoba menghitung missing value dari tanggal 14-06-2020 sebagai berikut.

$$\frac{y - y_a}{y_b - y_a} = \frac{x - x_a}{x_b - x_a}$$

$$x = x_a + (x_b - x_a)\frac{(y - y_a)}{(y_b - y_a)}$$

$$\frac{2 - 1}{5 - 1} = \frac{x - 2}{30 - 2}$$

$$x = 2 + (30 - 2)\frac{(2 - 1)}{(5 - 1)}$$

$$x = 2 + 28\frac{1}{4}$$

$$x = 9$$

Di mana [12] :

- y : Orde data yang akan di interpolasi
- y_a : Orde data sebelum data yang akan di interpolasi
- y_b : Orde data sesudah data yang akan di interpolasi
- x : Data hasil interpolasi
- x_a : Data orde sebelum data yang akan di interpolasi
- x_b : Data orde sesudah data yang akan di interpolasi

Hasil akhir terlihat seperti yang di jabarkan pada Tabel 2 dari tanggal *12-06-2020 - 17-06-2020* sebagai berikut.

Tabel 2 Data Klimatologi Sebelum dan Sesudah di Interpolate NaN

Tanggal	RR	Tanggal	RR
⋮	⋮	⋮	⋮
12-06-2020	0.0	12-06-2020	0.0
13-06-2020	2.0	13-06-2020	2.0
14-06-2020	NaN	14-06-2020	9.0
15-06-2020	NaN	15-06-2020	16.0
16-06-2020	NaN	16-06-2020	23.0
17-06-2020	30.0	17-06-2020	30.0
⋮	⋮	⋮	⋮

Seperti yang terlihat pada Tabel 2 baris data tanggal *14-06-2020 - 16-06-2020* dimana nilai *NaN* di gantikan dengan nilai rentang dari nilai sebelum dan sesudahnya berasarkan kedudukan nilai yang di interpolasi dan nilai rentang terdekat.

2.2.2. Normalisasi MinMaxScaller

Pada Sebelum data di processing ada baiknya data di normalisasi terlebih dahulu. di mana data yang di proses memiliki nilai rentang yang sama, tidak ada yang terlalu besar maupun terlalu kecil untuk setiap fitur data yang termasuk. Normalisasi data ini berguna agar proses analisis statistik pada data menjadi lebih mudah.

Normalisasi data yang di gunakan adalah normalisasi data *minmax* scaller, *Minmaxscaling* bekerja dengan scaling data dalam rentang tertentu (range nilai minimum hingga nilai maksimum), mengubah data berada pada rentang nilai 0 sampai 1. Sebagai contoh peneliti mencoba menormalisasi nilai dari tanggal *14-06-2020* sebagai berikut.

$$x_{norm} = \frac{(x - x_{min})}{(x_{max} - x_{min})}$$

$$x_{norm} = \frac{(9 - 0)}{(30 - 0)}$$

$$x_{norm} = 0.3$$

Di mana :

x_{norm} : Data hasil normalisasi
x : Data asli
x_{min} : Nilai minimum dari data *x*
x_{max} : Nilai maximum dari data *x*

Hasil akhir terlihat seperti yang di jabarkan pada Tabel 3 dari tanggal *12-06-2020* - *17-06-2020* sebagai berikut.

Tabel 3 Data Klimatologi Sebelum dan Sesudah di MinMaxScaller

Tanggal	RR	Tanggal	RR
⋮	⋮	⋮	⋮
12-06-2020	0.0	12-06-2020	0.0000
13-06-2020	2.0	13-06-2020	0.0667
14-06-2020	9.0	14-06-2020	0.3000
15-06-2020	16.0	15-06-2020	0.5333
16-06-2020	23.0	16-06-2020	0.7667
17-06-2020	30.0	17-06-2020	1.0000
⋮	⋮	⋮	⋮

Berdasarkan pada Tabel 3 semua nilai dari kolom RR di normalisasi dengan rentang nilai 0 – 1 berdasarkan nilai sebelumnya yang berkorelasi dengan nilai tertinggi dan terendah dari semua baris data yang di sertakan dalam perhitungan minmaxscaler

2.3. Perhitungan LSTM

Sebelum perancangan sistem dilakukan, peneliti melakukan analisis deskriptif untuk perancangan sistem dari Metode LSTM dengan perhitungan manual/numerik pada data Sebagian data yang di dapat langsung dari *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman.* Berikut contoh perhitungan manual jaringan LSTM

2.3.1. Inisialisasi Hyperparameter

Untuk melakukan perhitungan LSTM di perlukan menginisialisasi beberapa nilai hyperparameter yang perlu di tentukan sebelum tahap training data di lakukan seperti yang terlihat pada Tabel 4 berikut

Tabel 4. Nilai Inisialisasi Hyperparameter

Hyperparameter	**Nilai**
Jumlah *Epoch*	50
Ukuran *Batch* (n)	1
Panjang *Timestep/Sequence*	2
Features	rr
Jumlah *Units* LSTM	1
Jumlah *Hidden Layer*	1
Learning rate (γ)	0.1
Probabilitas *Dropout*	0

Berdasarkan fungsi optimizer untuk perbaharuan weight dan bias yang di gunakan adalah Stochastic Gradient Descent (SGD), maka perhabaruan weight dan bias di lakukan setiap proses timestep selesai, maka dari itu ukuran batch optimal adalah tidak lebih dari 1 seperti yang telihat pada Tabel 4 di atas.

2.3.2. Data Selection dan Preprocessing

Untuk data yang di gunakan sebanyak 60 baris data dengan *feature* Data Klimatologi yang di gunakan dalam perhitungan manual ini terbatas hanya menggunakan *feature (rr)* curah hujan, dimulai dari rentang waktu *16 Oktober 2004* sampai *14 Desember 2004*, *input* data *training* berbentuk tensor 3D di antaranya terdisi dari ukuran *batch* dimana hanya 1 *timestep*, jumlah *feature*s juga hanya 1 yaitu *feature (rr)* curah hujan, dan panjang *timestep*/*sequence* dengan panjang 2 baris data menjadi *input* x_1 dan x_2, seperti yang terlihat pada Tabel 5 berikut.

