

**KLASIFIKASI SUASANA PADA CERITA PENDEK BAHASA  
INDONESIA MENGGUNAKAN LSTM DAN BI-LSTM  
DENGAN WORD EMBEDDING FASTTEXT**

**SKRIPSI**

Diajukan untuk Memenuhi Salah Satu Syarat Memperoleh Gelar Sarjana  
Pendidikan Program Studi Informatika



Oleh:

Dylene Melynea Fernandez

NIM: 185314125

**PROGRAM STUDI INFORMATIKA  
FAKULTAS SAINS DAN TEKNOLOGI  
UNIVERSITAS SANATA DHARMA  
YOGYAKARTA  
2022**

**CLASSIFICATION OF INDONESIAN SHORT STORIES  
ATMOSPHERE USING LSTM AND BI-LSTM WITH WORD  
EMBEDDING FASTTEXT**

**A Thesis**

Presented as Partial Fullfillment of the Requirements  
to Obtain *Sarjana Komputer*  
in Department of Informatics



By :

Dylene Melynea Fernandez

Student ID: 185314125

**INFORMATICS STUDY PROGRAM  
FACULTY OF SCIENCE AND TECHNOLOGY  
SANATA DHARMA UNIVERSITY  
YOGYAKARTA  
2022**

HALAMAN PERSETUJUAN

SKRIPSI

KLASIFIKASI SUASANA PADA CERITA PENDEK BAHASA  
INDONESIA MENGGUNAKAN LSTM DAN BI-LSTM DENGAN WORD  
EMBEDDING FASTTEXT



(Cyprianus Kuntoro Adi, S.J. M.A., M.Sc., Ph.D)

HALAMAN PENGESAIAN

SKRIPSI

KLASIFIKASI SUASANA PADA CERITA PENDEK BAHASA  
INDONESIA MENGGUNAKAN LSTM DAN BI-LSTM DENGAN WORD  
*EMBEDDING FASTTEXT*

Yang Dipersembahkan dan Disusun Oleh :

Dylene Melynea Fernandez

NIM : 185314125

Telah dipertahankan di depan Tim Pengaji

Pada tanggal 5 Januari 2022

dan dinyatakan memenuhi syarat

Susunan Tim Pengaji

Nama Lengkap

Tanda Tangan

Ketua : JB. Budi Darmawan, S.T., M.Sc.

Sekretaris : Drs. Hari Suparwito, S.J., M.App. IT

Anggota : Cypriamus Kuntoro Adi, S.J., M.A., M.Sc., Ph.D

Yogyakarta, 27 Januari 2022

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

Dekan,

Prof. Ir. Sudi Mungkasi, Ph.D.



## HALAMAN PERSEMBAHAN

*“If I dream by myself, it's merely a dream. But if we all dream together, that's the start of a new future.” - Lee Sooman*



## **PERNYATAAN KEASLIAN KARYA**

Saya menyatakan dengan sungguh-sungguh bahwa skripsi yang saya tulis ini tidak memuat karya orang lain, kecuali yang telah disebutkan dalam kutipan dan daftar pustaka, sebagaimana layaknya karya ilmiah.

Yogyakarta, 27 Januari 2022  
Penulis,

Dm

(Dylene Melymea Hernandez)

**LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI  
KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Yang bertanda tangan di bawah ini, saya mahasiswa Universitas Sanata Dharma:

Nama : Dylene Melynea Fernandez

Nomor Induk Mahasiswa : 185314125

Demi pengembangan ilmu pengetahuan, saya memberikan kepada Perpustakaan Universitas Sanata Dharma karya ilmiah saya yang berjudul :

**KLASIFIKASI SUASANA PADA CERITA PENDEK BAHASA  
INDONESIA MENGGUNAKAN LSTM DAN BI-LSTM DENGAN WORD  
EMBEDDING FASTTEXT**

berserta perangkat yang diperlukan (bila ada). Dengan demikian saya memberikan kepada Perpustakaan Universitas Sanata Dharma hak untuk menyimpan, mengalihkan dalam bentuk media lain, mengelolanya dalam bentuk pangkalan data, mendistribusikan secara terbatas, dan mempublikasikannya di Internet atau media lain untuk kepentingan akademis tanpa perlu meminta izin dari saya maupun memberikan royalti kepada saya selama tetap mencantumkan nama saya sebagai penulis.

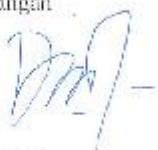
Demikian pernyataan ini yang saya buat dengan sebenarnya.

Dibuat di Yogyakarta

Pada tanggal : 28 Januari 2022

Yang menyatakan

tanda tangan



Dylene Melynea Fernandez

## ABSTRAK

Dalam karya sastra berbentuk prosa, terdapat suatu unsur intrinsik yaitu latar suasana. Latar suasana yang dibuat penulis diharapkan dapat diterima pembaca/penikmat cerita agar lebih memahami cerita dan juga merasakan karakter-karakter yang ada pada cerita. Identifikasi suasana pada cerita bukan hanya diaplikasikan demi kepentingan pembaca/penikmat cerita, namun juga untuk kebutuhan transformasi dari bentuk karya sastra prosa tertulis ke bentuk lainnya seperti pementasan teater, iklan, musikalisisasi, dan lainnya. Terdapat 630 data potongan cerita untuk penelitian ini yang didapatkan dari *web-scraping* di berbagai situs kumpulan cerita pendek bahasa Indonesia yang tidak sensitif terhadap hak cipta seperti cerpenmu.com, yang kemudian diberikan anotasi suasana oleh beberapa anotator yang merupakan pembaca aktif cerita pendek. Klasifikasi suasana didasarkan oleh nilai vektor-vektor tiap kata dalam susunan potongan cerita, atau dapat disebut *Word Embedding*. Model bahasa *Word Embedding* yang diaplikasikan yaitu dengan model *pre-trained* fastText versi wiki Indonesia yang dilanjutkan training dengan *domain-specific corpus*, yaitu dari cerita pendek pada dataset train dan test. Metode klasifikasi teks menggunakan *Long Short-Term Memory* dan *Bidirectional Long Short-Term Memory*. Pengujian menggunakan 630 data dengan model LSTM dan Bi-LSTM yang difokuskan pada variasi arsitektur layer-layernya, dan tuning parameter. Didapatkan performa terbaik yaitu dengan akurasi sebesar 60.8% untuk dataset cerita dengan 4 kelas yang sudah dilakukan *upsample* menggunakan arsitektur Bi-LSTM yang menggunakan regularisasi L2.

## ABSTRACT

In a literary work in the form of prose, there is an intrinsic element, namely the setting of the atmosphere. The setting of the atmosphere created by the author is expected to be accepted by the reader/connoisseur of the story in order to better understand the story and also feel the characters in the story. The identification of the atmosphere in the story is not only applied for the benefit of the readers/connoisseurs of the story, but also for the need for transformation from the form of written prose literature to other forms such as theater performances, advertisements, musicals, and others. There are 630 story snippets for this research which were obtained from web-scraping on various Indonesian short story collection sites that are not sensitive to copyright such as cerpenmu.com, which were then annotated by several annotators who are active readers of short stories. The atmosphere classification is based on the value of the vectors of each word in the arrangement of story pieces, or it can be called Word Embedding. The Word Embedding language model that is applied is the fastText version of the Indonesian wiki pre-trained model which is followed by training with a domain-specific corpus, namely from short stories on the train and test datasets. The text classification method uses Long Short-Term Memory and Bidirectional Long Short-Term Memory. The test uses 630 data with LSTM and Bi-LSTM models which are focused on variations in the architecture of the layers, and tuning parameters. The best performance was obtained with an accuracy of 60.8% for the story dataset with 4 classes that had been upsampled using the Bi-LSTM architecture using L2 regularization.

## KATA PENGANTAR

Puji dan syukur penulis haturkan kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunianya, penulis dapat menyelesaikan tugas akhir ini yang berjudul **“Klasifikasi Suasana Pada Cerita Pendek Bahasa Indonesia Menggunakan LSTM dan Bi-LSTM dengan Word Embedding FastText”**.

Dalam penyelesaian tugas akhir ini, penulis bermaksud menghaturkan terima kasih kepada seluruh pihak yang telah memberikan dukungan, motivasi, dan doa, ucapan terima kasih penulis sampaikan kepada :

1. Romo Cyprianus Kuntoro Adi, S.J. M.A., M.Sc., Ph.D. selaku dosen pembimbing yang telah memberikan waktu, arahan, serta bimbingan kepada penulis selama menyelesaikan tugas akhir ini.
2. Orang tua dan saudara penulis yang selalu memberikan dukungan kepada penulis.
3. Seluruh teman-teman Informatika 2018 yang sudah membantu dan mendukung penulis dalam pembuatan tugas akhir ini.
4. Seluruh pihak yang sudah mendukung penulis dalam menyelesaikan tugas akhir ini.

Semoga penelitian ini dapat berguna bagi pembaca, terutama bagi teman-teman Informatika.

## DAFTAR ISI

JUDUL TUGAS AKHIR.....	<a href="#">i</a>
JUDUL TUGAS AKHIR (BAHASA INGGRIS).....	<a href="#">ii</a>
HALAMAN PERSETUJUAN .....	iii
HALAMAN PENGESAHAN.....	iv
HALAMAN PERSEMBAHAN .....	v
PERNYATAAN KEASLIAN KARYA .....	vi
LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS .....	vii
ABSTRAK .....	viii
ABSTRACT .....	ix
KATA PENGANTAR .....	x
DAFTAR ISI.....	xi
DAFTAR TABEL.....	xiii
DAFTAR GAMBAR .....	xv
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah.....	2
1.3 Tujuan .....	2
1.4 Batasan Masalah.....	2
1.5 Metodologi Penelitian .....	3
1.6 Sistematika Penulisan .....	3
BAB II LANDASAN TEORI .....	5
2.1 Knowledge Discovery in Database (KDD).....	5
2.2 Klasifikasi .....	6
2.3 Word Embedding .....	6
2.4 Deep Learning .....	9
2.5 Long Short-Term Memory (LSTM).....	16
2.6 Bidirectional Long Short-Term Memory (Bi-LSTM).....	18
BAB III METODE PENELITIAN.....	19
3.1 Bahan dan Alat.....	19
3.2 Tahap-tahap Penelitian.....	19
3.2.1 Studi Pustaka .....	19
3.2.2 Pengumpulan Data .....	20
3.2.3 Perancangan Alat Uji .....	20
3.2.4 Pengolahan Data .....	20
3.3 Desain Alat Uji.....	22
3.3.1 Preprocessing .....	22
3.3.2 Transformasi Data .....	23
3.3.3 Pembagian Data Latih dan Uji .....	24
3.3.4 Modeling .....	25

3.3.5	Akurasi	40
3.4	Desain Antar Muka .....	40
3.4.1	Halaman Awal	40
3.4.2	Halaman Bentuk Model dan Uji	41
3.4.3	Halaman Uji Tunggal	42
3.5	Skenario pengujian.....	42
BAB IV IMPLEMENTASI DAN ANALISIS HASIL.....		48
4.1	Implementasi Perangkat Lunak.....	48
4.1.1	Preprocessing	48
4.1.2	Modeling	54
4.1.3	Evaluasi	55
4.2	Analisa Hasil .....	55
4.2.1	Pengujian Perangkat Lunak	55
4.2.2	Pengujian dengan Dataset	56
BAB V PENUTUP.....		74
5.1	Kesimpulan .....	74
5.2	Saran.....	75
Daftar Pustaka .....		76
Daftar Istilah/Glosarium.....		78
LAMPIRAN I: PENGUJIAN PERANGKAT LUNAK .....		89
LAMPIRAN 2: LAPORAN KLASIFIKASI, CONF MATRIX, GRAFIK LOSS		98

## DAFTAR TABEL

Tabel 3. 1 <i>Dataset dummy</i> .....	25
Tabel 3. 2 Format data untuk diproses .....	25
Tabel 3. 3 <i>Weight kernel epoch 1</i> iterasi 1 .....	26
Tabel 3. 4.....	26
Tabel 3. 5.....	27
Tabel 3. 6.....	27
Tabel 3. 7.....	27
Tabel 3. 8.....	27
Tabel 3. 9.....	28
Tabel 3. 10.....	28
Tabel 3. 11 <i>Weight kernel epoch 1</i> iterasi 2 .....	28
Tabel 3. 12.....	28
Tabel 3. 13.....	29
Tabel 3. 14.....	29
Tabel 3. 15.....	29
Tabel 3. 16.....	29
Tabel 3. 17.....	30
Tabel 3. 18.....	30
Tabel 3. 19 Prediksi, akurasi, <i>loss epoch 1</i> .....	30
Tabel 3. 20.....	32
Tabel 3. 21.....	32
Tabel 3. 22.....	32
Tabel 3. 23.....	32
Tabel 3. 24.....	32
Tabel 3. 25.....	32
Tabel 3. 26.....	33
Tabel 3. 27.....	33
Tabel 3. 28.....	33
Tabel 3. 29.....	33
Tabel 3. 30.....	33
Tabel 3. 31.....	33
Tabel 3. 32.....	34
Tabel 3. 33.....	34
Tabel 3. 34.....	34
Tabel 3. 35.....	34
Tabel 3. 36.....	35
Tabel 3. 37 <i>Weight kernel epoch 2</i> iterasi 1 .....	35
Tabel 3. 38.....	35
Tabel 3. 39.....	36
Tabel 3. 40.....	36
Tabel 3. 41.....	36
Tabel 3. 42.....	36
Tabel 3. 43.....	37
Tabel 3. 44.....	37

Tabel 3. 45 <i>Weight kernel epoch</i> 2 iterasi 2 .....	37
Tabel 3. 46.....	37
Tabel 3. 47.....	38
Tabel 3. 48.....	38
Tabel 3. 49.....	38
Tabel 3. 50.....	38
Tabel 3. 51.....	39
Tabel 3. 52.....	39
Tabel 3. 53 Prediksi, akurasi, <i>loss epoch</i> 2 .....	39
Tabel 3. 54 Skenario Arsitektur Sederhana .....	43
Tabel 3. 55 Skenario <i>hyperparameter</i> sederhana .....	43
Tabel 3. 56 Skenario Arsitektur Kompleks 1.....	44
Tabel 3. 57 Skenario <i>hyperparameter</i> Kompleks 1.....	44
Tabel 3. 58 Skenario Arsitektur Kompleks 2.....	45
Tabel 3. 59 Skenario <i>hyperparameter</i> Kompleks 2.....	46
Tabel 3. 60 Skenario <i>hyperparameter</i> Kompleks 3.....	46
Tabel 4. 1 Pengujian Arsitektur LSTM Sederhana dengan 6 Kelas .....	57
Tabel 4. 2 Pengujian Arsitektur Bi-LSTM Sederhana dengan 6 Kelas .....	58
Tabel 4. 3 Perbandingan f1-score High Acc Sederhana 6 Kelas .....	59
Tabel 4. 4 Pengujian Arsitektur Sederhana dengan 5 Kelas .....	59
Tabel 4. 5 Perbandingan f1-score High Acc Sederhana 5 Kelas .....	60
Tabel 4. 6 Pengujian Arsitektur Sederhana dengan 4 Kelas 5 Kelas .....	62
Tabel 4. 7 Perbandingan f1-score High Acc Sederhana 4 Kelas .....	62
Tabel 4. 8 Pengujian Arsitektur Kompleks LSTM 1 dengan 4 Kelas.....	63
Tabel 4. 9 Pengujian Arsitektur Kompleks 2 dengan 4 Kelas .....	65
Tabel 4. 10 Pengujian Arsitektur Kompleks 3 dengan 4 Kelas .....	67
Tabel 4. 11 Rekap ujicoba arsitektur sederhana dan kompleks dengan 4 kelas ...	69
Tabel 4. 12 Ujicoba dengan Stemmer Sastrawi .....	72

## DAFTAR GAMBAR

Gambar 2. 1 Ilustrasi <i>n-gram</i> fastText .....	7
Gambar 2. 2 ‘hav’ pada kata ‘behaved’ .....	7
Gambar 2. 3 ‘hav’ pada kata ‘have’ .....	7
Gambar 2. 4 Formula skor <i>subword</i> .....	8
Gambar 2. 5 Cara kerja model <i>fastText</i> <sup>[9]</sup> .....	8
Gambar 2. 6 Proses <i>training</i> pada <i>deep learning</i> .....	9
Gambar 2. 7 Gradient descent dengan momentum <sup>[11]</sup> .....	11
Gambar 2. 8 RMSProp Optimizer <sup>[11]</sup> .....	11
Gambar 2. 9 Pseudocode Adam Optimizer <sup>[12]</sup> .....	11
Gambar 2. 10 Fungsi Aktivasi Sigmoid <sup>[13]</sup> .....	12
Gambar 2. 11 Fungsi Aktivasi TanH <sup>[13]</sup> .....	13
Gambar 2. 12 Fungsi Aktivasi Softmax <sup>[13]</sup> .....	13
Gambar 2. 13 Neural Net Standar dan Neural Net dengan <i>Dropout</i> <sup>[15]</sup> .....	14
Gambar 2. 14 Penambahan nilai penalti <i>L2-regularization</i> pada <i>loss function</i> <sup>[16]</sup>	15
Gambar 2. 15 Categorical Cross-Entropy <sup>[17]</sup> .....	15
Gambar 2. 20 Modul berulang dalam RNN standar berisi <i>layer</i> tunggal <sup>[2]</sup> .....	16
Gambar 2. 21 Modul berulang pada LSTM berisi 4 <i>layer</i> saling berinteraksi <sup>[2]</sup> ..	16
Gambar 2. 22 <i>cell state</i> LSTM <sup>[2]</sup> .....	16
Gambar 2. 23 <i>forget gate</i> LSTM <sup>[2]</sup> .....	17
Gambar 2. 24 informasi baru yang diregulasi oleh <i>input</i> <sup>[2]</sup> .....	17
Gambar 2. 25 alur mendapatkan nilai <i>cell state</i> yang baru <sup>[2]</sup> .....	17
Gambar 2. 26 mekanisme <i>output</i> <sup>[2]</sup> .....	17
Gambar 2. 27 Ilustrasi model Bi-LSTM <sup>[18]</sup> .....	18
Gambar 3. 1 Skema Penelitian .....	19
Gambar 3. 2 Tahapan proses alat uji .....	22
Gambar 3. 3 .....	27
Gambar 3. 4 .....	27
Gambar 3. 5 .....	27
Gambar 3. 6 .....	28
Gambar 3. 7 .....	28
Gambar 3. 8 .....	29
Gambar 3. 9 .....	29
Gambar 3. 10 .....	29
Gambar 3. 11 .....	30
Gambar 3. 12 .....	30
Gambar 3. 13 .....	31
Gambar 3. 14 .....	36
Gambar 3. 15 .....	36
Gambar 3. 16 .....	36
Gambar 3. 17 .....	37
Gambar 3. 18 .....	38
Gambar 3. 19 .....	38
Gambar 3. 20 .....	38
Gambar 3. 21 .....	39

Gambar 3. 22 .....	39
Gambar 3. 23 GUI Halaman Awal.....	40
Gambar 3. 24 GUI Halaman Bentuk Model dan Uji .....	41
Gambar 3. 25 GUI Halaman Uji Tunggal .....	42
Gambar 4. 1 <i>Lowercasing</i> dan mengganti beberapa kata menjadi lebih sesuai....	48
Gambar 4. 2 <i>Load file</i> hapus <b>corpus.txt</b> berisi nama tokoh dan kata tidak perlu.	49
Gambar 4. 3 Penambahan tanda baca untuk dimasukkan ke <i>list stopword nltk</i> .	49
Gambar 4. 4 Normalisasi vektor, <i>stopwords &amp; punctuation removal, tokenizing</i> ..	49
Gambar 4. 5 Pemanggilan <i>function norm_sent_vector</i> .....	49
Gambar 4. 6 Daftar kata yang tidak ada pada <i>pretrained fastText wiki-id 300</i> ....	50
Gambar 4. 7 Tambahan kalimat sesuai kata di luar model <i>pretrained</i> .....	50
Gambar 4. 8 Melanjutkan latih <i>pretrained</i> model bahasa <i>fastText</i> .....	50
Gambar 4. 9 <i>One hot encoding</i> fitur target .....	51
Gambar 4. 10 <i>Padding</i> dan transformasi <i>input shape</i> .....	52
Gambar 4. 11 Penggunaan <i>train-test split sklearn</i> .....	52
Gambar 4. 12 Kelas mayor-minor dan <i>bias class weights</i> .....	53
Gambar 4. 13 Pemisahan <i>train-test</i> sekaligus <i>shuffling</i> .....	53
Gambar 4. 14 Kelas mayor-minor untuk data pada <i>train set</i> .....	53
Gambar 4. 15 <i>Upsampling</i> kelas minor dan penggabungan .....	54
Gambar 4. 16 Membuat <i>dict class_weights</i> untuk <i>hyperparameter class_weight</i>	54
Gambar 4. 17 Contoh pembangunan model <i>Bi-LSTM</i> dengan <i>regularizers</i> .....	54
Gambar 4. 18 <i>Compile</i> model yang telah dikonfigurasikan.....	55
Gambar 4. 19 <i>Code</i> untuk menunjukkan <i>confusion matrix</i> .....	55
Gambar 4. 20 <i>Code</i> untuk menunjukkan laporan klasifikasi .....	55
Gambar 4. 21 Grafik Jumlah Data Tiap Kelas .....	56
Gambar 4. 22 <i>train-val loss rmsprop</i> .....	57
Gambar 4. 23 <i>train-val loss adam</i> .....	57
Gambar 4. 24 <i>train-val loss sgd</i> .....	58
Gambar 4. 25 Grafik Jumlah Data Tiap Kelas untuk 5 Kelas.....	59
Gambar 4. 26 Grafik Jumlah Data Tiap Kelas untuk 4 Kelas.....	60
Gambar 4. 27 Grafik <i>loss</i> kompleks 1 no.1.....	63
Gambar 4. 28 Grafik <i>loss</i> kompleks 1 no.2.....	63
Gambar 4. 29 Grafik <i>loss</i> kompleks 1 no.5.....	64
Gambar 4. 30 Grafik <i>loss</i> kompleks 1 no.6.....	64
Gambar 4. 31 Grafik <i>loss</i> kompleks 1 no.7.....	64
Gambar 4. 34 Grafik <i>loss</i> kompleks 2 no.2 <i>LSTM</i> .....	66
Gambar 4. 34 Grafik <i>loss</i> kompleks 2 no.6 <i>LSTM</i> .....	66
Gambar 4. 34 Grafik <i>loss</i> kompleks 2 no.2 <i>Bi-LSTM</i> .....	66
Gambar 4. 35 Grafik <i>loss</i> kompleks 3 no.2 <i>LSTM</i> .....	68
Gambar 4. 36 Grafik <i>loss</i> kompleks 3 no.10 <i>LSTM</i> .....	68
Gambar 4. 37 Grafik <i>loss</i> kompleks 3 no.5 <i>LSTM</i> .....	68
Gambar 4. 38 Grafik <i>loss</i> kompleks 3 no.11 <i>LSTM</i> .....	68
Gambar 4. 39 Grafik <i>loss</i> kompleks 3 no.3 <i>Bi-LSTM</i> .....	69
Gambar 4. 40 Rekap akurasi hasil uji .....	70
Gambar 4. 41 Rekap loss hasil uji.....	71
Gambar 4. 42 <i>Loss val-set</i> dan tanpa <i>stemmer</i> dan <i>dengan stemmer</i> .....	72

Gambar 4. 43 Akurasi *val-set* dan tanpa *stemmer* dan dengan *stemmer*..... 72



## BAB I

### PENDAHULUAN

#### 1.1 Latar Belakang

Dalam karya sastra berbentuk prosa, terdapat suatu unsur intrinsik yaitu latar. Menurut Nofriani, latar-latar yang dibuat penulis diharapkan dapat diterima pembaca/penikmat cerita agar lebih memahami cerita, pesan yang terkandung di dalamnya, dan juga merasakan karakter-karakter yang ada pada cerita<sup>[1]</sup>.

Salah satu jenis latar yaitu latar suasana. Latar suasana dapat memberikan gambaran besar situasi dan kondisi dari suatu bagian cerita tanpa perlu mengetahui detail lainnya. Identifikasi suasana pada cerita bukan hanya diaplikasikan untuk kepentingan pembaca/penikmat cerita, namun juga untuk kebutuhan transformasi dari bentuk karya sastra prosa tertulis ke bentuk lainnya seperti pementasan teater, iklan, musikalisisasi, dan lainnya.

Dalam penentuan suasana, diperlukan adanya pemahaman konteks dalam cerita. Salah satu fitur dari cerita yaitu memiliki struktur sekuensial yang berarti susunan kata di kalimat sebelumnya dapat berdampak ke konteks kalimat setelahnya. Untuk mendapatkan penentuan suasana, dalam *deep learning* terdapat metode LSTM (*Long Short-Term Memory*) yang cocok untuk memecahkan permasalahan sekuensial, menurut Colah<sup>[2]</sup>.

Dalam penerapan metode-metode *deep learning* untuk permasalahan sekuensial, diperlukan juga teknik *feature-engineering Word Embedding*.

*Word Embedding* memetakan setiap kata pada dokumen ke dalam dense vector, di mana sebuah vektor merepresentasikan proyeksi kata di dalam ruang vektor. Posisi kata tersebut dipelajari dari teks atau berdasarkan kata-kata di sekitarnya. *Word embedding* ini dapat menangkap makna semantik dan sintaktik kata, jelas Nurdin dkk.<sup>[3]</sup>

Ada beberapa jenis *Word Embedding* yaitu *Word2Vec*, *Glove*, *FastText*.

Klasifikasi teks menggunakan LSTM dengan Word Embedding Glove dan Word2Vec pernah diteliti oleh Sari<sup>[4]</sup>. Dalam penelitiannya, dilakukan klasifikasi untuk 4 label artikel berita dan didapatkan akurasi 95,38% untuk model kelima

menggunakan *Word2Vec*.

Adapun penelitian lain oleh Abdillah, yaitu mengklasifikasi emosi pada lirik lagu menggunakan Bi-LSTM dengan *Word Embedding GloVe*<sup>[5]</sup>. Akurasi yang didapatkan yaitu 91.08% dengan diberlakukannya parameter *dropout layer*, dan *activity regularization* di arsitektur model Bi-LSTM.

Berdasarkan latar belakang tersebut, dalam tugas akhir ini penulis menggunakan metode LSTM dan Bi-LSTM dengan *Word Embedding FastText* untuk mengklasifikasikan suasana yang terkandung dalam potongan cerita pendek berbahasa Indonesia. Diharapkan dengan penelitian ini dapat membandingkan kinerja LSTM dan Bi-LSTM yang didukung oleh model *pretrained fastText* dengan mekanisme *subword*.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, maka didapat rumusan masalah sebagai berikut.

1. Bagaimana metode LSTM dan Bi-LSTM mampu untuk mengklasifikasi suasana pada cerita pendek berbahasa Indonesia dengan baik?
2. Berapa nilai loss dan akurasi untuk metode LSTM dan Bi-LSTM dengan *Word Embedding FastText*?
3. Bagaimana arsitektur dan parameter yang diterapkan pada model untuk mendapatkan hasil klasifikasi yang optimal?

## 1.3 Tujuan

- a) Menerapkan metode LSTM dan Bi-LSTM yang mampu untuk mengklasifikasi suasana pada cerita pendek berbahasa Indonesia dengan baik.
- b) Menguji loss dan akurasi untuk metode LSTM dan Bi-LSTM dengan *Word Embedding FastText*.
- c) Menguji arsitektur dan parameter yang diterapkan pada model untuk mendapatkan hasil klasifikasi yang optimal.

## 1.4 Batasan Masalah

Dalam penelitian ini, masalah yang dibatasi adalah:

- a) Merancang model dengan menggunakan metode LSTM dan Bi-LSTM untuk mengklasifikasi suasana pada cerita pendek berbahasa Indonesia.
- b) Corpus yang dijadikan model bahasa (*language model*) berasal dari *pre-trained model fastText* versi wiki Indonesia dimensi 300 ditambah dengan domain-specific corpus dari dataset cerita pendek.
- c) Dataset yang digunakan adalah potongan cerita pendek berbahasa Indonesia untuk setiap entrinya.

## 1.5 Metodologi Penelitian

Dalam melakukan penelitian ini, langkah-langkah yang dilakukan adalah sebagai berikut:

- a) Studi Pustaka

Pada langkah ini, peneliti mempelajari teori-teori melalui buku, artikel, jurnal yang berkaitan dengan algoritma Long Short-Term Memory dan Bidirectional Long Short-Term Memory dan metode-metode lain yang dibutuhkan.

- b) Pembuatan alat uji

Melakukan perancangan terhadap sistem yang dibangun, mulai dari perancangan *user interface* dan *listing* program.

- c) Evaluasi dan Analisa Hasil

Peneliti melakukan penarikan kesimpulan setelah melakukan uji coba pada sistem.

## 1.6 Sistematika Penulisan

- a) BAB I Pendahuluan.

Dalam bab ini diuraikan tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metode, peneltian, dan sistematika penelitian.

- b) BAB II Tinjauan Pustaka

Pada bab ini diuraikan landasan teori dan masalah yang berhubungan dengan *Word Embedding*, *Deep Learning* khususnya metode LSTM dan Bi-LSTM dengan peneliti terkait.

c) BAB III Metodologi Penelitian

Dalam bab ini akan dijelaskan data dan rencana langkah-langkah yang akan dilakukan dalam melakukan penelitian ini. Selain itu dijelaskan juga bagaimana proses penerimaan/implementasi dan analisis dari penerapan metode LSTM dan Bi-LSTM untuk mengklasifikasi suasana pada teks cerita pendek berbahasa Indonesia. Selain itu juga terdapat paparan desain model yang akan digunakan dalam penelitian ini.

d) BAB IV Implementasi dan Analisa Hasil

Pada bab ini akan diuraikan hasil implementasi dari desain yang sudah dirancang dan analisis dari hasil yang didapatkan.

e) BAB V Penutup

Pada bab ini akan diuraikan kesimpulan dari hasil percobaan-percobaan yang sudah dilakukan dan juga terdapat saran dari penulis untuk pengembangan dari penelitian ini.

f) Daftar Pustaka

Berisi tentang daftar referensi dan buku-buku yang digunakan dalam melakukan penelitian ini.

## BAB II

### LANDASAN TEORI

Pada Bab II, akan dijelaskan beberapa materi terkait penelitian ini, yaitu: *Knowledge Discovery in Database* (KDD), Klasifikasi, *Word Embedding*, *Deep Learning*, LSTM, dan Bi-LSTM. KDD membahas mengenai tahapan untuk mendapatkan ilmu dari data. Klasifikasi membahas pemahaman dasar mengenai klasifikasi itu sendiri sebagai metode cerdas untuk mendapatkan pengetahuan dari data yang ada. *Word Embedding* berkaitan dengan ekstraksi fitur dan transformasi data yang akan diolah. *Deep Learning* membahas modal-modal yang digunakan dalam arsitektur LSTM, Bi-LSTM, dan proses *embedding*. LSTM dan Bi-LSTM menjelaskan mengenai arsitektur yang digunakan pada penelitian ini.

#### 2.1 Knowledge Discovery in Database (KDD)

*Knowledge Discovery in Database* (KDD) merupakan proses penemuan pengetahuan dalam *database*. Secara lengkap KDD didefinisikan sebagai proses ekstraksi atau identifikasi pola, pengetahuan dan informasi potensial dari sekumpulan data yang besar. Pengetahuan dan informasi yang dihasilkan dari KDD bersifat sah, baru, mudah dimengerti, dan bermanfaat.

Menurut Han dan Kamber<sup>[6]</sup>, dalam proses pencarian suatu *knowledge* ada beberapa langkah yang perlu dilakukan, yaitu:

1. *Data cleaning* (untuk menghilangkan *noise* dan data yang inkonsisten)
2. *Data integration* (di mana banyak sumber data dikombinasikan)
3. *Data selection* (di mana data relevan terhadap tugas analisis yang diambil dari *database*)
4. *Data transformation* (di mana data ditransformasikan dan terkonsolidasi menjadi bentuk yang sesuai untuk mining dengan menunjukkan ringkasan atau agregasi operasi)
5. *Data mining* (proses esensial di mana metode cerdas diterapkan untuk mengekstrak pola data)
6. *Pattern evaluation* (untuk identifikasi pola sungguhan yang menarik yang

merepresentasikan pengetahuan menarik)

7. *Knowledge presentation* (di mana visualisasi dan teknik-teknik representasi pengetahuan digunakan untuk menyajikan pengetahuan hasil tambangan ke pengguna).

Langkah 1 hingga 4 adalah bentuk berbeda dari tahap praproses (*preprocessing*), di mana data disiapkan untuk penambangan. Penjabaran tahapan lebih lengkap mengenai praproses untuk data teks menurut Ganesan<sup>[7]</sup> yaitu meliputi: *lowercasing, stemming, lemmatization, stopword removal, normalization, noise removal*.

Pada langkah *data mining*, dapat adanya interaksi dengan pengguna atau basis pengetahuan (*knowledge base*). Pola yang menarik disajikan kepada pengguna dan dapat disimpan sebagai pengetahuan baru di dasar pengetahuan (*knowledge base*).

## 2.2 Klasifikasi

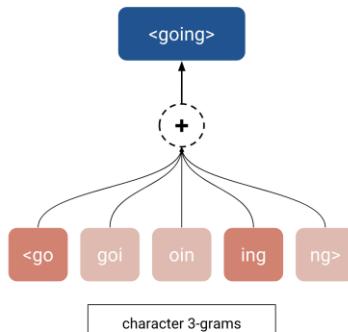
Klasifikasi adalah salah 1 metode cerdas yang terdapat di langkah data mining. Klasifikasi adalah bentuk analisis data yang mengekstrak model yang menggambarkan kelas data penting.

Sebuah model, yang disebut *classifier*/pengklasifikasi, memprediksi label kelas kategorikal (diskrit, tidak berurutan)<sup>[6]</sup>. Terdapat beberapa bentuk dari model klasifikasi salah satunya adalah *neural network*, dan salah satu kelas dari *neural network* yaitu RNN memiliki salah satu arsitektur yang memiliki karakteristik dapat mengingat informasi dalam jangka waktu yang lama yaitu LSTM.

## 2.3 Word Embedding

Word Embedding adalah jenis teknik mendapatkan fitur dari teks berupa kata yang ditransformasikan menjadi bentuk vektor-vektor bilangan riil. Salah satu jenis word embeddings yaitu FastText. FastText merupakan pengembangan dari Word2Vec, jelas Bojanowski<sup>[8]</sup>. Hal unik dari FastText yaitu terletak pada mekanisme subword. FastText mempertimbangkan informasi n-gram dari sebuah kata yang menyebabkan FastText bekerja baik untuk mengidentifikasi prefiks, sufiks, dan kata-kata yang tidak muncul pada data latih karena kata tersebut dipecah

menjadi n-gram dan didapatkan embedding vektornya, ungkap Nurdin dkk.<sup>[3]</sup>.



Gambar 2. 1 Ilustrasi *n-gram* fastText

Pada gambar 2.1 ditunjukkan bagaimana kata *going* ditemukan nilai vektornya yaitu dengan menggabungkan vektor-vektor dari *subword* ‘*going*’ itu sendiri. Terdapat 2 *subword* ‘*<go*’ dan ‘*ing*’ yang memungkinkan bahwa ‘*going*’ ada kaitannya dengan ‘*go*’ biasa. Tanda ‘*<*’ dan ‘*>*’ penting untuk menandakan prefiks dan sufiks dari suatu kata, karena seperti ‘*her*’ pada ‘*<where>*’ berbeda dengan ‘*<her>*’ pada ‘*<her>*’. Selain itu, penentuan bahwa ‘*going*’ lebih condong tersusun dari ‘*<go*’ dan ‘*ing*’ juga ditentukan dengan pertimbangan kemunculan *subword* itu sendiri. Karena ‘*<go*’ dan ‘*ing*’ lebih banyak ditemukan di kata lain, sehingga *FastText* juga dapat mempelajari partikelnya. Dalam memecah *subword* itu sendiri, *FastText* menggunakan hashing dari 1 sampai sejuta dengan fungsi *hashing Fowler Noll Vo*.

```
Hashing n-gram: <ha hash:78104
Hashing n-gram: <hav hash:1758378
Hashing n-gram: <have hash:1181405
Hashing n-gram: <have> hash:833369
Hashing n-gram: hav hash:1054492
Hashing n-gram: have hash:1919355
Hashing n-gram: have> hash:246303
Hashing n-gram: ave hash:1557521
Hashing n-gram: ave> hash:373757
Hashing n-gram: ve> hash:1088654
```

Gambar 2. 3 ‘hav’ pada kata ‘have’

```
Hashing n-gram: <be hash:168602
Hashing n-gram: <beh hash:915446
Hashing n-gram: < beha hash:859189
Hashing n-gram: < behav hash:1289465
Hashing n-gram: beh hash:709176
Hashing n-gram: beha hash:333083
Hashing n-gram: behav hash:583191
Hashing n-gram: behave hash:478326
Hashing n-gram: eha hash:175867
Hashing n-gram: ehav hash:703607
Hashing n-gram: ehave hash:1254998
Hashing n-gram: ehaved hash:941238
Hashing n-gram: hav hash:1054492
Hashing n-gram: have hash:1919355
Hashing n-gram: have> hash:1919355
Hashing n-gram: haved hash:696781
Hashing n-gram: haved> hash:1719817
Hashing n-gram: ave hash:1557521
Hashing n-gram: aved hash:807087
Hashing n-gram: aved> hash:1596739
Hashing n-gram: ved hash:1102944
Hashing n-gram: ved> hash:1619962
Hashing n-gram: ed> hash:477174
```

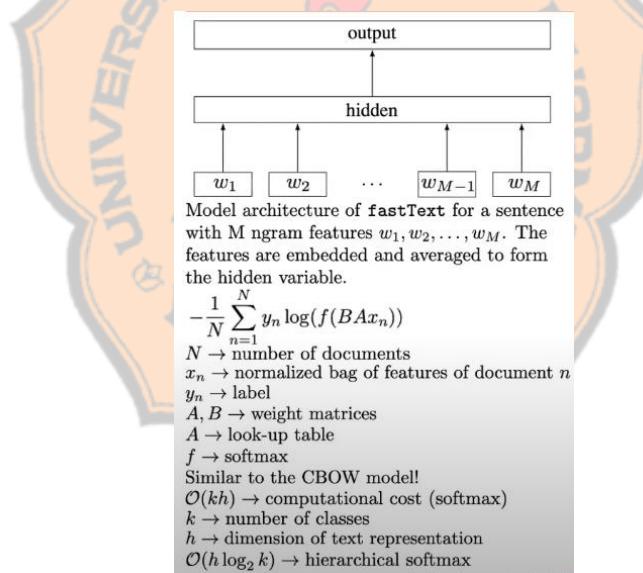
Gambar 2. 2 ‘hav’ pada kata ‘behaved’

Diperlihatkan pada Gambar 2.2 dan 2.3 yaitu hashing dari subword ‘hav’ di masing-masing 2 kata (‘have’ dan ‘behaved’) adalah sama, yang nantinya ini akan mengarah kepada *parameter sharing* yang membantu *FastText* dalam memahami bahwa suatu kata yang berimbuhan memiliki arti yang dekat dengan bentuk dasarnya.

$$s(w, c) = \sum \mathbf{z}_g^\top \mathbf{v}_c.$$

Gambar 2. 4 Formula skor subword

Penjelasan dari Gambar 2.4 adalah sebagai berikut: Terdapat  $G$  (banyak)  $n$ -gram pada kamus. Dan terdapat kata  $w$ , yang kemudian dinyatakan  $G_w \subset \{1, \dots, G\}$  yang merupakan kumpulan  $n$ -gram yang muncul di  $w$ . Selanjutnya adalah mengasosiasikan representasi vektor  $z_g$  untuk tiap  $n$ -gram  $g$ . Representasi kata yang final adalah total dari representasi-representasi vektor dari tiap  $n$ -gram tersebut<sup>[8]</sup>.



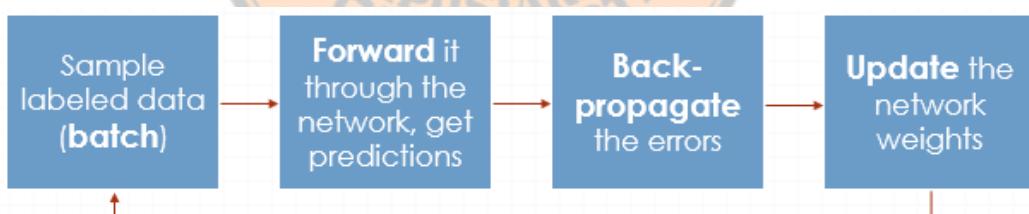
Gambar 2. 5 Cara kerja model *fastText*<sup>[9]</sup>

Pada Gambar 2.5, dijelaskan bahwa arsitektur model *fastText* untuk kalimat dengan fitur berjumlah  $M$  ngram yaitu terdiri dari *weight-weight* fitur  $n$ -gram tersebut yang kemudian diproses pada *hidden layer*, dan akhirnya diproses pada *output layer*. Fitur tersebut ditambatkan dan dirata-ratakan untuk membentuk variabel tersembunyi. Rumus yang digunakan ditunjukkan seperti pada Gambar 2.5.