Tabel 5 Data yang di Gunakan Dalam Perhitungan Manual LSTM

No	**rr**	**(rr) Min MaxScale**	***Input***		**y**
			x_1	x_2	
1	0.5	0.0052	0.0052	0.0206	1.0000
2	2	0.0206	0.0206	1.0000	0.5258
3	97	1.0000	1.0000	0.5258	0.6206
4	51	0.5258	0.5258	0.6206	0.2371
5	60.2	0.6206	0.6206	0.2371	0.1031
6	23	0.2371	0.2371	0.1031	0.1753
7	10	0.1031	0.1031	0.1753	0.0206
8	17	0.1753	0.1753	0.0206	0.0010
9	2	0.0206	0.0206	0.0010	0.1546
10	0.1	0.0010	0.0010	0.1546	0.0897

11	15	0.1546	0.1546	0.0897	0.2773
12	8.7	0.0897	0.0897	0.2773	0.1526
13	26.9	0.2773	0.2773	0.1526	0.4402
14	14.8	0.1526	0.1526	0.4402	0.1392
15	42.7	0.4402	0.4402	0.1392	0.0670
16	13.5	0.1392	0.1392	0.0670	0.1753
17	6.5	0.0670	0.0670	0.1753	0.1897
18	17	0.1753	0.1753	0.1897	0.2598
19	18.4	0.1897	0.1897	0.2598	0.2959
20	25.2	0.2598	0.2598	0.2959	0.2577
21	28.7	0.2959	0.2959	0.2577	0.0103
22	25	0.2577	0.2577	0.0103	0.1619
23	1	0.0103	0.0103	0.1619	0.1959
24	15.7	0.1619	0.1619	0.1959	0.0412
25	19	0.1959	0.1959	0.0412	0.7021
26	4	0.0412	0.0412	0.7021	0.0639
27	68.1	0.7021	0.7021	0.0639	0.3093
28	6.2	0.0639	0.0639	0.3093	0.6495
29	30	0.3093	0.3093	0.6495	0.2268
30	63	0.6495	0.6495	0.2268	0.5495
31	22	0.2268	0.2268	0.5495	0.0320
32	53.3	0.5495	0.5495	0.0320	0.0062
33	3.1	0.0320	0.0320	0.0062	0.0031
34	0.6	0.0062	0.0062	0.0031	0.0258
35	0.3	0.0031	0.0031	0.0258	0.1299
36	2.5	0.0258	0.0258	0.1299	0.8557
37	12.6	0.1299	0.1299	0.8557	0.0103
38	83	0.8557	0.8557	0.0103	0.3093
39	1	0.0103	0.0103	0.3093	0.0144
40	30	0.3093	0.3093	0.0144	0.0010
41	1.4	0.0144	0.0144	0.0010	0.0474
42	0.1	0.0010	0.0010	0.0474	0.5320
43	4.6	0.0474	0.0474	0.5320	0.0515
44	51.6	0.5320	0.5320	0.0515	0.0021
45	5	0.0515	0.0515	0.0021	0.0938
46	0.2	0.0021	0.0021	0.0938	0.7938
47	9.1	0.0938	0.0938	0.7938	0.0165
48	77	0.7938	0.7938	0.0165	0.1278
49	1.6	0.0165	0.0165	0.1278	0.0000
50	12.4	0.1278	0.1278	0.0000	0.4753
51	0	0.0000	0.0000	0.4753	0.0206
52	46.1	0.4753	0.4753	0.0206	0.0000
53	2	0.0206			
54	0	0.0000			
55	14.5	0.1495	0.1495	0.0072	0.0082
56	0.7	0.0072	0.0072	0.0082	0.0000
57	0.8	0.0082	0.0082	0.0000	0.0175
58	0	0.0000	0.0000	0.0175	0.0124
59	1.7	0.0175			
60	1.2	0.0124			

Seperti yang terlihat pada Tabel 5 dari 60 baris data yang di gunakan, data di bagi menjadi skala 9:1 untuk data *training* dan data *testing*, dimana baris data 1-54 di gunakan untuk data *training* dan baris data 55-60 di gunakan untuk data *testing*. Nilai dari *input* (x) dan label (y) di dasarkan dengan bentuk pola data *sliding windows* dimana panjang *input* di dasarkan pada Panjang *timestep* dari inisialisasi nilai *hyperprameter* seperti yang terlihat pada Tabel 5 dari nilai yang di gunakan, untuk perhitungan manual ini peneliti hanya menggunakan 2 nilai *timestep* yang berarti bahwa hanya terdapat 2 *input* untuk model yang di rancang yaitu *input* x_1 dan x_2, dimana *input* x_1 merupakan nilai dari baris *feature* ke n, dan nilai *input* x_2 merupakan nilai dari baris $n+1$, dan untuk nilai dari *label* **y** merupakan nilai dari baris *feature* ke $n+2$ dan begitu seterusnya hingga batas dari data yang di gunakan dari masing-masing data *training* dan *testing*.

2.3.3. Inisialisasi Dimensi Data dan Parameter

Dimensi pada setiap *parameter* sangat di perlukan untuk pembangunan blok dasar dari jaringan LSTM termasuk jenis jaringan syaraf tiruan yang lainnya. Dimensi dari semua *parameter* dari LSTM tergantung pada dimensi *unit* tersesembunyi. Untuk penjelasan lebih lengkapnya dapat di lihat pada Tabel 6 berikut

Tabel 6. Dimensi Parameter Pada Jaringan LSTM

Parameter	**Keterangan**	**Dimensi**
$Units$	Jumlah *neuron unit* dalam *hidden layer*	(n_h)
$feature$	Jumlah *Feature*	(f)
n_x	Ukuran *input* (panjang *timestep*)	(n_x)
$C^{<t-1>}$	Dimensi *cell state* sebelumnya	(n_h, f)
$h^{<t-1>}$	Dimensi *output* sebelumnya	(n_h, f)
$x^{<t>}$	*Input* saat ini	(n_x, f)
$[x^{<t>}, h^{<t-1>}]$	Gabungan *output* sebelumnya dan *input* saat ini	$(n\_x + n\_h, f)$
W_f, W_i, W_c, W_o	*Weight* untuk semua *gate*	$(n\_h, n\_x + n\_h)$
b_f, b_i, b_c, b_o	*Bias* untuk semua operasi	$(n\_h, 1)$

Seperti yang telihat pada Tabel 6 pada kolom Dimensi, Hampir setiap dimensi dari *parameter* di LSTM memiliki hubungan langsung maupun tidak langsung dengan *unit* tersembunyi (n_h) pada *layer* LSTM, dan penting untuk di pahami bahwa perkalian matrik dari 2 metrik berukuran $(a, b) * (b, c)$maka menghasilkan *output* berukuran (a, c). Setelah menganalisa dimensi komponen, tahap selanjutnya adalah menganalisa

dimensi *output* pada setiap komponen. Untuk penjelasan lebih lengkapnya dapat di lihat pada Tabel 7 berikut.