## 2.4 Deep Learning

*Deep Learning* adalah sub-bagian dari *Machine Learning* yang menyerupai pola pikiran manusia. *Deep Learning* terdapat 3 jenis yang populer: *Artificial Neural Network*, *Convolutional Neural Network*, dan *Recurrent Neural Network*. *Deep Learning* populer dalam menangani permasalahan *speech recognition*, *natural language processing*, *image processing*. Hal tersebut disebabkan karena *deep learning* membutuhkan *feature engineering* yang sesuai dan dapat melakukan *feature extraction* sekaligus.

Dalam melakukan *feature extraction* tersebut, *deep learning* memecah masalah menjadi bagian terkecil sehingga seperti proses belajar yang diharapkan semakin membaik dan memahami. Selain itu *deep learning* semakin baik performanya jika data yang digunakan juga sangat besar. Hal ini sangat relevan dengan ketiga bidang yang sebelumnya disebutkan karena data yang akan diolah pada ketiga bidang tersebut bukanlah data gambar mentah, ataupun rekaman suara lengkap, dan suatu komposisi kalimat lengkap yang diproses langsung, melainkan merupakan bentuk yang sudah dipecah menjadi bagian kecil seperti *pixel*, amplitudo di suatu sekuens, dan *corpus* yang dalam 1 data terdiri dari banyak masing-masing elemen tersebut.



Gambar 2. 6 Proses *training* pada *deep learning*

Proses *training* pada *deep learning* yaitu seperti ditunjukkan pada Gambar 2.6. Mula-mula data diambil sampelnya, yang dinamakan *batch*. Kemudian melakukan *feed-forward* yang terdapat beberapa perhitungan sesuai dengan arsitektur *deep learning* yang dipakai.

Pada proses *feed-forward*, input dikalikan dengan *weight-weight* dan diberi nilai *bias* juga, kemudian hasil dari perhitungan tersebut dilanjutkan sampai pada *output layer* yang terdapat fungsi aktivasi dalam menghitung probabilitas *output*,

setelah mendapatkan probabilitas *output*, maka didapatkan prediksi. Setelah melakukan *feed-forward* dan mendapatkan prediksi dan nilai *loss*, maka selanjutnya adalah melakukan *backpropagation*.

*Backpropagation* berguna untuk merunut balik turunan dari nilai *error* atau *loss* yang didapatkan. *Backpropagation* terdapat 2 tipe, *static* dan *recurrent*. Pada penelitian ini diterapkan *backpropagation* tipe *recurrent* karena arsitektur yang digunakan adalah bagian dari *Recurrent Neural Network*, yaitu LSTM dan Bi-LSTM. Pada proses *backpropagation*, dalam mendapatkan nilai-nilai turunan yang dimulai dari *loss* kemudian ke tiap *weight* adalah dengan menggunakan metode *chain-rule*. Karena alur dari *feed-forward* merupakan alur yang berkesinambungan, maka dalam melakukan *backpropagation* juga menyambungkan dengan hasil turunan sebelumnya (arah dari *loss* ke input).

Setelah melakukan *backpropagation*, maka didapatkan turunan *weight* yang sesuai untuk setiap *gate* (pada LSTM), maka dilakukan pembaruan *weight* dengan beberapa metode *gradient-optimizer*.

Ada beberapa properti dalam *deep learning* yaitu *gradient descent optimizer*, fungsi aktivasi, regularisasi, *loss function*, dan *attention*.

### 1. *Gradient descent optimizer*

Terdapat 2 *gradient descent optimizer* yang populer digunakan untuk domain *state-of-the-art* yaitu *RMSProp* dan *Adam*.

#### - *RMSProp (Root mean square propagation)*

*RMSProp* adalah algoritma yang mengatur *learning rate* berdasarkan besaran nilai rata-rata dari *weight*. *RMSProp* menggunakan nilai pertama dalam gradien untuk menentukan nilai rata-rata dari *weight*, jelas Duchi dkk.<sup>[10]</sup>. *RMSProp* hampir sama dengan *gradient descent* dengan momentum, perbedaannya hanya terletak di bagaimana gradien-gradien dihitung. Menurut Gandhi<sup>[11]</sup>, *RMSProp* membatasi osilasi pada arah vertikal sehingga memungkinkan untuk menaikkan *learning rate* yang menyebabkan algoritma dapat mengambil langkah yang besar pada arah horizontal yang lebih cepat konvergen. Nilai momentum dilambangkan dengan beta dan biasanya ditetapkan

dengan 0,9<sup>[11]</sup>.

$$v_{dw} = \beta \cdot v_{dw} + (1 - \beta) \cdot dw$$

$$v_{db} = \beta \cdot v_{db} + (1 - \beta) \cdot db$$

$$W = W - \alpha \cdot v_{dw}$$

$$b = b - \alpha \cdot v_{db}$$

Gambar 2. 7 Gradient descent dengan momentum<sup>[11]</sup>

$$v_{dw} = \beta \cdot v_{dw} + (1 - \beta) \cdot dw^2$$

$$v_{db} = \beta \cdot v_{db} + (1 - \beta) \cdot db^2$$

$$W = W - \alpha \cdot \frac{dw}{\sqrt{v_{dw}} + \epsilon}$$

$$b = b - \alpha \cdot \frac{db}{\sqrt{v_{db}} + \epsilon}$$

Gambar 2. 8 RMSProp Optimizer<sup>[11]</sup>

#### - Adam (Adaptive Moment Estimation)

*Adam* adalah algoritma optimasi pengganti untuk *stochastic gradient descent* yang menggabungkan sifat-sifat terbaik dari algoritma *AdaGrad* dan *RMSProp* untuk memberikan optimasi algoritma yang dapat menangani *sparse gradients* pada *noisy problem*. *Adam* relatif mudah dikonfigurasikan di mana parameter konfigurasi default bekerja dengan baik pada sebagian besar masalah.

Berikut adalah algoritma Adam dalam *pseudocode*:

---

```

● α = 0.001, β1 = 0.9, β2 = 0.999, η = 10-8 (Defaults)
m0 ← 0 (Initialize 1st moment vector)
v0 ← 0 (Initialize 2nd moment vector)
i ← 0 (Initialize step)
while Θi not converged do
    i ← i + 1
    gi ← ∇Θfi(Θi-1) (Get gradients at step i)
    mi ← β1 · mi-1 + (1 - β1) · gi (Update biased first moment estimate)
    vi ← β2 · vi-1 + (1 - β2) · gi2 (Update biased second raw moment estimate)
    ̂mi ← mi / (1 - β1i) (Compute bias-corrected first moment estimate)
    ̂vi ← vi / (1 - β2i) (Compute bias-corrected second raw moment estimate)
    Θi ← Θi-1 - α · ̂mi / (̂vi + η) (Update parameters)
end while
return Θi (resulting parameters)

```

---

Gambar 2. 9 Pseudocode Adam Optimizer<sup>[12]</sup>

## 2. Fungsi aktivasi

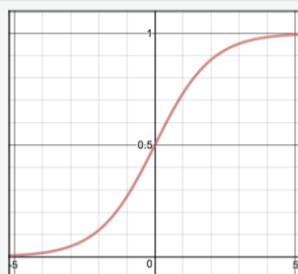
Merupakan fungsi yang digunakan untuk mendapatkan *output* dari *node*.

Ini juga dikenal sebagai Fungsi Transfer. Seperti yang dijelaskan Chaudary<sup>[13]</sup>, terdapat banyak fungsi aktivasi seperti *Sigmoid*, *ReLU*, *Leaky ReLU*, *TanH*, *Softmax*. Namun untuk klasifikasi multi kelas, fungsi aktivasi yang sesuai adalah *softmax*<sup>[13]</sup>. Fungsi aktivasi *Sigmoid* dan *TanH* diaplikasikan dalam gerbang-gerbang LSTM.

### - *Sigmoid*

*Sigmoid* merupakan fungsi aktivasi non-linear. *Sigmoid* memiliki kekurangan yaitu ketika gradien memiliki nilai yang sangat kecil, maka akan mematikan gradien tersebut. Oleh sebab itu, fungsi ini diterapkan pada *input*, *forget*, dan *output gate* LSTM untuk menghilangkan informasi-informasi yang insignifikan.

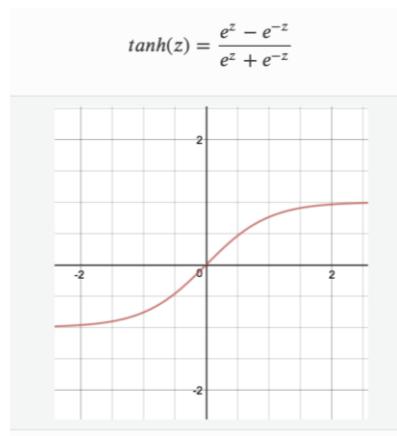
$$S(z) = \frac{1}{1 + e^{-z}}$$



Gambar 2. 10 Fungsi Aktivasi Sigmoid<sup>[13]</sup>

### - *TanH*

*TanH* merupakan fungsi aktivasi non-linear. *TanH* merupakan pengembangan dari fungsi *sigmoid* yang memetakan input ke output yang terbentang dalam *range* [-1,1]. Oleh karena itu, kekurangan fungsi *TanH* juga sama dengan fungsi *sigmoid* yaitu dapat mematikan gradien. Meskipun begitu, fungsi *TanH* memiliki kelebihan yaitu *zero-centered*. *TanH* digunakan juga pada gerbang LSTM untuk menyederhanakan angka yang terbentuk dari *multiplication* dan *addition*.



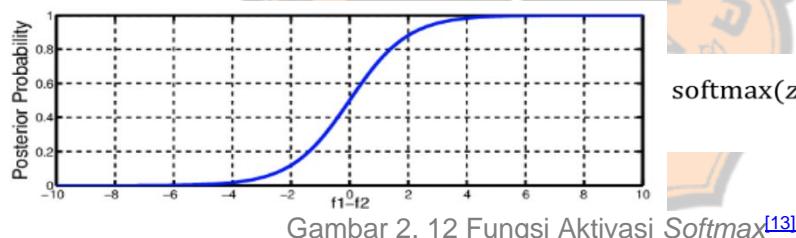
Gambar 2. 11 Fungsi Aktivasi  
TanH<sup>[13]</sup>

- *Softmax*

Umumnya, softmax digunakan pada lapisan terakhir *neural network* yang menghitung distribusi probabilitas pada n-peristiwa yang berbeda.

Keuntungan utama dari fungsi ini adalah mampu menangani banyak kelas. Total distribusi probabilitas yang dihasilkan fungsi softmax yaitu

- 1.



$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \text{ for } j = 1, \dots, K$$

Gambar 2. 12 Fungsi Aktivasi Softmax<sup>[13]</sup>

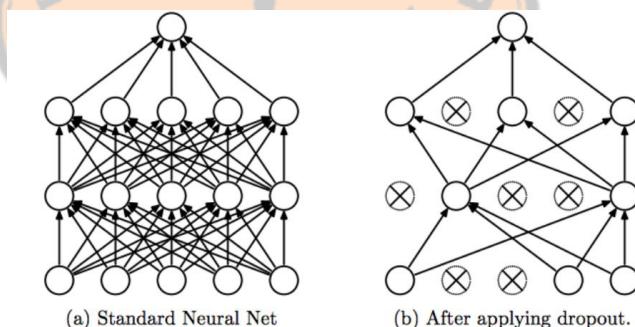
### 3. Regularisasi

Regularisasi adalah teknik yang memodifikasi model neural network dengan tujuan untuk mengurangi *generalization error*, bukan mengurangi *training error* seperti peran *loss function*. Beberapa cara digunakan untuk mengatasi masalah ini, salah satunya adalah dengan memberikan batasan/constraint atau penambahan penalti pada parameter yang digunakan.

Beberapa jenis regularisasi yang dapat digunakan pada proses training neural network adalah sebagai berikut:

- Dropout

Novindasari<sup>[14]</sup> membahas bahwa dropout adalah teknik pemberian nilai *keep probability* yang diberikan untuk setiap *hidden layer* pada arsitektur *neural network*. Nilai *keep probability* ini merupakan proporsi jumlah neuron yang akan digunakan dalam perhitungan dengan memilih secara random. Dengan menghilangkan beberapa neuron yang berbeda pada setiap proses *training*, maka model yang kita buat tidak hanya akan condong terhadap fitur tertentu. Penggunaan dropout menyebabkan ukuran arsitektur menjadi lebih kecil pada setiap proses training. Hal ini menjadikan arsitektur yang dibangun menjadi *less-complex*. Penggunaan *dropout* hanya berlaku pada proses *training* karena pada proses *testing*, semua neuron akan digunakan. Proses pemberian nilai *dropout* pada setiap *hidden layer* dapat ditentukan tergantung pada kompleksitas bobot pada layer itu sendiri. Contohnya, untuk *layer* yang memiliki dimensi yang besar dapat diberikan *dropout* yang besar, karena dimensi yang besar lebih rawan *overfitting*, dan sebaliknya.



Gambar 2. 13 Neural Net Standar dan Neural Net dengan Dropout<sup>[15]</sup>

- L2/*weight decay*

L2 atau *weight decay* bekerja dengan menambahkan nilai norm penalti pada *objective function*. Pemberian norm dilakukan hanya terhadap bobot saja karena besarnya bobot mempengaruhi bagaimana interaksi antar fitur. Bobot yang besar akan berpengaruh besar terhadap sedikit perubahan fitur masukan, maka akan diberi penalti terhadap bobot

tersebut. Semakin besar nilai bobot, semakin besar pula penaltinya, begitu pula sebaliknya, jelas Novindasari<sup>[14]</sup>.

$$L(x, y) \equiv \sum_{i=1}^n (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^n \theta_i^2$$

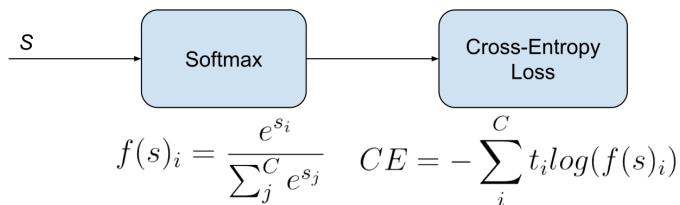
Gambar 2. 14 Penambahan nilai penalti  $L_2$ -regularization pada loss function<sup>[16]</sup>

- *Early Stopping*

Selain penambahan parameter dan dataset, terdapat teknik lain yang lebih sederhana yaitu *early stopping*. *Early stopping* adalah teknik untuk memberhentikan proses *training* ketika perbedaan performa antara *training* dan *validation* sudah melebihi batasan yang ditentukan sehingga diharapkan dapat menghindari terjadinya *overfitting*.

#### 4. Loss Function

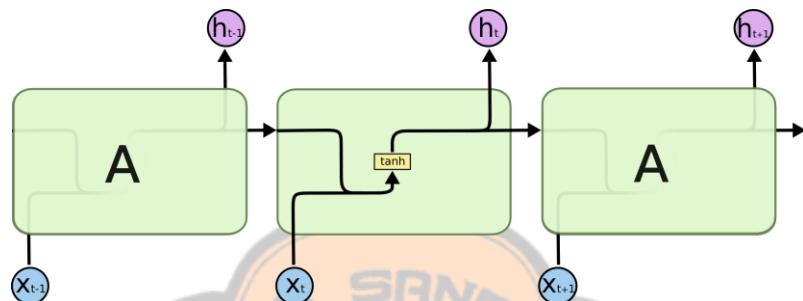
*Loss function* adalah perbedaan antara nilai yang diprediksi oleh model yang dibentuk dan nilai sebenarnya. Fungsi kerugian yang paling umum digunakan di *neural network* adalah *cross-entropy*. Dan untuk permasalahan klasifikasi multikelas, maka *categorical cross-entropy* merupakan salah satu loss function yang sesuai. *Categorical cross-entropy* merupakan gabungan antara fungsi *softmax* dan fungsi *loss cross-entropy*.



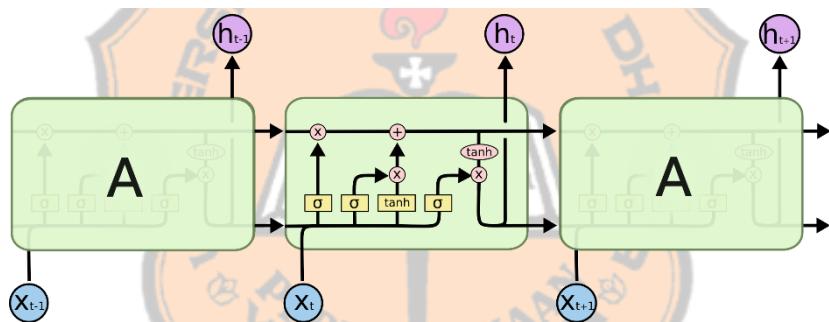
Gambar 2. 15 Categorical Cross-Entropy<sup>[17]</sup>

## 2.5 Long Short-Term Memory (LSTM)

LSTM merupakan pengembangan dari RNN yang memiliki perilaku default yaitu mengingat informasi dalam periode yang lama. Jika pada RNN memiliki struktur yang sederhana yang terdiri dari layer tunggal tanh, pada LSTM terdapat 4 layer yang saling berinteraksi, jelas Colah<sup>[2]</sup>.

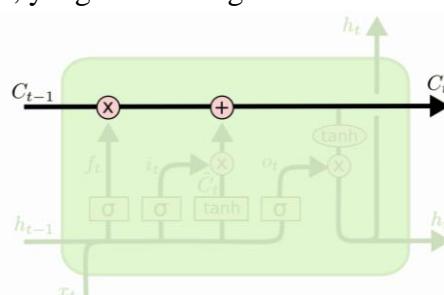


Gambar 2. 16 Modul berulang dalam RNN standar berisi *layer* tunggal<sup>[2]</sup>



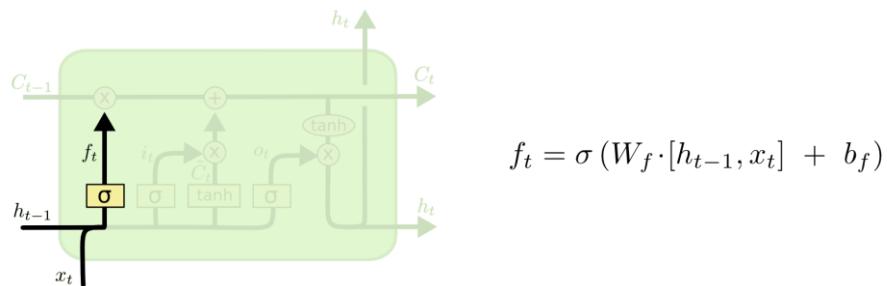
Gambar 2. 17 Modul berulang pada LSTM berisi 4 *layer* saling berinteraksi<sup>[2]</sup>

Kunci utama dari LSTM yaitu adalah cell state karena berjalan lurus ke seluruh rantai, dengan hanya beberapa interaksi linear yang tidak begitu berpengaruh besar. Sangat mudah bagi informasi untuk mengalir begitu saja tanpa perubahan. LSTM memiliki kemampuan untuk menghapus atau menambahkan informasi ke *cell state*, yang diatur dengan cermat oleh struktur yang disebut *gate*.

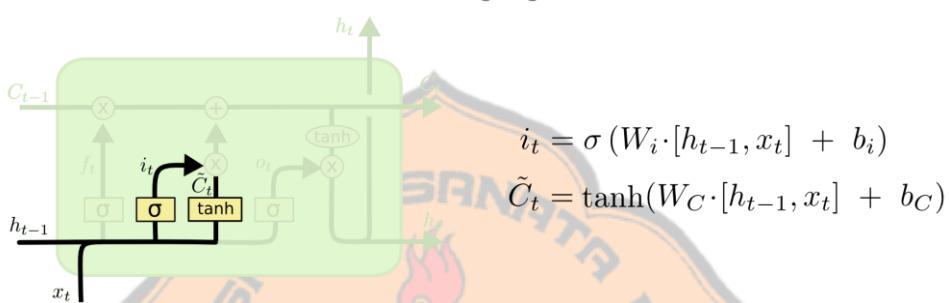


Gambar 2. 18 *cell state* LSTM<sup>[2]</sup>

Alur dari kerja struktur di dalam layer LSTM yaitu seperti pada ilustrasi berikut:

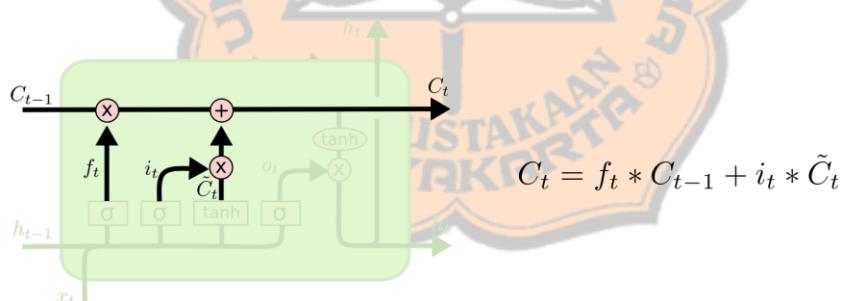


Gambar 2. 19 forget gate LSTM<sup>[2]</sup>



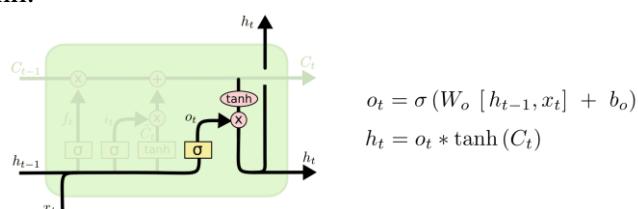
Gambar 2. 20 informasi baru yang diregulasi oleh input<sup>[2]</sup>

Gambar 2.24 bekerja untuk *gate* untuk memilih informasi yang penting dan tanh untuk menetapkan ukuran numerik angka pada vektor tidak mengembung.



Gambar 2. 21 alur mendapatkan nilai cell state yang baru<sup>[2]</sup>

Gambar 2.25 memperlihatkan alur mendapatkan nilai cell state yang baru dari kolaborasi nilai-nilai tahap sebelumnya pada forget gate dan multiplikasi input gate dengan fungsi tanh.



Gambar 2. 22 mekanisme output<sup>[2]</sup>

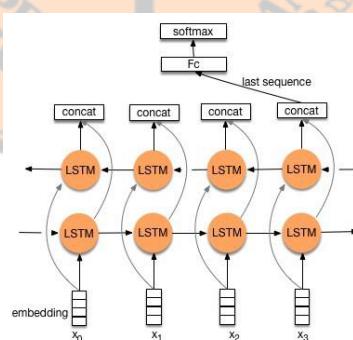
Gambar 2.26 mengilustrasikan mekanisme output yang berasal dari *cell state* namun versi terfilter. *Cell state* tersebut terfilter karena *cell state* yang sudah ditransformasikan ke dalam range [-1,1] oleh *tanh* dimultiplikasikan dengan fungsi *sigmoid* yang tentunya akan menghilangkan beberapa informasi yang dibawa oleh *cell state*.

Alur yang diilustrasikan dari gambar 2.22 s.d 2.26 adalah proses *forward pass*. Proses merupakan setengah dari keseluruhan proses model LSTM terbentuk. Proses selanjutnya yaitu *backward pass* yang di dalamnya terdapat kombinasi antara *backpropagation through time* untuk mendapatkan turunan gradien tiap gate dan *gradient-based descent* untuk memperbarui *weight-weight* lama yang dipengaruhi juga oleh *learning rate*.

## 2.6 Bidirectional Long Short-Term Memory (Bi-LSTM)

Bi-LSTM sama dengan LSTM, hanya saja Bi-LSTM memiliki 2 arah (*forward* dan *backward*). Oleh karena 2 arah tersebut, Bi-LSTM dapat menggunakan informasi secara kontekstual. Output yang dihasilkan pada Bi-LSTM adalah gabungan vektor dari kedua arah, jelas Abdillah<sup>[5]</sup>.

$$ht = ht^{\rightarrow} \oplus ht^{\leftarrow}$$



Gambar 2. 23 Ilustrasi model Bi-LSTM<sup>[18]</sup>

## BAB III

### METODE PENELITIAN

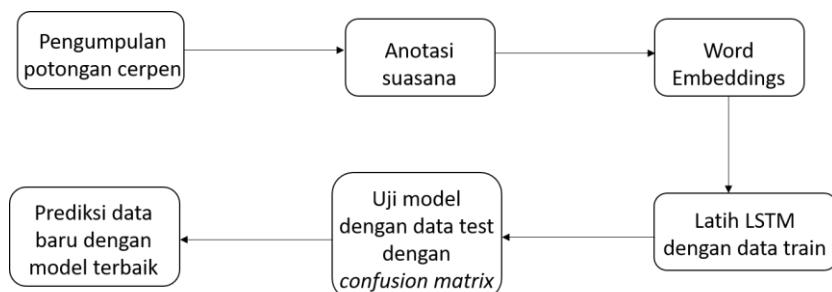
#### 3.1 Bahan dan Alat

Data yang digunakan pada penelitian adalah data hasil *web scraping* di situs-situs publikasi cerita pendek berbahasa Indonesia yang tidak sensitif akan hak cipta. Data berupa potongan cerita seperti paragraf tunggal atau paling sedikit hanya terdiri dari 2 kalimat atau lebih yang membawa suasana. Data direncanakan berjumlah 630 potongan cerita berbentuk paragraf, yang kemudian dibagi untuk *training* dan untuk *testing* berdasar pilihan user dalam menentukan rasio *test-train*. Data hasil *web scraping* tersebut juga disertai dengan proses anotasi suasana sebagai label/kelas target untuk klasifikasi. Pelabelan dilakukan oleh beberapa anotator yang juga merupakan pembaca aktif cerita pendek. Terdapat 6 kelas label yang dituju yaitu Bahagia, Sedih, Jenaka, Menegangkan, Netral, dan Misteri.

Selain dataset cerita pendek, juga digunakan data berbentuk vektor kata-kata yang dikumpulkan dari wiki Indonesia dengan dimensi 300 yang diperoleh dari situs resmi *fastText* dalam membentuk *language model*.

#### 3.2 Tahap-tahap Penelitian

Langkah-langkah penelitian digambarkan pada skema penelitian yang digambarkan pada gambar 3.1 berikut.



Gambar 3. 1 Skema Penelitian

##### 3.2.1 Studi Pustaka

Penulis mempelajari berbagai literatur yang berkaitan dengan

*Recurrent Neural Network* pada jaringan saraf tiruan khususnya mengenai arsitektur *Long Short-Term Memory* yang menggunakan algoritma optimasi penurunan gradien yang dikombinasikan dengan *backpropagation through time* dan literatur lainnya yang digunakan sebagai acuan membangun sistem.

### 3.2.2 Pengumpulan Data

Data diperoleh dari web-scraping situs-situs cerita pendek berbahasa Indonesia yang tidak sensitif terhadap hak cipta yang kemudian tiap potongan cerita akan diberi anotasi suasana oleh beberapa pembaca aktif cerita pendek.

### 3.2.3 Perancangan Alat Uji

Alat uji yang akan dibangun, membutuhkan sebuah *button* untuk menginput dataset potongan cerita yang kemudian akan ditampilkan. Kemudian membutuhkan button untuk melihat akurasi dan loss dari model yang terbentuk dalam melakukan klasifikasi dengan data khusus pemodelan (*train* dan *test*). Kemudian terdapat *field* untuk input potongan cerita baru tidak berlabel dan sebuah *button* untuk menampilkan probabilitas suasana yang terkandung dalam potongan tersebut dan suasana finalnya.

### 3.2.4 Pengolahan Data

Setelah data terkumpul, maka selanjutnya melakukan pengolahan data. Pengolahan data yang dilakukan adalah sebagai berikut:

#### (a) *Preprocessing*

*Preprocessing* adalah proses awal mengolah data teks. Data teks dibersihkan dari kata yang tidak terlalu signifikan artinya (*remove stopwords*) dan tanda baca yang tidak terlalu membawa suasana.

#### (b) Ekstraksi Fitur

Ekstraksi fitur adalah proses pengambilan fitur dari data yang sudah dilakukan *preprocessing*. Pada data teks, sejauh ini terdapat 2 cara

untuk mendapatkan fiturnya yaitu dengan metode *word embedding*.

(c) Klasifikasi

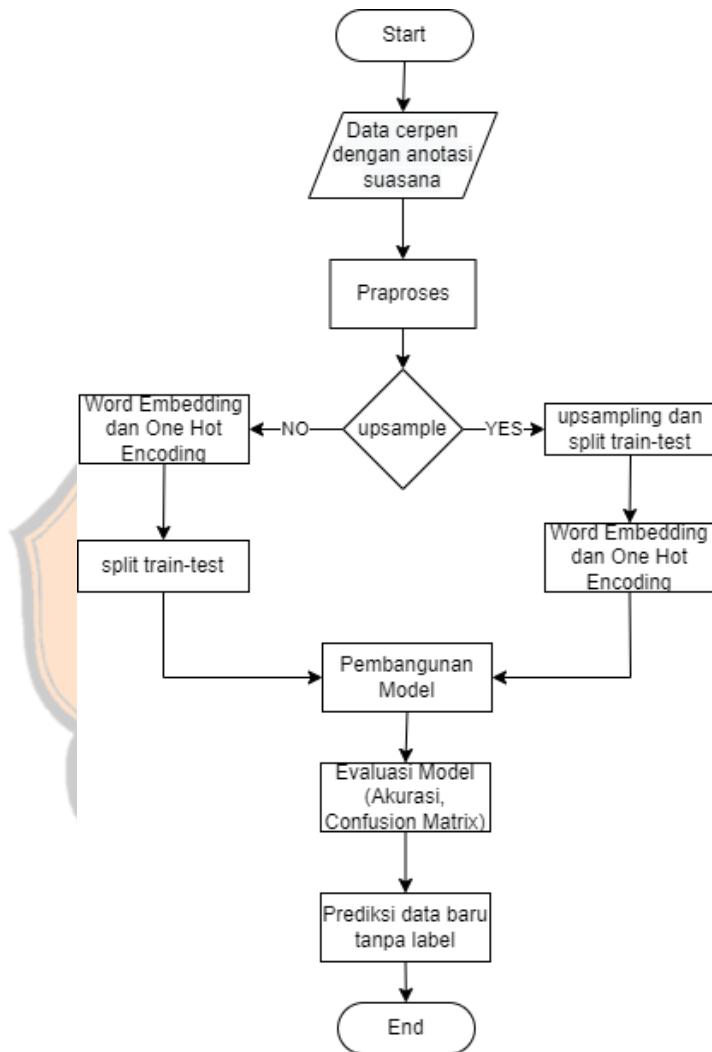
Klasifikasi adalah proses utama dalam penelitian ini untuk mendapatkan suasana dari potongan-potongan cerita. Metode klasifikasi yang digunakan adalah dengan *deep learning* arsitektur LSTM yang dimodifikasi dengan mengkonfigurasi *layer-layer* yang ada. Metode spesifik dalam LSTM adalah gabungan dari *forward pass* dan *backward pass*. Pada *backward pass* terdapat kombinasi antara proses *backpropagation through time* untuk mendapatkan penurunan gradien dan *stochastic gradient descent* untuk menghitung bobot baru berdasarkan penurunan gradien tiap gate yang juga dipengaruhi oleh *learning rate*. Hasil klasifikasi yang diharapkan adalah probabilitas yang didapat untuk tiap suasana bagi sepotong cerita dan juga dicantumkan suasana final yang memiliki nilai probabilitas tertinggi di antara suasana lainnya.

(d) Pengujian

Pengujian adalah proses yang melihat dan mencari performa terbaik dari sistem yang telah dibuat. Dalam melihat akurasi, dilakukan melalui *confusion matrix* dan juga dilihat dari nilai *loss* pada *epoch* terakhir. Data dibagi dengan menerapkan *train-test split* dengan porsi yang akan dijelaskan pada skenario pengujian. Pembagian data dilakukan di dalam sistem yang diatur agar setiap kelas label jumlahnya seimbang (*stratified*).

### 3.3 Desain Alat Uji

Alur atau tahapan proses yang dilakukan pada alat uji dapat dilihat pada Gambar 3.2.



Gambar 3. 2 Tahapan proses alat uji

#### 3.3.1 Preprocessing

Pada penelitian ini, dikarenakan representasi kata menggunakan *fastText* yang dapat menangani imbuhan, maka *stemming* tidak diikutkan. Dan tahap *lemmatization* juga tidak ikut diimplementasikan dikarenakan kurang tersedianya kamus *lemma* berbahasa Indonesia.

Sehingga pada tahap ini dilakukan pembersihan teks seperti *lowercasing*, terhadap kata yang tidak terlalu signifikan (*stopword*

*removal*) dan tanda baca yang tidak terlalu berpengaruh pada pembawaan suasana (*noise removal*), dan memperbaiki adanya kesalahan ketik atau penulisan yang tidak semestinya (contoh: 4L4Y) atau normalisasi.

Berikut adalah langkah dalam melakukan *lowercasing*:

Langkah 1: Gunakan fungsi *apply* untuk *dataframe* kolom potongan cerita

Langkah 2: Deklarasikan fungsi *lambda* yang di dalamnya menggunakan fungsi *lower* sebagai argumen dari fungsi *apply*

Langkah 3: Perbarui *dataframe* kolom potongan cerita dengan hasil terbaru (langkah 2)

Berikut adalah langkah dalam melakukan *remove stopwords*.

Langkah 1: Definisikan list *stopwords* atau menggunakan stopwords dari *nltk* untuk bahasa indonesia.

Langkah 2: Ganti kata yang berada di list *stopwords* dengan karakter kosong “”

Berikut adalah langkah dalam *remove punctuation*.

Langkah 1: Definisikan list *punctuation* yang sekiranya tidak membawa suasana seperti ‘/,;/^/#/)(/’

Langkah 2: Ganti kata yang berada di list *punctuation* dengan karakter kosong “”

Berikut adalah langkah dalam *normalize text*.

Langkah 1: Pisahkan beberapa potongan cerita yang mengandung kata yang perlu dinormalisasi

Langkah 2: Ganti kata tersebut menggunakan *function replace* pada modul string python

### 3.3.2 Transformasi Data

Pada tahap ini dilakukan transformasi tiap potongan cerita yang berbentuk teks menjadi vektor untuk tiap katanya. Tiap kata dapat

mengandung banyak fitur. Pada penelitian ini, digunakan *word embedding FastText* yang merupakan model *pre-trained* untuk meghasilkan 300 fitur per katanya, sehingga lanjutan latih *language model* ini dengan *domain-specific corpus* menggunakan data cerita pendek berbahasa Indonesia yang telah diperoleh juga akan menggunakan format 300 fitur per katanya.

Fitur tersebut didapat dari proses *training deep learning* yang terkandung dalam *fastText*. Cara kerja *fastText* kurang lebih sama dengan *Word2Vec*, hanya saja representasi vektor suatu kata didapat dari jumlah representasi vektor *subword-subword* yang membentuk kata tersebut.

Untuk data target kelas dilakukan transformasi dari berbentuk teks menjadi vektor yang berisi 1 untuk posisi yang benar dan 0 untuk posisi lainnya (*one hot encoding*).

Contoh one hot encoding:

Kelas 1: bahagia

Kelas 2: sedih

Karena ada 2 kelas maka elemen vektor terdiri dari 2, dan angka 1 mengindikasikan kelas tersebut berada di posisi tersebut.

Kelas 1: bahagia => [1 0]

Kelas 2: sedih => [0 1]

### 3.3.3 Pembagian Data Latih dan Uji

Pada tahap ini data dibagi menjadi data latih (*train*) dan data uji (*test*). Porsi train dan test diinput oleh user, yang input tersebut dilakukan dengan memilih button kombinasi test-train yaitu: 30-70, 20-80, 35-65. Pembagian data ini telah diberi parameter ‘*stratified = y*’ untuk tiap varian porsi train-test sehingga diharapkan data yang dibagi dapat seimbang untuk setiap kelasnya.

Pembagian data menggunakan fungsi *train\_test\_split*. Berikut adalah langkah-langkah dalam pembagian data.

Langkah 1: Definisikan variabel X yang menyimpan data fitur tiap kata,

dan  $y$  yang menyimpan data vektor tiap kelas (label).

Langkah 2: Masukkan  $X$ ,  $y$ ,  $test\_size = user\_test$ ,  $stratify = y$ ,  $random\_state = 42$  pada fungsi `train_test_split`.

Langkah 3: Store hasil split dari fungsi `train_test_split` ke variabel  $X\_train$ ,  $X\_test$ ,  $y\_train$ ,  $y\_test$

### 3.3.4 Modeling

Pada tahap ini dilakukan pembentukan model LSTM yang prosesnya yaitu melalui *forward pass*, *backward pass*, lalu memperbarui *weight-weight* dengan metode *gradient-based descent* dan kembali melakukan *forward pass*.

Berikut adalah contoh implementasi pembentukan model LSTM menggunakan data *dummy*.

Data *dummy* terdiri dari 2 data dan setiap data terdiri dari 2 sekuens dan tiap sekuens terdiri dari 2 vektor atau fitur. Kelas target yang ada yaitu berjumlah 2 (bahagia dan sedih).

*One hot encoding* kelas target:

Bahagia	: 1 0
Sedih	: 0 1

Tabel 3.1 Dataset *dummy*

Saya	bergirang	1 0	bahagia
Saya	bahagia	1 0	bahagia

Tabel 3.2 Format data untuk diproses

Data	sekuens	vektor/fitur
1	Saya	[-0.04749205, -0.06248924]
	bergirang	[-0.06357788, -0.02674592]
2	Saya	[-0.04749205, -0.06248924]
	bahagia	[0.04314232, -0.05018727]

Tabel 3.2 menunjukkan format data yang dapat diproses oleh

LSTM. Data biasa perlu dimodifikasi menjadi format di atas karena *input shape* dari LSTM atau metode lainnya di bawah Recurrent Neural Network adalah 3 (data, sekuens, fitur). Karena fokus dari penelitian ini adalah penerapan klasifikasinya, maka fitur didapatkan dari mencari langsung nilai *word vector* untuk tiap kata dengan *fastText* yang diimplementasikan dengan *gensim* yang tidak menggunakan model bahasa *pre-trained* tapi berdasarkan 2 data yang ada.

Selanjutnya data *dummy* ini akan melalui proses *forward pass* di epoch 1 yang terdiri dari 2 iterasi karena pada data, maksimal sekuens berjumlah 2. Setiap iterasi melakukan perhitungan *network* dan memperbarui *cell state* dan *hidden state* untuk tiap sekuens/kata. Setelah melakukan *forward pass* sebanyak 2 iterasi, maka akan dilanjutkan dengan melakukan prediksi dan melihat perfoma akurasi dan juga nilai loss.

EPOCH 1

ITERASI 1

*Weight* dan *bias* untuk tiap *gate* diinisialisasi dengan nilai random. *Cell state* awal dan *hidden state* awal diinisialisasikan dengan 0. Data yang digunakan pada epoch 1 adalah data pertama.