Tabel 7. Dimensi Output Pada Jaringan LSTM

Output	Dimensi
f_t	$(n\_h, n\_x + n\_h) * (n\_x + n\_h, m) + (n\_h, 1) = (n\_h, m)$
i_t	$(n\_h, n\_x + n\_h) * (n\_x + n\_h, m) + (n\_h, 1) = (n\_h, m)$
$\tilde{C}_t$	$(n\_h, n\_x + n\_h) * (n\_x + n\_h, m) + (n\_h, 1) = (n\_h, m)$
o_t	$(n\_h, n\_x + n\_h) * (n\_x + n\_h, m) + (n\_h, 1) = (n\_h, m)$
C_t	$(n\_h, m) * (n\_h, m) + (n\_h, m) * (n\_h, m) = (n\_h, m)$
h_t	$(n\_h, m) * (n\_h, m) = (n\_h, m)$

Berdasarkan Tabel 7 fungsi aktivasi *sigmoid* dan *tanh* di tetapkan sebagai elemen dari matrix sehingga dimensi *input* dan *output* tidak berubah. Setelah melakukan *preprocessing* data dari normalisasi, menentukan nilai *hyperparameter* dan membagi *input* data sampai dengan pola *sliding windows* untuk model LSTM yang di gunakan. Tahap selanjutnya adalah menginisialisasi nilai awal dari *parameter forward* step seperti *bias* dan *weight*, *cell state* sebelumnya, dan *output* sebelumnya.

- Dimensi dari *Bias* seperti yang terlihat pada Tabel 7 berdimensi $(n\_h, 1)$, di karenakan jumlah *unit* yang di gunakan adalah 1 maka bentuk dimensi menjadi (1, 1), maka nilai *Bias* dari *forget, input, cell state, output* bisa di inisialisasikan dengan nilai nol seperti : [0], [0], [0], [0]
- Dimensi dari C_o seperti yang terlihat pada Tabel 7 berdimensi (n_h, f), Dengan jumlah *unit* yang di gunakan adalah 1 dan jumlah *feature* adalah 1 maka bentuk dimensi menjadi (1, 1), Untuk nilainya dikarenakan *Cell State* sebelumnya tidak ada maka nilai C_o di inisialisasi menjadi : [0]
- Dimensi dari h_0 seperti yang terlihat pada Tabel 7 berdimensi (n_h, f), Dengan jumlah *unit* yang di gunakan adalah 1 dan jumlah *feature* adalah 1 maka bentuk dimensi menjadi (1, 1), Untuk nilainya dikarenakan *Output* sebelumnya tidak ada maka nilai h_0 di inisialisasi menjadi : [0]
- Untuk nilai *Weight* seperti yang terlihat pada Tabel 7 berdimensi $(n\_h, n\_x + n\_h)$, Dengan jumlah *unit* yang di gunakan adalah 1, Panjang *timestep* adalah 2 maka bentuk dimensi menjadi (1, 2+1), maka nilai *Weight* untuk masing-masing *Weight forget, input, cell state, output* bisa di inisialisasikan dengan nilai acak atau dengan rumus *xavier Initialization* seperti berikut :

$$W = \left[\pm\frac{1}{\sqrt{(n\_x + n\_h)}}, \pm\frac{1}{\sqrt{(n\_x + n\_h)}}, \pm\frac{1}{\sqrt{(n\_x + n\_h)}}\right]$$

$$W_f = [0.5774\ 0.5774\ 0.5774]$$

$$W_i = [0.5774\ 0.5774\ 0.5774]$$

$$W_c = [0.5774\ 0.5774\ 0.5774]$$

$$W_o = [0.5774\ 0.5774\ 0.5774]$$

Untuk saat ini standar pendekatan terbaik untuk inisialisasi bobot untuk setiap lapisan jaringan syaraf tiruan yang menggunakan fungsi aktivasi *sigmoid* dan *tanh* adalah dengan menggunakan rumus *xavier Initialization*, berdasarkan riset ilmuan dari *Google DeepMind* pada tahun 2010 oleh [13].

Selanjutnya adalah menginisialisasi nilai awal dari *parameter backward* step seperti Δh_{n+1}, f_{n+1}, dan δC_{n+1}.

- Untuk Δh_{n+1} di inisialisasi menjadi 0 di karenakan tidak ada *timestep* selanjutnya.
- Untuk f_{n+1} sama seperti sebelumnya untuk nilai dari f_{n+1} di inisialisasi menjadi 0.
- Untuk δC_{n+1} sama seperti sebelumnya untuk nilai dari δC_{n+1} di inisialisasi menjadi 0.

2.4. Training Model LSTM

Setelah di ketahui nilai weight awal kemudian barulah di mulai proses training. Pada proses training algoritma pada model LSTM yang di gunakan forward step dan backward step. Berikut contoh perhitungan manual jaringan LSTM.

2.4.1. Perhitungan Epoch 1 Batch 1

Perhitungan di mulai dari epoch 1 dan batch 1 untuk satu kali iterasi yang di hitung secara numerik atau manual sebagai berikut.

2.4.1.1. Forward Timestep ke 1 Batch 1

Setelah menentukan hyperparameter, parameter/nilai awal, dan dimensi selanjutnya adalah menghitung forward step timestep 1 seperti berikut.

$$\begin{aligned} f_t &= \sigma\left(W_f * [x^{<t>} \quad h^{<t-1>}] + b_f\right) \\ &= \sigma((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1)) \\ &= \sigma([0.5774\ 0.5774\ 0.5774] \\ &\qquad * [0.0052 \quad 0.0206 \quad 0] + [0]) \\ &= \sigma(0.0149) \\ &= \frac{1}{1 + e^{-0.0149}} \\ &= 0.5037 \end{aligned}$$

$$i_t = \sigma(W_i * [x^{<t>} \quad h^{<t-1>}] + b_i)$$

$$= \sigma((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1))$$
$$= \sigma([0.5774 \quad 0.5774 \quad -0.5774] * [0.0052 \quad 0.0206 \quad 0] + [0])$$
$$= \sigma(0.0149)$$
$$= \frac{1}{1 + e^{-0.0149}}$$
$$= 0.5037$$

$$\tilde{C}_t = tanh(W_c * [x^{<t>} \quad h^{<t-1>}] + b_c)$$
$$= tanh((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1))$$
$$= tanh([0.5774 \quad 0.5774 \quad 0.5774] * [0.0052 \quad 0.0206 \quad 0] + [0])$$
$$= tanh(0.0149)$$
$$= 2 * \left(\frac{1}{1 + e^{-2*0.0149}}\right) - 1$$
$$= 0.0149$$

$$o_t = \sigma(W_o * [x^{<t>} \quad h^{<t-1>}] + b_o)$$
$$= \sigma((n\_h, n\_x + n\_h) * (n\_x + n\_h, f) + (n\_h, 1))$$
$$= \sigma([0.5774 \quad 0.5774 \quad 0.5774] * [0.0052 \quad 0.0206 \quad 0] + [0])$$
$$= \sigma(0.0149)$$
$$= \frac{1}{1 + e^{-0.0149}}$$
$$= 0.5037$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$= 0.5037 * 0 + 0.5037 * 0.0149$$
$$= 0.0075$$

$$h_t = o_t * tanh(C_t)$$
$$= 0.5037 * tanh(0.0075)$$
$$= 0.5037 * \left(2 * \left(\frac{1}{1 + e^{-2*0.0075}}\right) - 1\right)$$
$$= 0.0037$$