$W =$	<table border="1"> <tr><td>wi1</td><td>wi2</td><td>bi</td></tr> <tr><td>wc1</td><td>wc2</td><td>bc</td></tr> <tr><td>wf1</td><td>wf2</td><td>bf</td></tr> <tr><td>wo1</td><td>wo2</td><td>bo</td></tr> </table>	wi1	wi2	bi	wc1	wc2	bc	wf1	wf2	bf	wo1	wo2	bo	=	<table border="1"> <tr><td>0.5</td><td>0.25</td><td>0.01</td></tr> <tr><td>0.3</td><td>0.4</td><td>0.05</td></tr> <tr><td>0.03</td><td>0.06</td><td>0.002</td></tr> <tr><td>0.02</td><td>0.04</td><td>0.001</td></tr> </table>	0.5	0.25	0.01	0.3	0.4	0.05	0.03	0.06	0.002	0.02	0.04	0.001
wi1	wi2	bi																									
wc1	wc2	bc																									
wf1	wf2	bf																									
wo1	wo2	bo																									
0.5	0.25	0.01																									
0.3	0.4	0.05																									
0.03	0.06	0.002																									
0.02	0.04	0.001																									

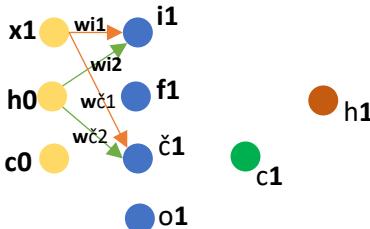
Tabel 3. 3 *Weight kernel epoch 1 iterasi 1*

$h_0$  dan  $c_0$ :

x1	x2	h0	c0
-0.04749	-0.06358	0	0
-0.06249	-0.02675	0	0

Tabel 3. 4

Forward Pass:  
Input gate:



$$(1) \text{ neti1} = (w_{i1}*x_1 + w_{i2}*h_0 + b_i) \\ i_1 = \sigma(\text{neti1}) = \dots$$

$$(2) \text{ netc1} = (w_{c1}*x_1 + w_{c2}*h_0 + b_c) \\ \check{c}_1 = \tanh(\text{netc1}) = \dots$$

Gambar 3. 3

$x_1$	$\text{neti1}$	$i_1$	$i_1$ decision
-0.04749	-0.01375	0.496564	1
-0.06249	-0.02124	0.494689	1

Tabel 3. 5

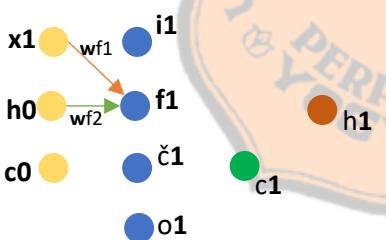
$$r = 0.34$$

bil random hasil generate < sigmoid, maka 1 (informasi diteruskan)

$x_1$	$\text{netc1}$	$\check{c}_1/\tanh(\text{netc1})$
-0.04749	0.035752	0.035737
-0.06249	0.031253	0.031243

Tabel 3. 6

Forget gate:



$$(3) \text{ netf1} = (wf_1*x_1 + wf_2*h_0 + bf) \\ f_1 = \sigma(\text{netf1}) = \dots$$

$x_1$	$\text{netf1}$	$f_1$	$f_1$ decision
-0.04749	0.000575	0.500144	1
-0.06249	0.000125	0.500031	1

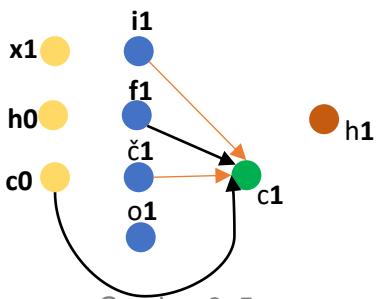
Tabel 3. 7

Gambar 3. 4

$$r = 0.438$$

bil random hasil generate < sigmoid, maka 1 (informasi diteruskan)

Memory cell:



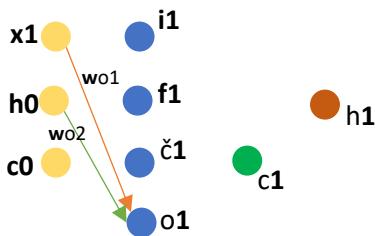
$$(4) \text{ update cell} = c_1 = i_1 * \check{c}_1 + f_1 * c_0$$

$x_1$	$c_1$
-0.04749	0.035737
-0.06249	0.031243

Tabel 3. 8

Gambar 3. 5

**Output gate:**



Gambar 3. 6

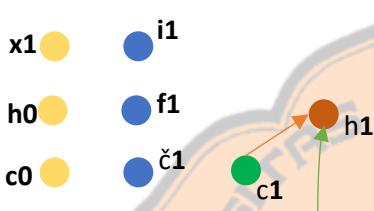
$$(5) \text{ neto1} = (w_{01} * x_1 + w_{02} * h_0 + b_0) \\ o_1 = \sigma(\text{neto1}) = \dots$$

x1	neto1	o1	o1 decision
-0.04749	0.000050159	0.500013	1
-0.06249	-0.00025	0.499938	1

Tabel 3. 9

bil random hasil generate < sigmoid,  
maka 1 (informasi diteruskan)

**Hidden layer/hidden state:**



Gambar 3. 7

$$(6) h_1 = o_1 * \tanh(c_1)$$

x1	h1
-0.04749	0.035722
-0.06249	0.031233

Tabel 3. 10

**ITERASI 2**

c1 dan h1 yang diperoleh pada iterasi 1, akan digunakan sebagai bahan input di iterasi 2, dan weight yang digunakan sama dengan weight pada iterasi 1.

Tabel 3. 11 Weight kernel epoch 1 iterasi 2

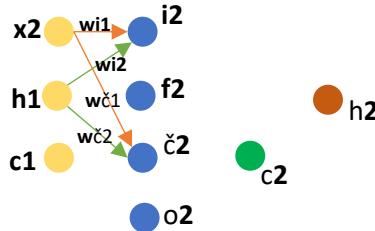
W=	wi1	wi2	bi	=	0.5	0.25	0.01
	wc1	wc2	bc		0.3	0.4	0.05
	wf1	wf2	bf		0.03	0.06	0.002
	wo1	wo2	bo		0.02	0.04	0.001

**h1 dan c1:**

x1	x2	h1	c1
-0.04749	-0.06358	0.035722	0.035737
-0.06249	-0.02675	0.031233	0.031243

Tabel 3. 12

**Input gate:**



$$(7) \text{ neti2} = (wi1*x2 + wi2*h1 + bi) \\ i2 = \sigma(\text{neti2}) = \dots$$

$$(8) \text{ netc2} = (wc1*x2 + wc2*h1 + bc) \\ \check{c}2 = \tanh(\text{netc2}) = \dots$$

Gambar 3. 8

x2	neti2	i2	i2 decision
-0.06358	-0.01286	0.496785	1
-0.02675	0.004435	0.501109	1

Tabel 3. 13

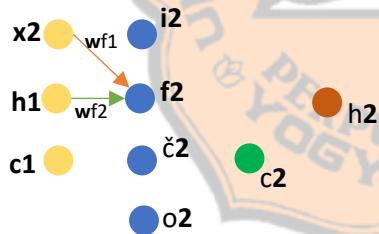
$r = 0.34$

bil random hasil generate < sigmoid, maka 1 (informasi diteruskan)

x2	netc2	č2/tanh(netc2)
-0.06358	0.045215	0.045185
-0.02675	0.054469	0.054416

Tabel 3. 14

**Forget gate:**



$$(9) \text{ netf2} = (wf1*x2 + wf2*h1 + bf) \\ f2 = \sigma(\text{netf2}) = \dots$$

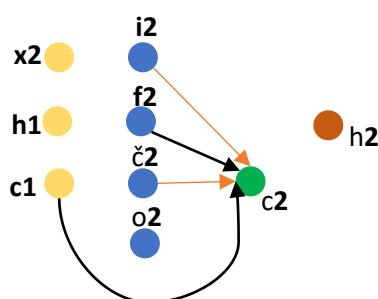
x2	netf2	f2	f2 decision
-0.06358	0.00223	0.50055	
-0.02675	0.00307	0.50076	1

Tabel 3. 15

$r = 0.438$

bil random hasil generate < sigmoid, maka 1 (informasi diteruskan)

**Memory cell:**

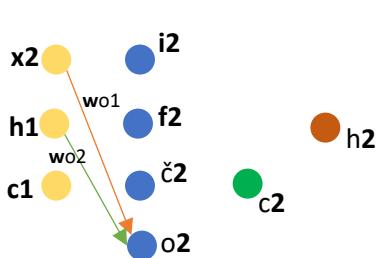


$$(10) \text{ update cell} = c2 = i2 * \check{c}2 + f2 * c1$$

x2	c2
-0.06358	0.080922
-0.02675	0.085659

Tabel 3. 16

**Output gate:**



Gambar 3. 11

$$(11) \text{ neto2} = (w_01 * x_2 + w_02 * h_1 + b_0) \\ o_2 = \sigma(\text{neto2}) = \dots$$

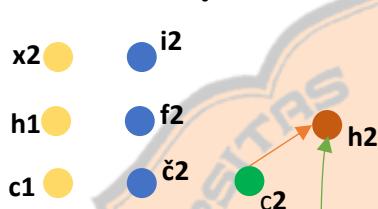
x2	neto2	o2	o2 decision
-0.06358 7	0.00115 7	0.50028 9	1
-0.02675 4	0.00171 4	0.50042 9	1

Tabel 3. 17

$$r = 0.027$$

bil random hasil generate < sigmoid,  
maka 1 (informasi diteruskan)

**Hidden layer/hidden state:**



Gambar 3. 12

$$(12) h_2 = o_2 * \tanh(c_2)$$

x2	h2
-0.06358	0.080746
-0.02675	0.08545

Tabel 3. 18

Iterasi 2 pada epoch 1 telah dilakukan, maka selanjutnya adalah tahap memprediksi dan melihat akurasi dan loss.

$$(13) \text{ Prediksi} = \text{softmax}(\text{output})$$

$$\text{Softmax output target 1} = \frac{e^{h[0]}}{e^{h[0]} + e^{h[1]}} \dots \dots \dots \text{(pers.1)}$$

$$\text{Softmax output target 2} = \frac{e^{h[1]}}{e^{h[0]} + e^{h[1]}} \dots \dots \dots \text{(pers.2)}$$

$$(14) \text{ Categorical Cross Entropy} =$$

$$-((y[0] \times \ln \tilde{y}[0]) + (y[1] \times \ln \tilde{y}[1])) \dots \dots \dots \text{(pers.3)}$$

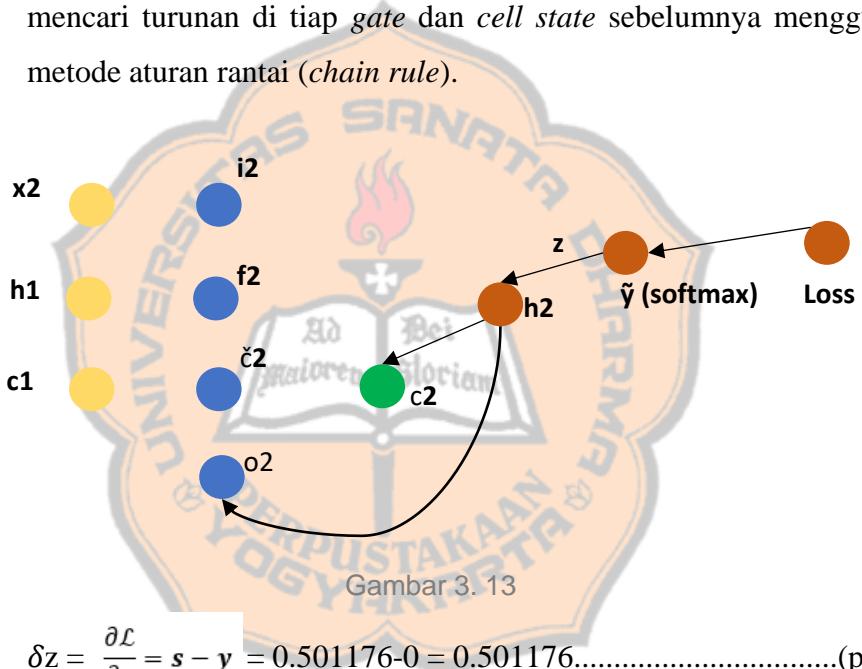
h2	e(h)	softmax/ $\tilde{y}$	y	ypred (nilai max softmax)	akurasi	loss (Categorical Cross Entropy)
0.080746	1.084095	0.498824	1	0	0	0.695502
0.08545	1.089207	0.501176	0	1		
			bahagia	sedih		

Tabel 3. 19 Prediksi, akurasi, loss epoch 1

Selanjutnya adalah melakukan *backward pass* yang di dalamnya terdapat metode *backpropagation through time* untuk mendapatkan turunan gradien tiap *gate* dan juga metode *gradient-descent based* untuk memperbarui *weight*.

### Backward Pass:

Jika dilihat dari ilustrasi tiap *gate*, maka kunci dari dapatnya *output* ada di  $c_2$  dan  $o_2$ . Oleh karena itu, maka dicari turunan dari jalur *loss* ke 2 *node* tersebut. Dari 2 *node* tersebut maka dapat dilanjutkan untuk mencari turunan di tiap *gate* dan *cell state* sebelumnya menggunakan metode aturan rantai (*chain rule*).



$$- \quad \delta z = \frac{\partial \mathcal{L}}{\partial z} = s - y = 0.501176 - 0 = 0.501176 \dots \dots \dots \text{(pers.4)}$$

Karena hanya terdapat 1 hasil prediksi final, maka yang digunakan hanyalah yang memiliki probabilitas tertinggi (karena 2 output tersebut akan menghasilkan  $\delta z$  yang sama).

$$- \quad \delta h_t = \frac{\partial z}{\partial h_t} = e^{ht} \dots \dots \dots \text{(pers.5)}$$

- Gradien descent terhadap output ( $\delta_{\text{ot}}$ )

$$\delta_{\text{ot}} = \delta z * \exp(ht) * \tanh(ct) \dots \text{(pers.6)}$$

$\delta o1$	$\delta o2$
0.018554	0.043871
0.01615	0.046646

Tabel 3. 20

- Gradien descent terhadap cell state( $\delta_{\text{ct}}$ )

$$\delta_{\text{ct}} = \delta z * \exp(h) * \text{ot} * (1 - \tanh^2(ct)) \dots \text{(pers.7)}$$

$\delta c1$	$\delta c2$
0.51874	0.53978
0.516572	0.541898

Tabel 3. 21

- Input gate( $\delta_{\text{it}}$ )

$$\delta_{\text{it}} = \delta_{\text{ct}} * \dot{c}_t \dots \text{(pers.8)}$$

$\delta i1$	$\delta i2$
0.018538	0.02439
0.016139	0.029488

Tabel 3. 22

- Forget gate( $\delta_{\text{ft}}$ )

$$\delta_{\text{ft}} = \delta_{\text{ct}} * c_{t-1} \dots \text{(pers.9)}$$

$\delta f1$	$\delta f2$
0	0.01929
0	0.016931

Tabel 3. 23

- Ajuan memory state baru( $\delta_{\text{at}}$ )

$$\delta_{\text{at}} = \delta_{\text{ct}} * i_t \dots \text{(pers.10)}$$

$\delta a1$	$\delta a2$
0.51874	0.53978
0.516572	0.541898

Tabel 3. 24

- Cell state sebelumnya( $\delta_{\text{ct}-1}$ )

$$\delta_{\text{ct}-1} = \delta_{\text{ct}} * f_t \dots \text{(pers.11)}$$

$\delta c0$	$\delta c1$
0.51874	0.53978
0.516572	0.541898

Tabel 3. 25

- Input ke pengajuan( $\delta_{\hat{a}t}$ )

$$\delta_{\hat{a}t} = \delta_{\text{at}} * (1 - \tanh^2(\text{net}_{\text{ct}})) \dots \text{(pers.12)}$$

$\delta\hat{a}1$	$\delta\hat{a}2$
0.518077	0.538678
0.516068	0.540294

Tabel 3. 26

- Net input ke input gate( $\delta\hat{i}$ )

$$\delta\hat{a} = \delta\hat{i}t^*it^*(1-it) \dots \text{(pers.13)}$$

$\delta\hat{i}1$	$\delta\hat{i}2$
0	0
0	0

Tabel 3. 27

- Net input ke forget gate( $\delta\hat{f}$ )

$$\delta\hat{f} = \delta\hat{f}t^*ft^*(1-ft) \dots \text{(pers.14)}$$

$\delta\hat{f}1$	$\delta\hat{f}2$
0	0
0	0

Tabel 3. 28

- Net input ke output gate( $\delta\hat{o}$ )

$$\delta\hat{o} = \delta\hat{o}t^*ot^*(1-ot) \dots \text{(pers.15)}$$

$\delta\hat{o}1$	$\delta\hat{o}2$
0	0
0	0

Tabel 3. 29

- $It = [xt, ht-1, 1] \dots \text{(pers.16)}$

	$t=1$	$t=2$
$xt$	-0.04749	-0.06358
	-0.06249	-0.02675
$ht-1$	0	0.035722
	0	0.031233
$1$	1	1
	1	1

Tabel 3. 30

- $[It]^T =$

	$xt$		$ht-1$		$1$	
$t=1$	-0.04749	-0.06249	0	0	1	1
$t=2$	-0.06358	-0.02675	0.035722	0.031233	1	1

Tabel 3. 31

- $\delta zt = [\delta \hat{t}, \delta \hat{a}, \delta \hat{f}, \delta \hat{o}]$  .....(pers.17)

	<b>t=1</b>	<b>t=2</b>
<b><math>\delta \hat{t}</math></b>	0	0
	0	0
<b><math>\delta \hat{a}</math></b>	0.518077	0.538678
	0.516068	0.540294
<b><math>\delta \hat{f}</math></b>	0	0
	0	0
<b><math>\delta \hat{o}</math></b>	0	0
	0	0

Tabel 3. 32

$$- \quad \delta Wt = \frac{\partial E}{\partial Wt} = \delta zt \times [It]^T \quad \text{(pers.18)}$$

Untuk  $[It]^T$  dan  $\delta zt$  masih dalam bentuk  $2 \times 1$  untuk tiap value per sekuens.

Oleh karena itu dibutuhkan proses menjadikannya  $1 \times 1$  per value per sekuens dengan cara merata-rata tiap value dari tiap sekuens tersebut. Maka hasil dari  $[It]^T$  dan  $\delta zt$  yang sudah berbentuk  $1 \times 1$  untuk tiap value dan sekuensnya adalah sebagai berikut.

$\delta zt =$	0	0
	0.517072452	0.539486
	0	0
	0	0

Tabel 3. 33

$[It]^T =$	-0.05499	0	1
	-0.04516	0.033477	1

Tabel 3. 34

$\delta Wt =$	0	0	0
	-0.0528	0.018061	1.056558
	0	0	0
	0	0	0

Tabel 3. 35

Setelah didapatkan nilai  $\delta Wt$ , maka tahap selanjutnya adalah memperbarui *weight* lama dengan metode *gradient-descent based*.

$$W_{baru} = W_{lama} - (\alpha * \delta Wt), \alpha = 0.5 \quad \text{(pers.19)}$$

W baru =

0.5	0.25	0.01
0.326399	0.39097	-0.47828
0.03	0.06	0.002
0.02	0.04	0.001

Tabel 3. 36

Nilai  $\alpha$  berada di range 0 sampai 1, dan pada implementasi kali ini diambil nilai  $\alpha = 0.5$ . Setelah mendapat nilai weight baru, maka weight ini dapat digunakan untuk *feed-forward-pass* pada epoch selanjutnya, epoch 2.

## EPOCH 2

### ITERASI 1

*Weight* dan *bias* didapat dari proses backward pass di epoch sebelumnya. *Cell state* awal dan *hidden state* awal diinisialisasikan dengan 0. Data yang digunakan pada epoch 2 adalah data selanjutnya yaitu data 2.

W=

wi1	wi2	Bi	=	0.5	0.25	0.01
wc1	wc2	Bc		0.326399	0.39097	-0.47828
wf1	wf2	Bf		0.03	0.06	0.002
wo1	wo2	Bo		0.02	0.04	0.001

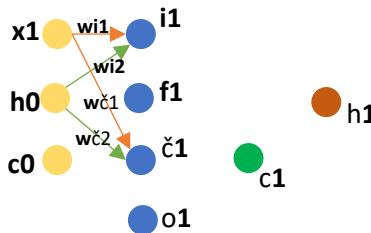
Tabel 3. 37 Weight kernel epoch 2 iterasi 1

h0 dan c0:

x1	x2	h0	c0
-0.04749	0.043142	0	0
-0.06249	-0.05019	0	0

Tabel 3. 38

Forward Pass:  
Input gate:



$$(1) \text{ neti1} = (wi1*x1 + wi2*h0 + bi) \\ i1 = \sigma(\text{neti1}) = \dots$$

$$(2) \text{ netc1} = (wc1*x1 + (wc2*h0) + bc) \\ c1 = \tanh(\text{netc1}) = \dots$$

Gambar 3. 14

x1	neti1	i1	i1 decision
-0.04749	-0.01375	0.496564	1
-0.06249	-0.02124	0.494689	1

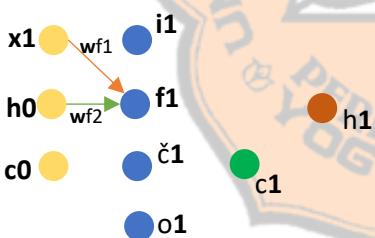
Tabel 3. 39

$r = 0.34$   
bil random hasil generate < sigmoid, maka 1 (informasi diteruskan)

x1	netc1	c̄1/tanh(netc1)
-0.04749	-0.49378	-0.45721
-0.06249	-0.49868	-0.46107

Tabel 3. 40

Forget gate:



$$(3) \text{ netf1} = (wf1*x1 + wf2*h0 + bf) \\ f1 = \sigma(\text{netf1}) = \dots$$

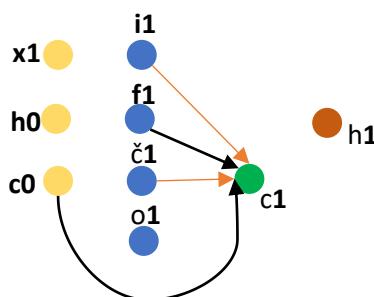
x1	netf1	f1	f1 decision
-0.04749	0.000575	0.500144	1
-0.06249	0.000125	0.500031	1

Tabel 3. 41

Gambar 3. 15

$r = 0.438$   
bil random hasil generate < sigmoid, maka (informasi diteruskan)

Memory cell:



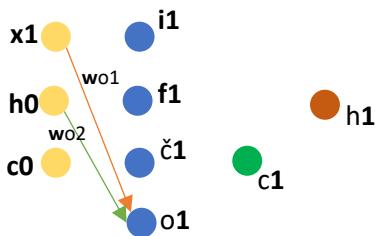
$$(4) \text{ update cell} = c1 = i1*c̄1 + f1*c0$$

x1	c1
-0.04749	-0.45721
-0.06249	-0.46107

Tabel 3. 42

Gambar 3. 16

**Output gate:**



Gambar 3. 17

$$(5) \text{ neto1} = (w_{01} * x_1 + w_{02} * h_0 + b_0) \\ o_1 = \sigma(\text{neto1}) = \dots$$

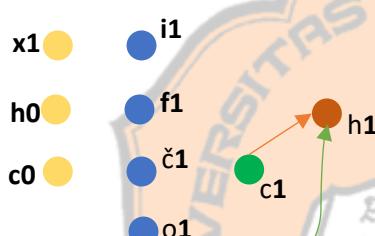
x1	neto1	o1	o1 decision
-0.04749 59	0.0000501 59	0.50001 3	1
-0.06249 8	-0.00025 8	0.49993 8	1

Tabel 3. 43

$$r = 0.027$$

bil random hasil generate < sigmoid,  
maka 1 (informasi diteruskan)

**Hidden layer/hidden state:**



$$(6) h_1 = o_1 * \tanh(c_1)$$

x1	h1
-0.04749	-0.42781
-0.06249	-0.43096

Tabel 3. 44

**ITERASI 2**

c1 dan h1 yang diperoleh pada iterasi 1, akan digunakan sebagai bahan input di iterasi 2, dan weight yang digunakan sama dengan weight pada iterasi 1.