2.4.1.2. Backward Timestep ke 1 Batch 1

Dikarenakan ukuran *batch* tidak lebih dari 1 maka setiap forward step selesai dalam 1 *timestep* langsung di lakukan backward step.

$$\delta h_t = h_t - y^{<t>} + \Delta h_{t+1}$$
$$= 0.0037 - 1.0 + 0$$
$$= -0.9962$$

$$\delta C_t = \delta h_t * o_t * (1 - tanh(C_t)^2) + \delta C_{t+1} * f_{t+1}$$
$$= -0.9962 * 0.5037 * (1 - tanh(0.0075)^2) + 0 * 0$$
$$= -0.5018$$

$$\delta o_t = \delta h_t * tanh(C_t) * o_t * (1 - o_t)$$
$$= -0.9962 * tanh(0.0075) * 0.5037 * (1 - 0.5037)$$
$$= -0.0019$$

$$\widetilde{\delta C}_t = \delta C_t * i_t * \left(1 - \tilde{C}_t^{\,2}\right)$$
$$= -0.5018 * 0.5037 * (1 - 0.0149^2)$$
$$= -0.2527$$

$$\delta i_t = \delta C_t * \tilde{C}_t * i_t * (1 - i_t)$$
$$= -0.5018 * 0.0149 * 0.5037 * (1 - 0.5037)$$
$$= -0.0019$$

$$\delta f_t = \delta C_t * C_{t-1} * f_t * (1 - f_t)$$
$$= -0.5018 * 0 * 0.5037 * (1 - 0.5037)$$
$$= 0.0000$$

$$U_f = weight\ h\ dari\ W_f\ [x_1 \quad x_2 \quad h]$$
$$= [0.5774 \quad 0.5774 \quad 0.5774]$$
$$= 0.5774$$

$$U_i = weight\ h\ dari\ W_i\ [x_1 \quad x_2 \quad h]$$
$$= [0.5774 \quad 0.5774 \quad 0.5774]$$
$$= 0.5774$$

$$U_c = weight\ h\ dari\ W_c\ [x_1 \quad x_2 \quad h]$$
$$= [0.5774 \quad 0.5774 \quad 0.5774]$$
$$= 0.5774$$

$$U_o = weight\ h\ dari\ W_o\ [x_1 \quad x_2 \quad h]$$
$$= [0.5774 \quad 0.5774 \quad 0.5774]$$
$$= 0.5774$$

$$\Delta h_t = [U_f \quad U_i \quad U_c \quad U_o] * [\delta f_t \quad \delta i_t \quad \widetilde{\delta C}_t \quad \delta o_t]$$
$$= \begin{bmatrix} 0.5774 \\ 0.5774 \\ 0.5774 \\ 0.5774 \end{bmatrix} * \begin{bmatrix} 0.0000 \\ -0.0019 \\ -0.2527 \\ -0.0019 \end{bmatrix}$$
$$= -0.1481$$

2.4.1.3. Perbaharui Bias dan Weight Batch 1

Berdasarkan optimizer yang di gunakan adalah *Stochastic Gradient Descent* (SDG), maka perbaharuan *bias* dan *weight* di lakukan di setiap *timestep*.

$$W_f^{new} = W_f^{old} - \gamma * \sum_{t=1}^{n} [\delta f_t] * [x^{<t>} \quad h^{<t-1>}]$$

$$= W_f^{old} - 0.1 * ([0.0000] * [0.0052 \quad 0.0206 \quad 0.0000])$$
$$= [0.5774 \quad 0.5774 \quad 0.5774] - 0.1 * [0.0000 \quad 0.0000 \quad 0.0000]$$
$$= [0.5774 \quad 0.5774 \quad 0.5774]$$

$$b_f^{new} = b_f^{old} - \gamma * \sum_{t=1}^{n} [\delta f_t]$$
$$= 0 - 0.1 * ([0.0000])$$
$$= 0.0000$$

$$W_i^{new} = W_i^{old} - \gamma * \sum_{t=1}^{n} [\delta i_t] * [x^{<t>} \quad h^{<t-1>}]$$

$$= W_i^{old} - 0.1 * ([-0.0019] * [0.0052 \quad 0.0206 \quad 0.0000])$$
$$= [0.5774 \quad 0.5774 \quad 0.5774] - 0.1 * [-0.00001 \quad -0.00004 \quad 0.000]$$
$$= [0.577401 \quad 0.577404 \quad 0.5774]$$

$$b_i^{new} = b_i^{old} - \gamma * \sum_{t=1}^{n} [\delta i_t]$$
$$= 0 - 0.1 * ([-0.0019])$$
$$= 0.0002$$

$$W_c^{new} = W_c^{old} - \gamma * \sum_{t=1}^{n} [\widetilde{\delta C}_t] * [x^{<t>} \quad h^{<t-1>}]$$

$$= W_c^{old} - 0.1 * ([-0.2527] * [0.0052 \quad 0.0206 \quad 0.0000])$$
$$= [0.5774 \quad 0.5774 \quad 0.5774] - 0.1 * [-0.0013 \quad -0.0052 \quad 0.0000]$$
$$= [0.57753 \quad 0.57792 \quad 0.5774]$$

$$b_c^{new} = b_c^{old} - \gamma * \sum_{t=1}^{n} [\widetilde{\delta C}_t]$$
$$= 0 - 0.1 * ([-0.2527])$$
$$= 0.0253$$

$$W_o^{new} = W_o^{old} - \gamma * \sum_{t=1}^{n} [\delta o_t] * [x^{<t>} \quad h^{<t-1>}]$$

$$= W_o^{old} - 0.1 * ([-0.0019] * [0.0052 \quad 0.0206 \quad 0.0000])$$
$$= [0.5774 \quad 0.5774 \quad 0.5774] - 0.1 * [-0.00001 \quad -0.00004 \quad 0.000]$$
$$= [0.577401 \quad 0.577404 \quad 0.5774]$$

$$b_o^{new} = b_o^{old} - \gamma * \sum_{t=1}^{n} [\delta o_t]$$
$$= 0 - 0.1 * ([-0.0019])$$
$$= 0.0002$$

2.4.1.4. Perbaharui Bias dan Weight Batch 1

Nilai Akurasi di hitung menggunakan rumus MSE dari perhitungan kereluruhan timeteps pada *batch* 1 seperti berikut.

$$MSE = \frac{\sum_{t=1}^{n}(y_t - h_t)^2}{n}$$
$$= \frac{((1.0000 - 0.0037)^2)}{1}$$
$$= 0.992463094$$

Berdasarkan hasil nilai *error* dari data *batch* 1 di dapat nilai *error* sebesar 0.992463094. di mana nilai *error* 0.992463094 memiliki arti bahwa prediksi *feature (rr)* memiliki tingkat ketidakakuratan lebih kurang sebesar 0.992463094dalam memprediksi curah hujan dari data curah hujan yang di normalisasi minmaxscaller

2.4.2. Perhitungan Epoch 1 Batch 2-52

Perhitungan *batch* 2-52 sama seperti sebelumnya, hanya saja beberapa *parameter* seperti *bias* dan *weight* di perbaharui setelah menghitung *Stochastic Gradient Descent* (SGD) dari perhitungan *batch* sebelumnya.

2.4.2.1. Forward Epoch 1 Batch 2-52

Forward batch 2-52 berikut hasil perhitungan *forward* pada *batch* 2-52 seperti yang terlihat pada Tabel 8 berikut**.**

Tabel 8. Dimensi Output Pada Jaringan LSTM

batch	***Forward***					
	Gate				C_t	h_t
	f_t	i_t	$\widetilde{C}_t$	o_t		
2	0.6432	0.6432	0.5477	0.6432	0.3523	0.2177
3	0.7070	0.7078	0.7254	0.7078	0.5134	0.3345
4	0.6597	0.6615	0.6108	0.6618	0.4041	0.2537
5	0.6213	0.6230	0.4924	0.6232	0.3068	0.1854
6	0.5490	0.5502	0.2301	0.5503	0.1266	0.0693
7	0.5401	0.5414	0.1992	0.5416	0.1079	0.0582
8	0.5282	0.5294	0.1508	0.5296	0.0798	0.0422
9	0.5031	0.5042	0.0492	0.5043	0.0248	0.0125
10	0.5225	0.5237	0.1310	0.5238	0.0686	0.0359
11	0.5352	0.5365	0.1824	0.5366	0.0978	0.0523
12	0.5528	0.5543	0.2575	0.5545	0.1427	0.0786
13	0.5617	0.5633	0.2928	0.5635	0.1649	0.0921
14	0.5847	0.5868	0.3879	0.5870	0.2276	0.1336
15	0.5829	0.5849	0.3801	0.5851	0.2223	0.1300
16	0.5297	0.5314	0.1775	0.5316	0.0943	0.0501
17	0.5349	0.5368	0.2017	0.5369	0.1083	0.0579
18	0.5525	0.5545	0.2731	0.5547	0.1514	0.0839
19	0.5645	0.5667	0.3234	0.5669	0.1833	0.1028
20	0.5795	0.5820	0.3834	0.5823	0.2231	0.1292
21	0.5792	0.5819	0.3856	0.5821	0.2244	0.1299
22	0.5386	0.5409	0.2294	0.5411	0.1241	0.0668
23	0.5248	0.5272	0.1788	0.5274	0.0943	0.0496
24	0.5515	0.5540	0.2860	0.5542	0.1584	0.0874
25	0.5342	0.5366	0.2174	0.5368	0.1166	0.0623
26	0.6057	0.6090	0.4936	0.6092	0.3006	0.1878
27	0.6088	0.6118	0.4995	0.6120	0.3056	0.1914

28	0.5537	0.5567	0.3098	0.5569	0.1725	0.0951
29	0.6350	0.6389	0.5956	0.6392	0.3805	0.2321
30	0.6239	0.6278	0.5619	0.6280	0.3527	0.2128
31	0.6102	0.6149	0.5281	0.6152	0.3247	0.1930
32	0.5832	0.5874	0.4351	0.5877	0.2555	0.1470
33	0.5055	0.5090	0.1361	0.5092	0.0693	0.0352
34	0.5013	0.5047	0.1184	0.5049	0.0598	0.0301
35	0.5042	0.5076	0.1298	0.5078	0.0659	0.0334
36	0.5225	0.5261	0.2058	0.5263	0.1082	0.0567
37	0.6385	0.6432	0.6218	0.6435	0.3999	0.2445
38	0.6225	0.6267	0.5706	0.6269	0.3576	0.2151
39	0.5460	0.5499	0.3134	0.5501	0.1723	0.0939
40	0.5466	0.5506	0.3144	0.5508	0.1731	0.0944
41	0.5022	0.5058	0.1376	0.5059	0.0796	0.0351
42	0.5070	0.5105	0.1570	0.5107	0.0802	0.0409
43	0.5829	0.5869	0.4520	0.5871	0.2653	0.1522
44	0.5834	0.5876	0.4526	0.5878	0.2660	0.1528
45	0.5077	0.5112	0.1659	0.5113	0.0848	0.0433
46	0.5138	0.5173	0.1914	0.5175	0.0990	0.0511
47	0.6254	0.6296	0.5946	0.6298	0.3744	0.2253
48	0.6149	0.6190	0.5604	0.6192	0.3469	0.2066
49	0.5208	0.5242	0.2305	0.5243	0.1208	0.0630
50	0.5184	0.5218	0.2199	0.5220	0.1147	0.0596
51	0.5682	0.5718	0.4133	0.5719	0.2364	0.1327
52	0.5711	0.5749	0.4223	0.5751	0.2428	0.1369

Pada Tabel 8 memperlihatkan hasil forward step LSTM pada *epoch* 1 dari setiap *timestep* di dalam *batch* 2-52 dengan ukuran *batch* tidak lebih dari 1 *timestep*.

2.4.2.2. Backward Epoch 1 Batch 2-52

Backward batch 2-52 berikut hasil perhitungan *backward* pada *batch* 2-52 seperti yang terlihat pada Tabel 9 berikut**.**