Tabel 3. 45 Weight kernel epoch 2 iterasi 2

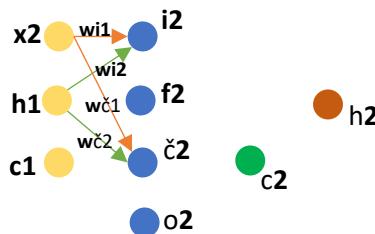
W=	w <sub>i1</sub>	w <sub>i2</sub>	b <sub>i</sub>	=	0.5	0.25	0.01
	w <sub>c1</sub>	w <sub>c2</sub>	b <sub>c</sub>		0.326399	0.39097	-0.47828
	w <sub>f1</sub>	w <sub>f2</sub>	B <sub>f</sub>		0.03	0.06	0.002
	w <sub>o1</sub>	w <sub>o2</sub>	B <sub>o</sub>		0.02	0.04	0.001

**h1 dan c1:**

x1	x2	h1	c1
-0.04749	0.043142	-0.42781	-0.45721
-0.06249	-0.05019	-0.43096	-0.46107

Tabel 3. 46

**Input gate:**



$$(7) \text{ neti2} = (wi1*x2 + wi2*h1 + bi) \\ i2 = \sigma(\text{neti2}) = \dots$$

$$(8) \text{ netc2} = (wc1*x2 + (wc2*h1) + bc) \\ \check{c}2 = \tanh(\text{netc2}) = \dots$$

Gambar 3. 18

x2	neti2	i2	i2 decision
0.043142	-0.075381112	0.481164	1
-0.05019	-0.122833622	0.46933	1

Tabel 3. 47

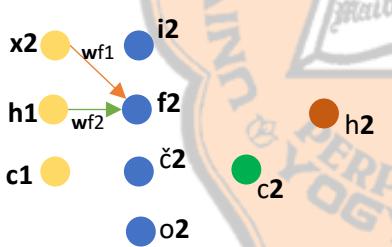
$r = 0.34$

bil random hasil generate < sigmoid, maka 1 (informasi diteruskan)

x2	netc2	č2/tanh(netc2)
0.043142	-0.631457984	-0.55906
-0.05019	-0.663152577	-0.58046

Tabel 3. 48

**Forget gate:**



$$(9) \text{ netf2} = (wf1*x2 + wf2*h1 + bf) \\ f2 = \sigma(\text{netf2}) = \dots$$

x2	netf2	f2	f2 decision
0.043142	-0.022374276	0.494407	1
-0.05019	-0.025363215	0.49366	1

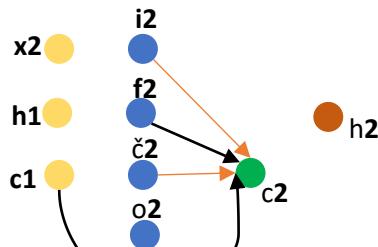
Tabel 3. 49

Gambar 3. 19

$r = 0.438$

bil random hasil generate < sigmoid, maka 1  
(informasi diteruskan)

**Memory cell:**



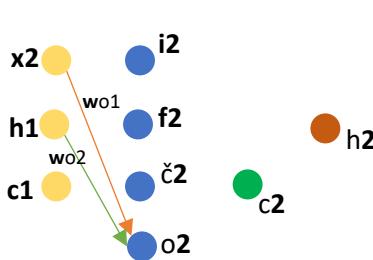
$$(10) \text{ update cell} = c2 = i2*\check{c}2 + f2*c1$$

x2	c2
0.043142	-1.016267205
-0.05019	-1.041532601

Tabel 3. 50

Gambar 3. 20

**Output gate:**



Gambar 3. 21

$$(11) \text{ neto2} = (w_01 * x_2 + w_02 * h_1 + b_0) \\ o_2 = \sigma(\text{neto2}) = \dots$$

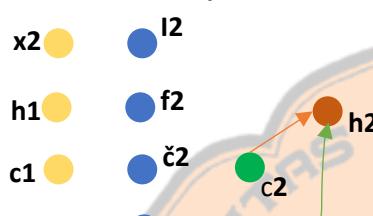
x2	neto2	o2	o2 decision
0.043142	-0.015249517	0.496188	1
-0.05019	-0.017242143	0.49569	1

Tabel 3. 51

$$r = 0.027$$

bil random hasil generate < sigmoid,  
maka 1 (informasi diteruskan)

**Hidden layer/hidden state:**



Gambar 3. 22

$$(12) h_2 = o_2 * \tanh(c_2)$$

x2	h2
0.043142	-0.768341773
-0.05019	-0.778492554

Tabel 3. 52

Iterasi 2 pada epoch 1 telah dilakukan, maka selanjutnya adalah tahap memprediksi dan melihat akurasi dan loss.

$$(13) \text{ Prediksi} = \text{softmax}(\text{output})$$

$$\text{Softmax output target 1} = \frac{e^{h[0]}}{e^{h[0]} + e^{h[1]}} \dots \dots \dots \text{(pers.1)}$$

$$\text{Softmax output target 2} = \frac{e^{h[1]}}{e^{h[0]} + e^{h[1]}} \dots \dots \dots \text{(pers.2)}$$

$$(14) \text{ Categorical Cross Entropy} =$$

$$-((y[0] \times \ln \tilde{y}[0]) + (y[1] \times \ln \tilde{y}[1])) \dots \dots \dots \text{(pers.3)}$$

Tabel 3. 53 Prediksi, akurasi, loss epoch 2

h2	e(h)	softmax/ $\tilde{y}$	Y	y <sub>pred</sub> (nilai max softmax)	akurasi	loss (Categorical Cross Entropy)
-0.76834	0.463781	0.502538	1	1	$\frac{1}{2} = 0.5$	0.688085
-0.77849	0.459098	0.497462	0	0		
				bahagia	bahagia	

Hasil yang didapatkan yaitu pada *epoch* 2 telah terjadi perbaikan pada model. Akurasi yang sebelumnya 0 pada *epoch* 1 menjadi 0.5 pada *epoch* 2 dan nilai *loss* yang sebelumnya 0.695502 pada *epoch* 1 menjadi 0.688085 pada *epoch* 2.

### 3.3.5 Akurasi

Pada tahap ini dilakukan perhitungan akurasi menggunakan *confusion matrix*. Akurasi diperoleh dengan menjumlah data yang diprediksi benar dibagi total melakukan prediksi yang kemudian dibuat dalam bentuk persentase.

*Confusion matrix* didapatkan dengan menggunakan fungsi `confusion_matrix` dengan parameter `y_true` yaitu label kelas yang sebenarnya dan `y_pred` yaitu label kelas hasil prediksi.

## 3.4 Desain Antar Muka

### 3.4.1 Halaman Awal



Gambar 3. 23 GUI Halaman Awal

Pada halaman awal terdapat 2 tombol. Tombol pertama yaitu “bentuk model dan uji” untuk mengarah ke halaman yang berfokus kepada membentuk model dan pengujinya (akurasi dan loss) terdapat di halaman tersebut. Tombol kedua yaitu “Uji Tunggal dengan model

terpilih” untuk mengarah ke halaman yang berfokus kepada memprediksi data tanpa label menggunakan model terpilih yang sudah dilatih sebelumnya.

### 3.4.2 Halaman Bentuk Model dan Uji

The screenshot shows a user interface titled "Halaman bentuk model dan uji". On the left, there's a section for "Import dataset" with a table for "Potongan cerita" and "Label". Below it, a dropdown menu for "Pilih rasio split test-train" offers options: 20-80, 30-70, and 35-65. A "Buat model" button is at the bottom of this section. In the center, a table titled "Hasil pembuatan model" compares "Model 1" and "Model 2" across "Akurasi" and "Loss". Two buttons, "Lihat Prediksi Model 1" and "Lihat Prediksi Model 2", are positioned below the table. To the right, a table titled "Prediksi dengan model ke-N" shows probabilities for five categories (P1-P5) and a final prediction ("Pred final") for each entry in the "Potongan cerita" column.

Gambar 3. 24 GUI Halaman Bentuk Model dan Uji

Pada halaman ini, terdapat 2 *section*. *Section* pertama yaitu untuk menginput dataset dan input rasio *split test-train* dan tombol “Buat model”. Pada input rasio *split test-train* berupa *radio button* sehingga hanya dapat dipilih salah satunya. *Section* kedua yaitu hasil uji berupa performa akurasi dan loss untuk tiap model dan terdapat 2 tombol untuk melihat hasil prediksi tiap model. Ketika salah 1 tombol diklik, maka tombol lainnya diatur *disabled*. Hasil prediksi yang ditampilkan adalah probabilitas tiap suasana, p1 untuk suasana bahagia, p2 untuk suasana sedih, p3 untuk suasana jenaka, p4 untuk suasana menegangkan, p5 untuk suasana netral. Kemudian terdapat juga kolom prediksi final yaitu berupa suasana yang memiliki probabilitas tinggi di antara lainnya.

### 3.4.3 Halaman Uji Tunggal

**Uji Tunggal dengan model terpilih**

<b>Input potongan cerita</b> <input style="width: 100%; height: 100px; margin-bottom: 5px;" type="text"/> atau <input style="width: 100%; height: 30px; margin-bottom: 5px;" type="button" value="Import dataset"/> <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;"> <b>Potongan cerita</b>  </div>	<b>Prediksi dengan model terpilih</b> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Potongan cerita</th> <th style="width: 15%;">P1</th> <th style="width: 15%;">P2</th> <th style="width: 15%;">P3</th> <th style="width: 15%;">P4</th> <th style="width: 15%;">P5</th> <th style="width: 15%;">Pred final</th> </tr> </thead> <tbody> <tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table> <div style="margin-top: 10px;"> <input style="width: 100%; height: 30px;" type="button" value="Prediksi"/>  <input style="width: 100%; height: 30px;" type="button" value="Reset input"/> </div>	Potongan cerita	P1	P2	P3	P4	P5	Pred final																																			
Potongan cerita	P1	P2	P3	P4	P5	Pred final																																					

Gambar 3. 25 GUI Halaman Uji Tunqal

Pada halaman uji tunggal terdapat *textfield* untuk menginput potongan cerita tunggal, tombol untuk mengimpor dataset potongan cerita tak berlabel, tabel untuk memperlihatkan hasil impor dataset potongan cerita tak berlabel, tombol “Prediksi”, tombol “Reset input”, dan tabel yang memperlihatkan hasil prediksi dengan probabilitas tiap suasannya. Ketika belum ada data yang terinput, maka tombol “Prediksi” dan “Reset input” diatur *disabled*. Ketika *textfield* sebelumnya terisi namun setelahnya dipilih tombol “Import dataset”, maka tulisan pada *textfield* otomatis akan terhapus. Jika ada hasil impor dataset, maka yang diprediksi adalah hasil impor bukan yang berada di *textfield*. Untuk memprediksi potongan cerita yang terdapat di *textfield* ketika hasil impor dataset sudah data, maka dapat memilih tombol “Reset input” dan kembali menginput di bagian *textfield*.

## 3.5 Skenario pengujian

Pengujian akan dilakukan dengan menguji beberapa jenis kombinasi antara arsitektur dan nilai dari beberapa *hyperparameter* untuk masing-masing jenis arsitektur. Detail dari skenario pengujian dapat dilihat pada tabel-tabel berikut.

## 1. Sederhana

Pada arsitektur sederhana hanya terdiri dari *dropout layer*, *LSTM/Bidirectional layer*, dan *dense layer* di akhir sebagai *output*. <max\_len>, <unit\_sel>, <jumlah\_kelas> berturut-turut adalah nilai panjang maksimal teks dari keseluruhan data potongan cerita, nilai yang mengindikasikan jumlah sel pada *LSTM/Bidirectional layer*, dan nilai yang mengarah kepada jumlah kelas target pada *dataset* yang nilai jelasnya nanti akan mengikuti pada saat implementasi.

Selain itu juga terdapat pengujian beberapa *value* dari beberapa *hyperparameter* seperti yang tertampil pada Tabel 55.

Tabel 3. 54 Skenario Arsitektur Sederhana

<b>Layer (type) – LSTM</b>	<b>Layer (type) – Bi-LSTM</b>	<b>Output Shape</b>	<b>Param</b>
dropout_1 (Dropout)	dropout_1 (Dropout)	(None, <max_len>, 300)	Menyesuaikan saat implementasi
LSTM_1 (LSTM)	bidirectional_1 (Bidirectional)	(None, <unit_sel>)	Menyesuaikan saat implementasi
dropout_2 (Dropout)	dropout_2 (Dropout)	(None, <unit_sel>)	Menyesuaikan saat implementasi
dense_1 (Dense)	dense_1 (Dense)	(None, <jumlah_kelas>)	Menyesuaikan saat implementasi

Tabel 3. 55 Skenario *hyperparameter* sederhana

<b>No</b>	<b>Hyperparameter</b>	<b>Nilai/value</b>
1	<i>Unit neuron</i>	64, 128, 256
2	<i>Dropout layer rate</i>	0.2 – 0.6
3	<i>Batch size</i>	64, 128, 256
4	<i>Optimizer</i>	<i>adam, rmsprop, sgd</i>

## 2. Kompleks 1

Arsitektur kompleks 1 memiliki struktur yang hampir sama dengan sederhana namun *layer dropout* di awal pada arsitektur sederhana sebelumnya diganti dengan *spatial dropout 1D layer*. Pada pengujian *hyperparameter* juga terdapat perbedaan dengan sederhana yaitu selain adanya *spatial dropout 1D*, juga terdapat konfigurasi nilai *dropout unit* dan *recurrent dropout unit* pada *layer*

LSTM/*Bidirectional*.

Tabel 3. 56 Skenario Arsitektur Kompleks 1

<b>Layer (type) – LSTM</b>	<b>Layer (type) – Bi-LSTM</b>	<b>Output Shape</b>	<b>Param</b>
spatial_dropout1d_2 (SpatialDropout1D)	spatial_dropout1d_2 (SpatialDropout1D)	(None, <max_len>, 300)	Menyesuaikan saat implementasi
LSTM_1 (LSTM)	bidirectional_1 (Bidirectional)	(None, <unit_sel>)	Menyesuaikan saat implementasi
dropout_2 (Dropout)	dropout_2 (Dropout)	(None, <unit_sel>)	Menyesuaikan saat implementasi
dense_1 (Dense)	dense_1 (Dense)	(None, <jumlah_kelas>)	Menyesuaikan saat implementasi

Tabel 3. 57 Skenario hyperparameter Kompleks 1

<b>No</b>	<b>Hyperparameter</b>	<b>Nilai/value</b>
1	<i>Unit neuron</i>	64, 128, 256
2	<i>Dropout layer rate</i>	0.2 – 0.6
3	<i>Dropout rate unit sel LSTM/Bi-LSTM</i>	0.2 – 0.6
4	<i>Recurrent dropout rate unit sel LSTM/Bi-LSTM</i>	0.2 – 0.6
5	<i>Spatial dropout layer rate</i>	- (tidak ada), 0.3 – 0.4
6	<i>Optimizer</i>	<i>adam, rmsprop, sgd</i>

### 3. Kompleks 2

Arsitektur kompleks 2 memiliki struktur yang cukup berbeda dari arsitektur sebelumnya dikarenakan adanya pengembalian sekuens dari *hidden state* sel sebelum-sebelumnya yang mengakibatkan adanya *stack layer LSTM/Bidirectional*. Pada Kompleks 2, perbedaan dari Kompleks 1 adalah terdapat *dropout layer* setelah penambahan *spatial dropout 1D layer* di awal, 2 tumpukan LSTM/Bidirectional layer yang mengembalikan sekuens yang dilanjutkan dengan 1 LSTM/Bidirectional layer biasa yang dikonfigurasikan beberapa *hyperparameter* di dalamnya seperti *dropout unit*, dan *recurrent dropout unit*. Terdapat *dense layer* yang bukan hanya terdapat di akhir struktur, namun juga terdapat di sebelum dan sesudah *stack LSTM/Bidirectional layer*.

<unit\_sel/2>, <unit\_dense>, <unit\_sel LSTM\_3/bidirectional\_3> berturut-turut adalah nilai unit *neuron* yang dibagi 2, dan nilai unit untuk *dense layer* yang

sebenarnya juga merupakan setengah dari nilai unit *neuron* awal, dan nilai unit *neuron* pada tumpukan LSTM/*Bidirectional* terakhir yang sebenarnya juga memiliki nilai yang sama dengan <unit\_dense>. Kemudian juga diuji untuk *hyperparameter* aktivasi pada 2 tumpukan *layer* LSTM/*Bidirectional* seperti yang tertera lengkapnya pada Tabel 59.

Tabel 3. 58 Skenario Arsitektur Kompleks 2

<b>Layer (type) – LSTM</b>	<b>Layer (type) – Bi-LSTM</b>	<b>Output Shape</b>	<b>Param</b>
spatial_dropout1d_2 (SpatialDropout1D)	spatial_dropout1d_2 (SpatialDropout1D)	(None, <max_len>, 300)	Menyesuaikan saat implementasi
dropout_1 (Dropout)	dropout_1 (Dropout)	(None, <max_len>, 300)	Menyesuaikan saat implementasi
dense_1 (Dense)	dense_1 (Dense)	(None, <max_len>, <unit_dense>)	Menyesuaikan saat implementasi
LSTM_1 (LSTM)	bidirectional_1 (Bidirectional)	(None, <max_len>, <unit_sel>)	Menyesuaikan saat implementasi
LSTM_2 (LSTM)	bidirectional_2 (Bidirectional)	(None, <max_len>, <unit_sel>)	Menyesuaikan saat implementasi
LSTM_3 (LSTM)	bidirectional_3 (Bidirectional)	(None, <unit_sel/2>)	Menyesuaikan saat implementasi
dense_2 (Dense)	dense_2 (Dense)	(None, <unit_dense>)	Menyesuaikan saat implementasi
dropout_2 (Dropout)	dropout_2 (Dropout)	(None, <unit_sel LSTM_3/bidirectional_3>)	Menyesuaikan saat implementasi
dense_3 (Dense)	dense_3 (Dense)	(None, <jumlah_kelas>)	Menyesuaikan saat implementasi

Tabel 3. 59 Skenario *hyperparameter* Kompleks 2

No	<i>Hyperparameter</i>	Nilai/value
1	<i>Unit neuron</i>	64, 128, 256
2	<i>Dropout layer rate</i>	0.2 – 0.6
3	<i>Dropout rate unit sel LSTM/Bi-LSTM</i>	0.2 – 0.6
4	<i>Recurrent dropout rate unit sel LSTM/Bi-LSTM</i>	0.2 – 0.6
5	<i>Return sequences</i>	<i>True, False</i>
5a	Aktivasi stack LSTM 1 ret seq=True	- (tidak ada), <i>sigmoid, tanh</i>
5b	Aktivasi stack LSTM 2 ret seq=True	- (tidak ada), <i>sigmoid, tanh</i>
6	<i>Spatial dropout layer rate</i>	- (tidak ada), 0.3 – 0.4
7	<i>Optimizer</i>	<i>adam, rmsprop, sgd</i>

#### 4. Kompleks 3

Kompleks 3 berfokus pada pengujian hyperparameter yang di mana arsitekturnya berasal dari pilihan antara arsitektur kompleks 1 atau arsitektur kompleks 2. Pada kompleks 3, perbedaan pengujian *hyperparameter* dari kompleks 2 adalah dengan mengkonfigurasikan regularisasi L2 untuk *kernel, recurrent* dan *bias* pada *layer LSTM/Bidirectional* yang biasa (setelah tumpukan yang menggunakan *return sequence*) dan juga menguji pengaplikasian *class weight* yang nilainya didasarkan pada perhitungan rasio kelas minor terhadap kelas mayor dan koefisien yang sudah ditentukan oleh penulis yaitu (1 untuk mayor, 1.2 untuk minor1, 1.4 untuk minor2, dan 1.6 untuk minor3).

Tabel 3. 60 Skenario *hyperparameter* Kompleks 3

No	<i>Hyperparameter</i>	Nilai/value
1	<i>Unit neuron</i>	64, 128, 256
2	<i>Dropout layer rate</i>	0.2 – 0.6
3	<i>Dropout rate unit sel LSTM/Bi-LSTM</i>	0.2 – 0.6
4	<i>Recurrent dropout rate unit sel LSTM/Bi-LSTM</i>	0.2 – 0.6
5	<i>Return sequences</i>	<i>True, False</i>
5a	Aktivasi stack LSTM 1 ret seq=True	- (tidak ada), <i>sigmoid, tanh</i>

5b	Aktivasi <i>stack LSTM 2 ret seq=True</i>	- (tidak ada), <i>sigmoid, tanh</i>
6a	Regularisasi <i>L2 kernel sel LSTM/Bi-LSTM</i>	0.02 – 0.04
6b	Regularisasi <i>L2 recurrent sel LSTM/Bi-LSTM</i>	0.02 – 0.04
6b	Regularisasi <i>L2 bias sel LSTM/Bi-LSTM</i>	0.02 – 0.04
7	<i>Spatial dropout layer rate</i>	- (tidak ada), 0.3 – 0.4
8	<i>Class weight</i>	- (tidak ada), Yes (1, 1.2*1/bias, 1.4*1/bias, 1.6*1.bias)
7	<i>Optimizer</i>	<i>adam, rmsprop, sgd</i>

Skenario di atas nilainya dapat berubah bergantung pada hasil implementasi jika sudah didapatkan kesimpulan mana yang lebih baik untuk dilanjutkan ke percobaan selanjutnya agar tidak terlalu banyak perulangan untuk konfig yang tidak begitu baik hasilnya.

Selain itu juga terdapat pengujian jika banyak kelas diubah. Hal ini dipertimbangkan karena adanya pemikiran bahwa dataset yang ada tidak sempurna dan mungkin perlu dilakukan seleksi kelas target dengan melihat karakteristik data mentahnya dan setelah diterapkan metode cerdasnya sebagai bahan perbandingan. Variasi banyak kelas bergantung pada hasil tiap percobaan kelas pada saat implementasi yang dimulai dari 6 kelas, yang diukur juga dengan melihat f1-score.

## BAB IV

### IMPLEMENTASI DAN ANALISIS HASIL

#### 4.1 Implementasi Perangkat Lunak

##### 4.1.1 Preprocessing

Langkah *preprocessing* dalam perangkat lunak terdapat di kelas *preprocessing.py*. Secara keseluruhan, pada *preprocessing*, dilakukan *data cleaning*, penambahan kata untuk model bahasa *fastText*, *one hot encoding*, *padding* kalimat, transformasi *input shape*, pemisahan data latih dan uji, dan opsi *upsample* untuk menyamakan jumlah data latih tiap kelasnya sebelum ditentukan porsi latihnya.

*Data cleaning* yang dilakukan yaitu dengan menerapkan *lowercasing*, hapus nama tokoh, corpus lain yang tidak membawa konteks, dan tanda baca (*stopwords removal*, *punctuation removal*), normalisasi *word vector*. Hal-hal dalam *data cleaning* tersebut dilakukan melalui program yang juga menggunakan list *stopwords* indonesia dari *nltk* dan daftar *punctuation* (tanda baca) dari modul *string* python. Normalisasi *word vector* dilakukan dengan membagi vektor-vektor pada suatu kata tersebut dengan besaran vektornya agar nantinya jika dihitung kembali, setiap kata memiliki besaran yang sama yaitu 1, namun arahnya yang kemungkinan berbeda. *Listing* program dalam melakukan data cleaning dapat dilihat pada Gambar 4.1 s.d. Gambar 4.5.

```

df['potongan_cerita'] = df.potongan_cerita.apply(lambda x: x.lower())
df['potongan_cerita'] = df.potongan_cerita.str.replace('-', ' ')
df['potongan_cerita'] = df.potongan_cerita.str.replace('_', '')
df['potongan_cerita'] = df.potongan_cerita.str.replace('.', '')
df['potongan_cerita'] = df.potongan_cerita.str.replace('nggak','tidak')
df['potongan_cerita'] = df.potongan_cerita.str.replace('ga','tidak')
df['potongan_cerita'] = df.potongan_cerita.str.replace('gak','tidak')
df['potongan_cerita'] = df.potongan_cerita.str.replace('keliatannya','kelihatannya')
df['potongan_cerita'] = df.potongan_cerita.str.replace('enyalah','enyahlah')
df['potongan_cerita'] = df.potongan_cerita.str.replace('gimik','gimmick')
df['potongan_cerita'] = df.potongan_cerita.str.replace('menyoraki','menyorakki')
df['potongan_cerita'] = df.potongan_cerita.str.replace('gibah','ghibah')
df['potongan_cerita'] = df.potongan_cerita.str.replace('ghibabin','ghibah')
df['potongan_cerita'] = df.potongan_cerita.str.replace('naikin','naikkan')
df['potongan_cerita'] = df.potongan_cerita.str.replace('sapiku','sapi')
df['potongan_cerita'] = df.potongan_cerita.str.replace('melongokan','melongo')
df['potongan_cerita'] = df.potongan_cerita.str.replace('disukain','disukai')
df['potongan_cerita'] = df.potongan_cerita.str.replace('berengsek','brengsek')
df['potongan_cerita'] = df.potongan_cerita.str.replace('menyabet','sabot')
df['potongan_cerita'] = df.potongan_cerita.str.replace('nyabet','sabet')
df['potongan_cerita'] = df.potongan_cerita.str.replace('ngejaga','jaga')
```

Gambar 4. 1 *Lowercasing* dan mengganti beberapa kata menjadi lebih sesuai

```

file = open(r"C:\Users\LENOVO\Documents\sem7\Skripsi\hapus_corpus.txt", "r")
hapus_list = []
for line in file:
    hapus_list.append(line.strip())

```

Gambar 4. 2 Load file hapus **corpus.txt** berisi nama tokoh dan kata tidak perlu

```

134 def preprocessing(df, X_test, punctuation, X_train=None, y_train=None, y_
135     punctuation = list(punctuation)
136     punctuation.remove('!')
137     punctuation.remove('?')
138     stop_words = set(stopwords.words('indonesian'))
139     stop_words.update(['.', ',', '"', "'", ':', ';', '(', ')',
140     '|', '[', ']', '{', '}', '``', '\'', '\"', '‘', '‘'])
141     stop_words = list(stop_words)
142     sw_indo = stop_words+list(punctuation)

```

Gambar 4. 3 Penambahan tanda baca untuk dimasukkan ke *list stopword nltk*

```

60 def norm_sent_vector(sentence, hapus, wvec_model, stopwords):
61     vecs = [wvec_model[word.lower()] for word in word_tokenize(sentence)
62             if word.lower() not in stopwords
63             and word.lower() not in hapus]
64     norm_vecs = [vec/np.linalg.norm(vec)
65                 for vec in vecs if np.linalg.norm(vec) > 0]
66     return norm_vecs
67

```

Gambar 4. 4 Normalisasi vektor, stopwords &amp; puctuation removal, tokenizing

```

101 def get_wordvec_enc(stories, hapus, wv, sw):
102     encoded_stories = []
103     for story in stories:
104         vecs = norm_sent_vector(story, hapus, wv, sw)

```

Gambar 4. 5 Pemanggilan function *norm\_sent\_vector*

Penambahan kata untuk model bahasa *fastText* dilakukan di luar pembangunan perangkat lunak karena membutuhkan ruang penyimpanan yang cukup besar, sehingga dalam pembangunan dan pengujian perangkat lunak, yang dilakukan adalah memuat model bahasa *fastText* yang sudah ditambahkan kata yang terkandung dalam *dataset*, yang sebelumnya sudah dilatih di Google Colab. *Listing* program penambahan kata terhadap model bahasa *pretrained fastText wiki-id 300*

dapat dilihat pada Gambar 4.6 s.d. 4.8.

```
print("sample words not found: ", np.random.choice(words_not_found, 800))

sample words not found: ['nyaletuk' 'dan...' 'memelototi' 'dalamku' 'benakku' 'nyatu' 'nyaletuk'
'yaaaah' 'cekalan' 'papamu' 'seperikan' 'jambakin' 'mengeliat' 'kulirik'
'menghadapku' 'bergelombangnya' ''sekarang' 'ngelewatin' 'hubunganku'
'pembalasanku' 'pulang-' 'kubuka' 'ngejaga' 'kuselesaikan'
'eugh' 'nampar' 'pananku' 'kuredam' 'makanya-' 'i' 'susanya'
'seperikan' 'ditaroh' 'ketiakkku' 'kupikirkan' 'mengantarku' '4' 'keawal'
'leherku' 'kusentuh' ''sayang' 'elvan' 'saputera' 'pietak' 'seruku'
'sebelahku' 'jleb' 'menghubungiku' 'kuungkapkan' 'srin' 'cekalan'
'kulkasaku' 'asneanya' 'indah' 'sader' 'ngelirik' 'kubelli' 'isakanaku'
'dalamku' 'sweta' 'jajahat-' 'enyalah' 'pliesetin' 'diisengin' 'menghadapku'
'kutunggu' 'gimik' 'menggunakanku' 'mengangguya' 'memujamu'
'pemikiranku' 'traktiran' 'jahat-dia' 'terlonjak' 'menyudutkanku'
'traktiran' 'tercekat' 'ukungan' 'ratuku' 'almarhum' '25' 'sahutku'
'ceritain' 'gatuu' 'kriekt-' 'ngelunjak' 'senang-' 'membangunkanku'
'jaraku' 'kananya' 'komunitasku' 'arahku' 'brukku' 'tudungnya' '20'
'dipikiranku' 'yakali' 'dhik' 'gapapa' 'seuls' 'didetik' 'mengernyit'
'laper' 'ngebantuin' 'pulang-' 'mendengat' 'nduduh' 'nuggetnya'
'mengangguya' 'cinloknya' 'dulu...' 'kebawa' 'kuselesaikan' 'menyorakkan'
'anda...' 'gibah' 'keniatan' 'seuls' 'ada...' 'iso' 'secempreng' '068'
'meremang' 'gendutku' 'diaryku' '100x150' 'mukidi' 'sousu' 'tenangkan'
'astreanya' 'kegelapannya' 'cangkirnya' 'komunitasku' 'memlin' 'cieee'
'memanyunkan' 'ditooo...' 'konsentrasi' 'kamu' 'diomelin' 'mengadopsiku'
'mengijakkan' 'rambutku' 'kelontongnya' 'lokita' 'pribadiku' 'mentayi'
'palangan' 'aku' 'berhentian' 'naikin' 'dasiku' 'bermonolog' 'ujarku'
'leptopku' 'hayolah' ''bora' 'meyerahanku' 'diomelin' 'mengijakkan'
'memain' 'arahku' 'nyobain' 'menghampiriku' 'tawaku' 'lelakiku' 'keawal'
'mutusin' 'craps' 'busanya' 'punggungku' 'ghibahin' 'direkatakannya'
'sesusah' 'maksain' 'lihatnya' 'munos' 'mi' 'sapiku' 'pedulin'
'mengantarku' 'dicepatkan' 'mentransfusi' 'dan...' 'melongokkan'
'menyudutkanku' 'laper' 'gapapa' 'menggumamkan' 'lirishnya' 'benernya'
'seakrab' 'disukain' 'keniatan' '300' 'contongnya' 'kuselesaikan'
']
```

Gambar 4. 6 Daftar kata yang tidak ada pada *pretrained fastText wiki-id 300*

```
tambahan = ["'Kelakuan Ngelunjak Orang di Chat yang Bikin Naik Darah",
"'Udah dikasi hati malah ngelunjak, KELAKUAAN",
"'makin lama makin ngelunjak lu ye, bikin gua emosi",
"'masih kecil udah ngelunjak, nanti gedé jadi apa",
"'belum apa-apa udah ngelunjak, giman ga marah",
"'kenapa kok marah?' kta dia, gue jawab 'soalnya ngelunjak banget tu orang',
"'gatau batasan, gatau malu njir... bener-bener ngelunjak",
"'kalo aku salah, diomeli ya dimarahin gitu juga gapapa, asal jangan didiemin.",
"'iya saya salah inikan mau diomongin baik baik gausah saysa diomeli juga. Sama kayak kamu ngambekan suka marah. Apasih inikan juga salah kamu mas",
"'Sebentar ini diberesin dulu nanti tumpah semua. Aku dimarahin terus. Yahh gak bisa sabar. makanya sabar biar nggak diomeli mulu",
"'emosi dapat banget sial gue sampai tahan napas NGERASA DIOMELI JG",
"'kenapa kalo aku salah ga dikasih tau malah diomeli? bilangnya ga pernah bantuin, ga becus, padahal aku cuma nanya ini harus apa lagi,tapi malah kemana mar
'yaah kenapa jadiinya gtu. udah bodomat hadeh, trus ini gimanaya ha haduh",
"'Adeh kenapa?' ibi bertanya, "Gatau yaah, coba inget-inget",
"'Yaah kok jadiinya gini? Haduh terus giman bisa selesai hfft"
'"Capek banget fyuuuh, Kirain udah selesai semua ternyata yaah masih banyak tersisa",
"'papa atau bokap lo merhatiin lo banget ya. Dipantau terus supaya anaknya terjaga",
"'papa pulang kerja kapan Ma?",
"'Nanti papa atau ayah kamu bakal ke sini beliin balon. Ditunggu ya, jangan nangis. Ayah atau papa pasti lagi nyari-nyari juga",
"'Kemaren gua ngrobrol sama bokap atau papa lo. Ya banyak cerita tentang bumi datar, akademik. Bokap atau papa lo sayang banget sama lo ya, sabar nungguin.",
"'Ini pembalasanku, membalsam kebaikannu",
"'Tunggu saja pembalasanku, ga kuberi ampuh karena sudah dendam",
"']"]
```

Gambar 4. 7 Tambahan kalimat sesuai kata di luar model *pretrained*

```
p_crta = list(df.potongan_cerita)
for addition in tambahan:
    addition = addition.lower()
    addition = addition.replace('-', ' ')
    p_crta.append(addition)

sentences = [word_tokenize(cerita) for cerita in tqdm(p_crta)]
model.build_vocab(sentences, update=True)

model.train(sentences, total_examples=len(sentences), epochs = 100)
```

Gambar 4. 8 Melanjutkan latih *pretrained* model bahasa *fastText*

*One hot encoding, padding, dan transformasi input shape* berada di tahap yang sama. *One hot encoding* yaitu merubah fitur target menjadi 1 untuk kelas yang dimaksud, dan 0 untuk lainnya. *One hot encoding* dilakukan melalui program dengan memanfaatkan *if statement*. *Padding* yaitu menambahkan karakter ke ruang kosong sehingga dari setiap potongan cerita dalam 630 potongan cerita tersebut memiliki panjang yang sama. *Padding* dilakukan melalui program yang juga dibarengi dengan transformasi *input shape*. Transformasi *input shape* terjadi karena pengaplikasian *for loop* bertingkat yang di dalam *for loop* terdalamnya memasukkan elemen *padding* yang sudah ditetapkan *shape* untuk tiap ruang yang kosong dari potongan cerita yaitu (1,300) dan dilakukan sebanyak panjang maksimum dan *for loop* terdalam tersebut dilakukan sebanyak jumlah potongan cerita yang ada di *dataset*, sehingga didapatkan input shape dengan format (*jumlah\_data, max\_length, feature\_dim*). Listing dalam melakukan *one hot encoding, padding, dan transformasi input shape* dapat dilihat pada Gambar 4.9 s.d. 4.10.

```
83 def suasana_encode(suasana):
84     if suasana == 1:
85         return [1, 0, 0, 0, 0]
86     elif suasana == 2:
87         return [0, 1, 0, 0, 0]
88     elif suasana == 3:
89         return [0, 0, 1, 0, 0]
90     elif suasana == 4:
91         return [0, 0, 0, 1, 0]
92     else:
93         return [0, 0, 0, 0, 1]
94     # else:
95     #     return [0,0,0,0,1]
```

Gambar 4. 9 *One hot encoding* fitur target

```

112 def get_padded_encoded_stories(encoded_stories, max_length):
113     """
114     utk kalimat pendek, siapin zero padding utk semua input jd punya length sama
115     """
116     padded_stories_encoding = []
117     for enc_story in encoded_stories:
118         zero_padding_cnt = max_length - enc_story.shape[0]
119         pad = np.zeros((1, 300))
120         for i in range(zero_padding_cnt):
121             enc_story = np.concatenate((pad, enc_story), axis=0)
122         padded_stories_encoding.append(enc_story)
123     return padded_stories_encoding

```

Gambar 4. 10 Padding dan transformasi *input shape*

Pemisahan data latih dan uji menggunakan modul sklearn dengan *function* *train\_test\_split*. Rasio *split* diambil dari variabel yang menangkap input *user*, dan *random\_state* 42 digunakan agar tiap eksekusi di *runtime* yang sama, hasil acakan *dataset* adalah sama sehingga lebih baik dalam penyelidikan. Listing program pemisahan dataset untuk latih dan uji dapat dilihat pada Gambar 4.11.

```

261 train_data, test_data = train_test_split(df, test_size=test_set
262                                , random_state=42, shuffle=True)

```

Gambar 4. 11 Penggunaan *train-test split* *sklearn*

*Upsample* dilakukan secara opsional, jika *user* memilih untuk melakukan *upsample*. *Upsample* dilakukan dengan metode *resample* (replikasi data) dari masing-masing kelas sehingga jumlahnya mengikuti kelas dengan jumlah terbanyak (kelas mayoritas). Dalam *upsample*, juga terdapat pendefinisian *class\_weights* yang dapat digunakan untuk *hyperparameter* *lstm class\_weight* dalam melakukan *fitting*. *Listing* program untuk *upsample* dapat dilihat pada Gambar 4.11 s.d 4.16.

```

184     #memisahkan kelas mayoritas dan minoritas
185     data_majority = df[df.suasana == 4]
186     data_minor1 = df[df.suasana == 2]
187     data_minor2 = df[df.suasana == 1]
188     data_minor3 = df[df.suasana == 5]
189     data_minor4 = df[df.suasana == 3]
190     # data_minor5 = df[df.suasana == '6']
191
192     #dipake buat definisi class weight nanti
193     bias1= data_minor1.shape[0]/data_majority.shape[0]
194     bias2= data_minor2.shape[0]/data_majority.shape[0]
195     bias3= data_minor3.shape[0]/data_majority.shape[0]
196     bias4= data_minor4.shape[0]/data_majority.shape[0]
197     # bias5= data_minor5.shape[0]/data_majority.shape[0]

```

Gambar 4. 12 Kelas mayor-minor dan *bias class weights*

```

202 train = pd.concat([data_majority.sample(frac=fractions,random_state=rs)
203                     , data_minor1.sample(frac=fractions, random_state=rs)
204                     , data_minor2.sample(frac=fractions, random_state=rs)
205                     , data_minor3.sample(frac=fractions, random_state=rs)
206                     , data_minor4.sample(frac=fractions, random_state=rs)])
207     # ,data_minor5.sample(frac=fractions, random_state=rs)])
208 test = pd.concat([data_majority.drop(
209                     data_majority.sample(frac=fractions, random_state=rs).index),
210                     data_minor1.drop(data_minor1.sample(frac=fractions, random_state=rs).index),
211                     data_minor2.drop(data_minor2.sample(frac=fractions, random_state=rs).index),
212                     data_minor3.drop(data_minor3.sample(frac=fractions, random_state=rs).index),
213                     data_minor4.drop(data_minor4.sample(frac=fractions, random_state=rs).index)
214                 # , data_minor5.drop(data_minor5.sample(frac=fractions, random_state=rs).index
215 train = shuffle(train)
216 test = shuffle(test)

```

Gambar 4. 13 Pemisahan *train-test* sekaligus *shuffling*

```

229 #proses upsampling
230 #misahin mayoritas dan minor di data training untuk upsampling
231 data_majority = train[train.suasana==4]
232 data_minor1 = train[train.suasana==2]
233 data_minor2 = train[train.suasana==1]
234 data_minor3 = train[train.suasana==5]
235 data_minor4 = train[train.suasana==3]
236 # data_minor5 = train[train.suasana=='6']

```

Gambar 4. 14 Kelas mayor-minor untuk data pada *train set*

```

1 248      #upsample kelas minor
2 249      data_minor1_upsampled = resample(data_minor1, replace = True
3 250          , n_samples=data_majority.shape[0], random_state=rs)
4 251      data_minor2_upsampled = resample(data_minor2, replace = True
5 252          , n_samples=data_majority.shape[0], random_state=rs)
6 253      data_minor3_upsampled = resample(data_minor3, replace = True
7 254          , n_samples=data_majority.shape[0], random_state=rs)
8 255      data_minor4_upsampled = resample(data_minor4, replace = True
9 256          , n_samples=data_majority.shape[0], random_state=rs)
10 257      # data_minor5_upsampled = resample(data_minor5, replace = True
11 258          , n_samples=data_majority.shape[0], random_state=rs)
12 259
13 260      #gabungin kelas mayoritas dengan kelas minor yang sudah di-upsample
14 261      data_upsampled = pd.concat([data_majority, data_minor1_upsampled, data_minor2_upsampled
15 262          , data_minor3_upsampled
16 263          , data_minor4_upsampled])
17 264          # , data_minor5_upsampled])

```

Gambar 4. 15 *Upsampling* kelas minor dan penggabungan

```

271  class_weights = {0:1, 1:1.2/bias1, 2:1.4/bias2, 3: 1.6/bias3
272      , 4:1.8/bias4}
273      # , 5:2/bias5}
274      return data_upsampled, test, class_weights

```

Gambar 4. 16 Membuat *dict* *class\_weights* untuk *hyperparameter* *class\_weight*

#### 4.1.2 Modeling

Pembuatan model *LSTM* dan *Bi-LSTM* menggunakan *Keras* berbentuk API. Dibutuhkan mengimpor kelas *Sequential* dari modul ‘models’ pada *keras*, kelas *Dense*, *SpatialDropout1D*, *Dropout*, *LSTM*, *Bidirectional*, dan *BatchNormalization* dari modul ‘layers’ pada *keras*, dan kelas *l2* dari modul ‘regularizers’ pada *keras*. *Listing* pembangunan model dapat dilihat pada Gambar 4.17 s.d. 4.18.

```

# LSTM model
lstm = Sequential()
lstm.add(SpatialDropout1D(0.2))
lstm.add(Bidirectional(LSTM(128,kernel_regularizer=l2(0.02), recurrent_regularizer=l2(0.02),
                           bias_regularizer=l2(0.02), dropout=0.2, recurrent_dropout=0.2)))
lstm.add(Dense(2, activation='softmax'))

```

Gambar 4. 17 Contoh pembangunan model *Bi-LSTM* dengan *regularizers*

```
lstm.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Gambar 4. 18 *Compile* model yang telah dikonfigurasikan

#### 4.1.3 Evaluasi

Evaluasi dilihat dari akurasi dan *loss* dari *train-set* dan *validation-set*. Untuk akurasi dan *loss*, akan tertampil ketika model sedang dibangun untuk setiap *epoch*-nya karena menggunakan API Keras Call seperti yang ditunjukkan di Gambar 4.18. Namun, evaluasi juga dapat ditunjukkan melalui *confusion matrix* dan laporan klasifikasi yang terdapat di kelas *ConfusionMatrixDisplay* dan *function classification\_report* dari modul ‘metrics’ pada *sklearn*. Dalam menunjukkan *confusion matrix*, dibutuhkan nilai fitur target pada *validation set* dalam bentuk sebelum dilakukan *one hot encoding*, dan nilai fitur target prediksi dengan bentuk yang sudah dilakukan *argmax* (dipilih kelas yang memiliki nilai maksimum). *Listing* program untuk menampilkan *confusion matrix* dan laporan klasifikasi dapat dilihat pada Gambar 4.19 dan 4.20.