Tabel 9. Dimensi Output Pada Jaringan LSTM

batch	***Backward***						
			Gate				
	δh_t	δC_t	δo_t	$\widetilde{\delta C}_t$	δi_t	δf_t	Δh_t
2	-0.3081	-0.1755	-0.0239	-0.0790	-0.0221	0.0000	-0.0722
3	-0.2861	-0.1573	-0.0280	-0.0528	-0.0236	0.0000	-0.0602
4	0.0166	0.0094	0.0014	0.0039	0.0013	0.0000	0.0038
5	0.0823	0.0467	0.0057	0.0221	0.0054	0.0000	0.0192
6	-0.1060	-0.0574	-0.0033	-0.0299	-0.0033	0.0000	-0.0211
7	0.0376	0.0201	0.0010	0.0105	0.0010	0.0000	0.0072
8	0.0411	0.0217	0.0008	0.0112	0.0008	0.0000	0.0074
9	-0.1421	-0.0716	-0.0009	-0.0360	-0.0009	0.0000	-0.0218
10	-0.0538	-0.0281	-0.0009	-0.0144	-0.0009	0.0000	-0.0094
11	-0.2250	-0.1196	-0.0055	-0.0620	-0.0054	0.0000	-0.0421
12	-0.0740	-0.0402	-0.0026	-0.0208	-0.0026	0.0000	-0.0150
13	-0.3481	-0.1909	-0.0140	-0.0983	-0.0137	0.0000	-0.0728
14	-0.0078	-0.0044	-0.0004	-0.0022	-0.0004	0.0000	-0.0017
15	0.0609	0.0340	0.0032	0.0170	0.0031	0.0000	0.0135
16	-0.1253	-0.0660	-0.0029	-0.0340	-0.0029	0.0000	-0.0230
17	-0.1318	-0.0699	-0.0035	-0.0360	-0.0035	0.0000	-0.0249
18	-0.1764	-0.0957	-0.0065	-0.0491	-0.0065	0.0000	-0.0358
19	-0.1931	-0.1059	-0.0086	-0.0537	-0.0084	0.0000	-0.0408

20	-0.1299	-0.0720	-0.0069	-0.0358	-0.0067	0.0000	-0.0285
21	0.1182	0.0654	0.0063	0.0324	0.0061	0.0000	0.0259
22	-0.0951	-0.0507	-0.0029	-0.0260	-0.0029	0.0000	-0.0183
23	-0.1463	-0.0765	-0.0034	-0.0390	-0.0034	0.0000	-0.0265
24	0.0458	0.0248	0.0018	0.0126	0.0018	0.0000	0.0093
25	-0.6397	-0.3388	-0.0185	-0.1732	-0.0183	0.0000	-0.1212
26	0.1139	0.0635	0.0079	0.0292	0.0075	0.0000	0.0258
27	-0.1279	-0.0714	-0.0090	-0.0328	-0.0085	0.0000	-0.0290
28	-0.5544	-0.2997	-0.0234	-0.1508	-0.0229	0.0000	-0.1138
29	0.0053	0.0030	0.0004	0.0012	0.0004	0.0000	0.0012
30	-0.3367	-0.1872	-0.0267	-0.0804	-0.0246	0.0000	-0.0760
31	0.1610	0.0893	0.0120	0.0396	0.0112	0.0000	0.0362
32	0.1408	0.0776	0.0085	0.0369	0.0082	0.0000	0.0310
33	0.0321	0.0163	0.0006	0.0081	0.0006	0.0000	0.0053
34	0.0044	0.0022	0.0001	0.0011	0.0001	0.0000	0.0007
35	-0.0965	-0.0488	-0.0016	-0.0243	-0.0016	0.0000	-0.0159
36	-0.7989	-0.4156	-0.0215	-0.2094	-0.0213	0.0000	-0.1456
37	0.2342	0.1289	0.0204	0.0509	0.0184	0.0000	0.0518
38	-0.0942	-0.0521	-0.0076	-0.0220	-0.0070	0.0000	-0.0211
39	0.0794	0.0424	0.0034	0.0210	0.0033	0.0000	0.0160
40	0.0934	0.0499	0.0040	0.0248	0.0039	0.0000	0.0188
41	-0.0123	-0.0062	-0.0002	-0.0031	-0.0002	0.0000	-0.0020
42	-0.4911	-0.2492	-0.0098	-0.1241	-0.0098	0.0000	-0.0830
43	0.1007	0.0551	0.0063	0.0257	0.0060	0.0000	0.0220
44	0.1507	0.0826	0.0095	0.0386	0.0091	0.0000	0.0330
45	-0.0506	-0.0257	-0.0011	-0.0128	-0.0011	0.0000	-0.0086
46	-0.7428	-0.3806	-0.0183	-0.1897	-0.0182	0.0000	-0.1306
47	0.2089	0.1147	0.0174	0.0467	0.0159	0.0000	0.0462
48	0.0787	0.0433	0.0062	0.0184	0.0057	0.0000	0.0175
49	0.0630	0.0326	0.0019	0.0162	0.0019	0.0000	0.0115
50	-0.4156	-0.2141	-0.0118	-0.1063	-0.0117	0.0000	-0.0750
51	0.1121	0.0607	0.0064	0.0288	0.0061	0.0000	0.0238
52	0.1369	0.0743	0.0080	0.0351	0.0077	0.0000	0.0293

Pada Tabel 9 memperlihatkan hasil backward step LSTM pada *epoch* 1 dari setiap *timestep* di dalam *batch* 2-52 dengan ukuran *batch* tidak lebih dari 1 *timestep*.

3. Hasil dan Pembahasan

Hasil pembahasan peneliti menguji dengan mejabarkan Hasil Proyeksi dan nilai error dari perhitungan LSTM yang telah di di lakukan pada tahap sebelumnya.

3.1. Hasil MSE Epoch 1 Batch 1-52

Nilai *Error* Hasil Training Epoch 1 (satu) di hitung menggunakan rumus MSE dari perhitungan kereluruhan timeteps pada *batch* 1-52 dan hasilnya dapat di lihat pada Tabel 10 berikut.

Tabel 10. Nilai Error Epoch 1 Batch 1-52

Batch	y_t	h_t	**MSE**
1	1.000000	0.003776	0.992463
2	0.525773	0.217691	0.094914
3	0.620619	0.334509	0.081859
4	0.237113	0.253746	0.000277
5	0.103093	0.185386	0.006772
6	0.175258	0.069303	0.011226
7	0.020619	0.058186	0.001411
8	0.001031	0.042180	0.001693
9	0.154639	0.012518	0.020198
10	0.089691	0.035888	0.002895
11	0.277320	0.052340	0.050616
12	0.152577	0.078612	0.005471
13	0.440206	0.092096	0.121181
14	0.139175	0.131349	0.000061
15	0.067010	0.127959	0.003715
16	0.175258	0.049988	0.015693

17	0.189691	0.057914	0.017365
18	0.259794	0.083358	0.031130
19	0.295876	0.102768	0.037291
20	0.257732	0.127808	0.016880
21	0.010309	0.128495	0.013968
22	0.161856	0.066793	0.009037
23	0.195876	0.049575	0.021404
24	0.041237	0.087081	0.002102
25	0.702062	0.062329	0.409258
26	0.063918	0.177787	0.012966
27	0.309278	0.181410	0.016350
28	0.649485	0.095103	0.307339
29	0.226804	0.232145	0.000029
30	0.549485	0.212775	0.113373
31	0.031959	0.192999	0.025934
32	0.006186	0.146992	0.019826
33	0.003093	0.035225	0.001033
34	0.025773	0.030139	0.000019
35	0.129897	0.033411	0.009310
36	0.855670	0.056741	0.638287
37	0.010309	0.244476	0.054834
38	0.309278	0.215074	0.008875
39	0.014433	0.093857	0.006308
40	0.001031	0.094395	0.008717
41	0.047423	0.035144	0.000151
42	0.531959	0.040860	0.241178
43	0.051546	0.152215	0.010134
44	0.002062	0.152757	0.022709
45	0.093814	0.043252	0.002557
46	0.793814	0.051054	0.551693
47	0.016495	0.225346	0.043619
48	0.127835	0.206567	0.006199
49	0.000000	0.063048	0.003975
50	0.475258	0.059627	0.172749
51	0.020619	0.132724	0.012568
52	0.000000	0.136923	0.018748

Berdasarkan Tabel 10 hasil nilai *error* MSE dari *epoch* 1 di dapat nilai *error* seperti yang terlihat pada kolom MSE memiliki arti bahwa prediksi *feature (rr)* setiap *batch* memiliki tingkat ketidakakuratan lebih kurang seperti yang terlihat pada kolom tersebut dalam memprediksi curah hujan dari data curah hujan yang di normalisasi minmaxscaller.

3.2. Hasil MSE Epoch 1-50

Nilai Error Hasil perhitungan *Epoch* 1-50 nilai *error* di hitung menggunakan total dari rumus MSE seperti pada Tabel 11 berikut.

Tabel 11. Nilai Error Epoch 1-50

Epoch	$\sum MSE$
1	0.082276
2	0.075683
3	0.072499
4	0.070754
5	0.069697
6	0.069002
7	0.068510
8	0.068140
9	0.067844
10	0.067598
11	0.067386
12	0.067198
13	0.067029
14	0.066873
15	0.066729
16	0.066594
17	0.066467
18	0.066348
19	0.066235
20	0.066127
21	0.066025
22	0.065927
23	0.065834
24	0.065745
25	0.065660
26	0.065578
27	0.065500
28	0.065425
29	0.065352
30	0.065283
31	0.065216
32	0.065151
33	0.065089
34	0.065029
35	0.064971
36	0.064915
37	0.064861
38	0.064808
39	0.064758
40	0.064709
41	0.064661
42	0.064615
43	0.064570
44	0.064527
45	0.064485
46	0.064444
47	0.064404
48	0.064366
49	0.064328
50	0.064291

Berdasarkan Tabel 11 hasil nilai *error* MSE dari *epoch* 2-52 di dapat nilai *error* seperti yang terlihat pada kolom $\sum MSE$, dimana nilai *error* selalu berkurang seiring dengan perulangan *epoch* selama 50 kali. memiliki arti bahwa prediksi *feature* (rr) setiap perulangan *epoch* yang di lakukan memiliki tingkat ketidakakuratan terus berkurang seperti yang terlihat pada kolom tersebut dalam memprediksi curah hujan dari data curah hujan yang di normalisasi minmaxscaller.

3.2. Weight dan Bias Hasil Training Model

Setelah melakukan *Training* Model LSTM maka di ketahui nilai *bias* dan *weight* optimal dari 50 *epoch* seperti pada Tabel 12 berikut.

Tabel 12. Bias dan Weight dari Learned Model

Parameters	***Forget Gate***	***Input Gate***	***Memory Gate***	***Output Gate***
bias	[0.0000]	[0.0257]	[0.8835]	[−0.0236]
weight	[0.5774 0.5774 0.5774]	[0.4437 0.3524 0.5774]	[0.3817 0.3405 0.5774]	[0.4342 0.3135 0.5574]

Hasil optimal dari *bias* dan *weight* dari learned model di gunakan untuk melakukan pengujian dari data *testing*. Pada proses *testing* algoritma pada model LSTM yang di gunakan hanya *forward* step sebanyak data yang di *testing* seperti yang terlihat pada Tabel 13**.**

Tabel 13. Hasil Testing Model

t	y_t	h_t	MSE
1	0.0082	0.1878	0.032252586
2	0.0000	0.1716	0.029432851
3	0.0175	0.1709	0.023530073
4	0.0124	0.1716	0.025350684

Pada Tabel 13 untuk nilai *error* sama seperti pada tahap *training*, peneliti hanya menggunakan rumus MSE untuk perhitungan nilai *error*.

$$\sum MSE = \frac{0.032252586 + 0.029432851 + 0.023530073 + 0.025350684}{4}$$
$$= 0.027641549$$

Berdasarkan hasil nilai *error* MSE dari proses *testing* di dapat nilai *error* sebesar 0.027641549. Dimana nilai *error* 0.027641549memiliki arti bahwa prediksi *feature (rr)* pada saat *testing* memiliki tingkat ketidakakuratan lebih kurang sebesar 0.027641549 dalam memprediksi *feature/variabel (rr)* curah hujan. Dapat di katakan bahwa Proyeksi atau perhitungan manual LSTM yang di lakukan peneliti dengan hanya menggunakan 1 *feature/variabel* yaitu *(rr)* curah hujan, dengan beberapa nilai *hyperparameter* lainnya seperti yang terlihat pada Tabel 4, dan menggunakan *deep learning* dengan metode *Long Short-Term Memory* dengan algoritma *backpropagation* masih jauh dari nilai keakuratan yang di harapkan.

3.3. Konversi Hasil Prediksi Dalam Nilai Nyata

Untuk tingkatan curah hujan memiliki nilai rentang berdasarkan tingkatan jumlah curah hujan dalam satuan milimeter, Satu milimeter hujan berarti air hujan yang turun di wilayah seluas satu meter persegi akan memiliki ketinggian satu milimeter jika air hujan tidak meresap, mengalir, atau menguap, berdasarkan nilai akumulasi yang di dapatkan peneliti pada *Stasiun Klimatologi Kelas II Sicincin Padang Pariaman.* Ambang batas nilai yang digunakan untuk menentukan intensitas hujan terlihat seperti pada Tabel 14 berikut.

Tabel 14. Rentang Nilai Curah Hujan

Nilai	**Keterangan**	**Satuan**
0	Berawan	$mm/hari$
$0.5 - 20$	Hujan ringan	$mm/hari$
$20 - 50$	Hujan sedang	$mm/hari$
$50 - 100$	Hujan lebat	$mm/hari$
$100 - 150$	Hujan sangat lebat	$mm/hari$
> 150	Hujan ekstrem	$mm/hari$

Seperti yang terlihat pada Tabel 14 nilai curah hujan di ukur berdasarkan tingkatan curah hujan dalam 1 *milimeter*, maka dari itu perlu di lakukan konversi nilai hasil Prediksi ke nilai nyata agar nilai Prediksi tersebut dapat di implementasikan untuk Prediksi curah hujan dalam kehidupan nyata.

Berdasarkan output hasil prediksi masih berbentuk nilai normalisasi *minmaxscaller*, maka dari itu perlu di lakukan konversi nilai hasil Prediksi dari hasil Prediksi Perhitungan Sistem yang di rancang ke nilai nyata dengan di lakukan denormalisasi nilai, seperti yang terlihat pada Tabel 15 berikut.

Tabel 15. Denormalisasi Nilai Hasil Prediksi Perhitungan Manual

Hari Ke	**Prediksi**	**Denormalisasi Prediksi**	**Nilai Real Hasil Prediksi**
1	0.1878	18.2	Hujan ringan
2	0.1716	16.6	Hujan ringan
3	0.1709	16.6	Hujan ringan
4	0.1716	16.6	Hujan ringan

Berdasarkan Tabel 15 hasil denormalisasi hari ke 1 untuk Perhitugan Manual dari nilai 0.1878 di dapat nilai real sebesar 18.2 yang memiliki arti bahwa Prediksi feature *(rr)* curah hujan untuk hari ke 1 berdasarkan perhitungan manual di perkirakan dalam rentang nilai 0.5 – 20 mm/hari, dapat di katakan prediksi curah hujan untuk hari 1 berdasarkan nilai rentang pada Tabel 14 dapat di katakan akan terjadi Hujan ringan.

4. Kesimpulan

Berdasarkan pehitungan lstm yang di lakukan, yang telah diurai pada penjelasan sebelumnya, dengan melakukan penelitian dan penganalisaan dengan menggunakan metode-metode penelitian yang di butuhkan maka dapat diambil kesimpulan sebagai berikut.

1) Proses prediksi curah hujan dengan menggunakan pendekatan *Deep Learning* menggunakan metode

Long Short-Term Memory dapat menghasilkan alternatif dalam pengambilan keputusan, dimana dalam pengambilan keputusan metode *Long Short-Term Memory* memberikan ketepatan hasil prediksi sesuai nilai *error* dari hasil evaluasi, yang mana besar dari nilai *error mean squared error* prediksi tersebut bisa di gunakan sebagai alternatif pengambil keputusan, dalam memprediksi curah hujan.

2) Menerapkan *Deep Learning* menggunakan metode *Long Short-Term Memory* dapat melakukan prediksi curah hujan dengan menggunakan data pada masa lampau, dimana data curah hujan pada masa lampau di gunakan untuk melatih model *Long Short-Term Memory* sehingga model tersebut dapat memberika pola gambaran data selanjutnya untuk prediksi curah hujan di masa depan.
3) Pengujian *Deep Learning* menggunakan metode *Long Short-Term Memory* di implementasikan ke dalam sebuah sistem yang di bangun untuk memprediksi curah hujan agar proses pengujian dapat di lakukan lebih praktis dan efisien dimana pengaturan nilai-nilai *hyperparameter* dan jangka waktu Prediksi curah hujan dapat di lakukan dengan lebih mudah sesuai kebutuhan, Riwayat Proyeksi juga di simpan ke dalam *database* setiap melakukan proyeksi, sehinggga dapat menjadi perbandingan dalam melakukan pengujian di masa depan.

Saran. Berdasarkan kesimpulan yang telah diuraikan sebelumnya menggunakan metode LSTM untuk Proyeksi curah hujan, maka peneliti menyampaikan baeberapa saran yang diharapkan menjadi bahan pertimbangan dengan harapan agar berguna bagi penelitian ini dan penyempurnaan penelitian selanjutnya.

1) Penelitian ini masih dapat di kembangkan dengan menggunakan fungsi optimasi yang lebih efektif untuk penanganan perubahan nilai *Learning* rate sejalan dengan perubahan nilai *MSE* setiap iterasi sehingga dapat memperoleh hasil prediksi yang lebih akurat.
2) Menambahkan metode Proyeksi lain yang seperti metode *ARIMA* maupun metode *Deep Learning* seperti *Convolution Neural Network (CNN)* untuk meningkatkan peforma dalam memproyeksi curah hujan di masa depan.

Daftar Rujukan

[1] M. A. Aditya, R. D. Mulyana, I. P. Eka dan S. R. Widianto, "Penggabungan Teknologi Untuk Analisa Data Berbasis Data Science," *Seminar Nasional Teknologi Komputer & Sains (SAINTEKS)*, pp. 51-56, February 2020.

[2] E. Supriyadi, "PREDIKSI PARAMETER CUACA MENGGUNAKAN DEEP LEARNING LONG-SHORT TERM MEMORY (LSTM)," *JURNAL METEOROLOGI DAN GEOFISIKA*, vol. 21, pp. 55-67, 16 Oktober 2019.

[3] J. Hu, H. Niu, J. Carrasco, B. Lennox dan F. Arvin, "Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14413-14423, 29 October 2020.

[4] V. Zhou, "Machine Learning for Beginners: An Introduction to Neural Networks," 6 March 2019. [Online]. Available: https://towardsdatascience.com/machine-learning-for-beginners-an-introduction-to-neural-networks-d49f22d238f9.

[5] A. Puspaningrum, M. S. Bunga dan I. , "Klasifikasi Perubahan Perangkat Lunak pada Mobile App Review dengan Menggunakan Metode Long Short Term Memory (LSTM)," *Jurnal IKRA-ITH Informatika*, vol. 3, pp. 41-46, November 2020.

[6] E. D. Tarkus, S. R. Sompie dan A. Jacobus, "Implementasi Metode Recurrent Neural Network pada Pengklasifikasian Kualitas Telur Puyuh," *Jurnal Teknik Informatika*, vol. 15, pp. 137-144, 30 Juni 2020.

[7] M. R. S. Putra, A. B. Osmond dan A. S. R. Ansori, "ESTIMASI HARGA KEBUTUHAN POKOK DI KOTA BANDUNG DAN PROVINSI JAWA BARAT MENGGUNAKAN METODE LSTM," *e-Proceeding of Engineering*, vol. 7, pp. 1455-1459, 1 April 2020.

[8] Y. Wibisono dan M. L. Khodra, "Pengenalan Entitas Bernama Otomatis untuk Bahasa Indonesia dengan Pendekatan Pembelajaran Mesin," 09 April 2018.

[9] H. Schneiderman dan T. Kanade, "A statistical method for 3D object detection applied to faces and cars," *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 06 August 2002.

[10] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath dan B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Megazine*, pp. 82-97, 18 October 2012.

[11] J. W. G. Putra, Pengenalan Konsep Pembelajaran Mesin dan Deep Learning, Tokyo, 2020.

[12] I. H. Al Amin, V. Lusiana dan B. Hartono, "PENCARIAN LINTASAN PADA COLLISION DETECTION MENGGUNAKAN PENDEKATAN INTERPOLASI LINIER," *Prosiding SINTAK*, vol. 2, pp. 57-61, 14 November 2018.

[13] X. Glorot dan Y. Bagio, "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.