```
320 ConfusionMatrixDisplay.from_predictions(np.argmax(test_Y, axis=1).tolist()
321                                     , np.argmax(Y_pred, axis=1).tolist())
```

Gambar 4. 19 *Code* untuk menunjukkan *confusion matrix*

```
310 Y_pred = lstm.predict(test_X, batch_size=128)
311 df_test = pd.DataFrame([{'true': np.argmax(test_Y, axis=1).tolist()
312                           , 'pred': np.argmax(Y_pred, axis=1).tolist()}])
313 report = classification_report(df_test.true, df_test.pred)
```

Gambar 4. 20 *Code* untuk menunjukkan laporan klasifikasi

### 4.2 Analisa Hasil

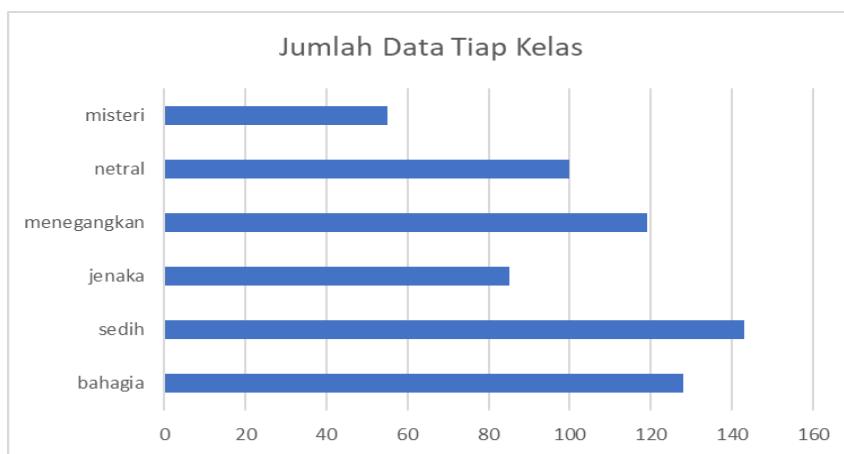
#### 4.2.1 Pengujian Perangkat Lunak

Dari hasil pengujian, perangkat lunak sudah berjalan sesuai harapan dan tidak ada perbaikan. Hasil pengujian perangkat lunak dapat

dilihat pada Lampiran I.

#### 4.2.2 Pengujian dengan Dataset

Dilakukan pengujian terhadap *dataset* yang telah disiapkan dengan mengkombinasikan *layer* yang digunakan dan *hyperparameter* model.



Gambar 4. 21 Grafik Jumlah Data Tiap Kelas

##### 4.2.2.1 Pengujian Arsitektur Sederhana

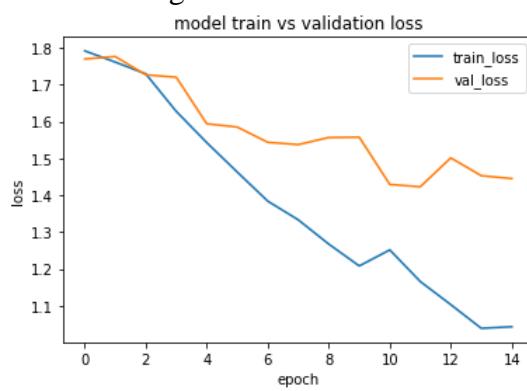
Arsitektur sederhana yang dimaksud adalah hanya terdiri dari layer LSTM dengan unit  $< 300$ , *dropout layer* (**do**) untuk menonaktifkan beberapa *neuron* sesuai dengan *rate* yang diberikan, dan *dense layer* dengan aktivasi *softmax* sebagai *layer output*. Pada pengujian ini, diambil *epoch* 150 sebagai batas maksimum namun dapat berhenti sebelum mencapai 150 karena menggunakan *early-stopping* untuk menghindari terjadinya *overfitting* yang terlalu parah. Nilai *batch size* (**BS**) yang berfungsi untuk mengambil sejumlah *sample* sebelum model terbarui, dan *optimizer* (**Op**) untuk merubah atribut *weights* dan *learning rate* juga dapat diujicobakan. Dalam pengujian ini juga akan dilakukan ujicoba jika *dataset* memiliki 6 kelas, 5 kelas (kelas misteri digabung dengan kelas menegangkan), dan 4 kelas (salah 1 kelas dengan nilai f1 kecil dihapus).

Tabel 4.1 Pengujian Arsitektur LSTM Sederhana dengan 6 Kelas

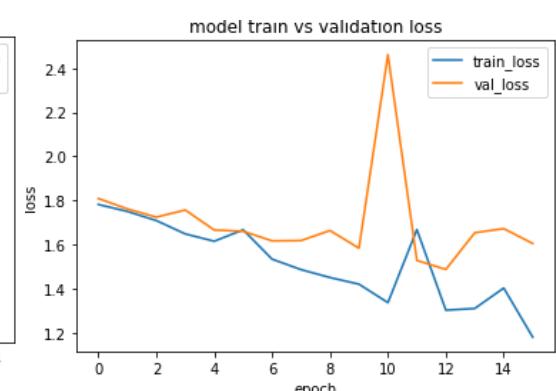
No	n Kls	Unit LSTM	do	BS	Op	Stop at (ep)	Acc Lat	Acc Val	Acc lat up	Acc Val up
1	6	128	0.2	128	adam	13, 15	56.9	40.5	64.3	50.4
2	6	256	0.2	128	adam	13, 16	53.2	40.5	63.4	38.6
3	6	64	0.2	128	adam	5,12	27.8	25.4	58.8	37
4	6	128	0.4	128	adam	14, 17	54.6	38.9	65.9	39.3
5	6	128	0.6	128	adam	11, 13	41.2	34.9	59.2	45.5
6	6	128	0.2	64	adam	10, 12	49.8	39.7	64.3	42.5
7	6	128	0.2	256	adam	17, 13	46	34.9	53.7	41.7
8	6	128	0.2	128	rmsprop	16, 16	48.4	32.5	60.7	41.7
9	6	128	0.2	128	sgd	29, 4	24	25.4	18.1	19.7

Perbandingan *loss* dari *high acc optimizer* adam, rmsprop,

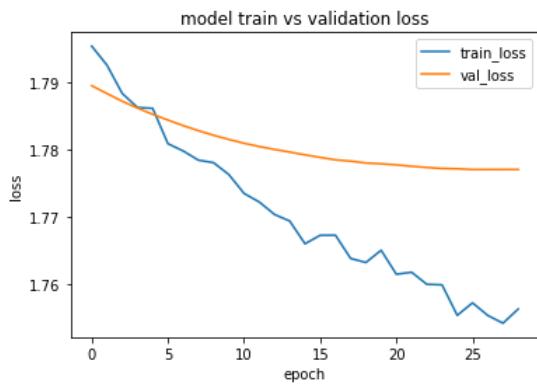
sgd :



Gambar 4. 23 train-val loss adam



Gambar 4. 22 train-val loss rmsprop



Gambar 4. 24 train-val loss sgd

Dari gambar perbandingan tersebut, dapat terlihat bahwa *optimizer sgd* tidak begitu baik dibandingkan kedua *optimizer* lainnya. Untuk *optimizer* yang mencapai *val\_loss* terendah yaitu *adam*, namun jika dilihat kecenderungan kedekatan antara *train\_loss* dan *val\_loss*, *rmsprop* menunjukkan hal yang baik untuk itu. Sehingga, uji selanjutnya akan menggunakan *optimizer adam* ataupun *rmsprop*.

Dari percobaan pada Tabel 4.1, didapatkan konfigurasi arsitektur sederhana yang menghasilkan akurasi validasi terbaik, dan *val loss* yang terbaik adalah di no.1 dengan melakukan *upsample* data.

**Tabel 4.2 Pengujian Arsitektur Bi-LSTM Sederhana dengan 6 Kelas**

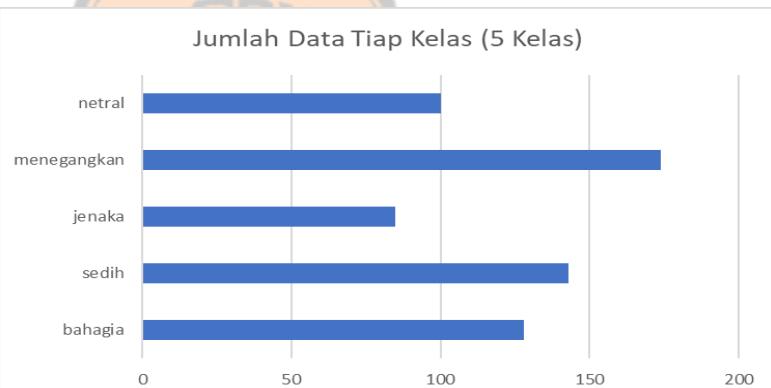
No	n Kls	Unit Bi-LSTM	do	BS	Op	Stop at (ep)	Acc lat up	Acc Val up
1	6	128	0.2	128	adam	13	54.4	37.8
2	6	128	0.2	128	rmsprop	11	40.9	41.7

**Tabel 4.3 Perbandingan f1-score High Acc Sederhana 6 Kelas**

LSTM	bahagia	sedih	lucu	thrill	netral	misteri
f1	62	54	60	37	49	11

Bi-LSTM	bahagia	sedih	lucu	thrill	netral	misteri
f1	49	55	43	22	28	17

Berdasarkan f1-score 6 kelas, kelas terakhir (misteri) memiliki skor yang paling rendah, begitu pula dengan kedua terendah yaitu kelas ke-4 (menegangkan/*thrill*). Kedua kelas tersebut cukup memiliki kesamaan, sehingga untuk percobaan selanjutnya, kelas misteri dengan jumlah data 55 akan digabung ke kelas *thrill* dengan jumlah 119 sehingga secara keseluruhan nantinya akan ada 5 kelas.



Gambar 4. 25 Grafik Jumlah Data Tiap Kelas untuk 5 Kelas

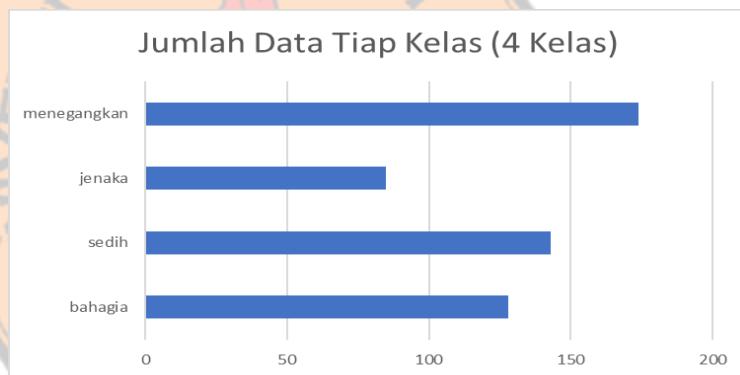
Tabel 4.4 Pengujian Arsitektur Sederhana dengan 5 Kelas

No	n Kls	Unit LSTM	do	BS	Op	Stop at (ep)	Acc lat up	Acc Val up
1	5	128	0.2	128	adam	13	72.2	48
2	5	128	0.2	128	rmsprop	14	70.5	50.4
No	n Kls	Unit Bi- LSTM	do	BS	Op	Stop at (ep)	Acc lat up	Acc Val up
1	5	128	0.2	128	adam	14	52.3	41.7
2	5	128	0.2	128	rmsprop	18	64.1	44

**Tabel 4.5 Perbandingan f1-score High Acc Sederhana 5****Kelas**

LSTM	bahagia	sedih	lucu	thrill	netral
f1	51	61	53	47	41
Bi-LSTM	bahagia	sedih	lucu	thrill	netral
f1	37	59	10	58	23

Berdasarkan f1-score 5 kelas, kelas terakhir (netral) di LSTM memiliki skor terkecil, dan di Bi-LSTM memiliki skor terkecil kedua. Oleh karena itu, untuk percobaan selanjutnya, kelas netral dengan jumlah data 100 akan dihapus sehingga secara keseluruhan nantinya akan ada 4 kelas.



Gambar 4. 26 Grafik Jumlah Data Tiap Kelas untuk 4 Kelas

**Tabel 4.6 Pengujian Arsitektur Sederhana dengan 4 Kelas**

No	n Kls	Unit LSTM	do	BS	Op	Stop at (ep)	Acc lat up	Acc Val up	Acc lat up	Acc Val up
1	4	128	0.2	128	adam	19, 13	68.3	60.3	70.3	61.7
2	4	256	0.2	128	adam	7, 17	43.1	47.6	78.4	55.1
3	4	64	0.2	128	adam	13, 17	49.6	54	75.4	49.5

4	4	128	0.4	128	adam	11, 15	59.1	51.6	71	57.9
5	4	128	0.6	128	adam	13, 15	54	47.6	69.1	57
6	4	128	0.2	64	adam	10, 8	56.9	50.8	70.1	53.3
7	4	128	0.2	256	adam	17, 16	62.3	50	67.8	56.1
8	4	128	0.2	128	rmsprop	20, 21	58.7	48.4	66.4	57.9
9	4	128	0.2	128	sgd	94, 5	42.5	47.6	23.6	37.4

Dari hasil percobaan seperti yang tertampil pada Tabel 4.6, dapat terlihat bahwa konfig 1 menghasilkan akurasi validasi tertinggi dibanding konfig lainnya. Selain itu juga dapat terlihat bahwa hasil *upsample* juga mempengaruhi kenaikan akurasi, walaupun untuk konfig no.9, hasil *upsample* tidak lebih baik hasilnya. Dapat terlihat juga bahwa optimizer sgd tidak begitu baik.

Hal yang disebutkan di atas sejalan dengan yang terjadi pada pengujian dengan 6 kelas, namun perbedaannya pada pengujian 4 kelas akurasi validasi yang didapatkan lebih baik.

Oleh karena itu, maka ujicoba selanjutnya baik dengan LSTM maupun Bi-LSTM akan menggunakan konfigurasi yang sama dengan no.1 dengan melakukan *upsample* data, dan juga dengan *optimizer* selain konfig no.1 yaitu *rmsprop*.

No	n Kls	Unit Bi- LSTM	do	BS	Op	Stop at (ep)	Acc lat	Acc up	Val up
1	4	128	0.2	128	adam	16	75.7	57.9	
2	4	128	0.2	128	rmsprop	18	67.3	50.5	

**Tabel 4.7 Perbandingan f1-score High Acc Sederhana 4****Kelas**

LSTM	bahagia	sedih	lucu	thrill
<b>f1</b>	73	63	63	45
Bi-LSTM	bahagia	sedih	lucu	thrill
<b>f1</b>	60	65	48	56

Tabel 4.7 Perbandingan f1-score High Acc Sederhana 4

Dari hasil percobaan arsitektur dan konfigurasi sederhana, akurasi tertinggi diperoleh di konfigurasi LSTM no.1 untuk 4 kelas yaitu 61.7%. Namun, jika melihat grafik *loss* yang terlampir pada Lampiran 2, maka dapat dikatakan model tersebut mengalami *overfitting*.

#### 4.2.2.2 Pengujian Arsitektur Kompleks

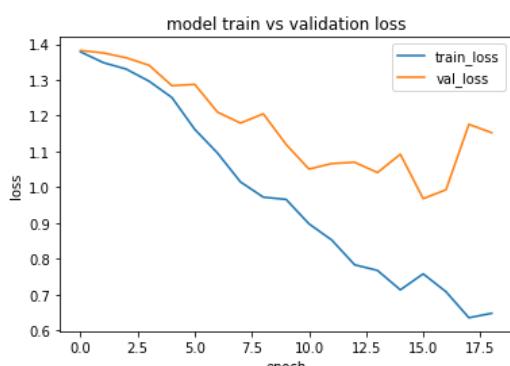
Karena pada pengujian arsitektur sederhana terjadi *overfitting*, maka pada arsitektur kompleks ini akan diuji *layer* yang bagaimana saja yang baik digunakan, takaran *regularizers* seperti apa yang baik. Seperti pengujian sebelumnya, diambil *epoch* 150 sebagai batas maksimum namun dapat berhenti sebelum mencapai 150 karena menggunakan *early-stopping* untuk menghindari terjadinya *overfitting* yang terlalu parah. Jenis *optimizer* juga tetap akan dilakukan ujicoba. *Return sequence* memiliki *layer Dense* sebelum dan setelah *layer LSTM* dengan *return sequence*. Arsitektur pada *layer LSTM* yang dikenakan *return sequence* yaitu berupa *stack LSTM* sebanyak 3. Dalam

pengujian ini akan pada *dataset* dengan 4 kelas karena berdasarkan hasil sebelumnya, 4 kelas lebih memiliki akurasi tinggi.

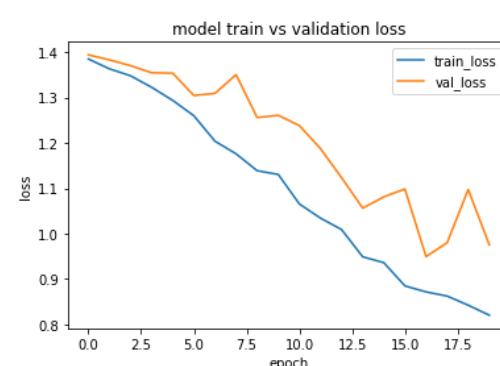
**Tabel 4.8 Pengujian Arsitektur Kompleks LSTM 1 dengan 4 Kelas**

No	n K ls	Unit LST M	Do lyr	Do unit, rec drop unit	Sp Do lyr	op	Stop at (ep)	Acc lat up	Acc Val up
1	4	64	0.2	(0.2, 0.2)	-	adam	19	77.3	56
2	4	64	0.2	(0.2, 0.2)	0.3	adam	20	69.2	60.8
3	4	128	0.2	(0.2, 0.2)	0.3	adam	17	73.4	58.9
4	4	256	0.2	(0.2, 0.2)	0.3	adam	15	67.8	49.5
5	4	128	0.2	(0.4, 0.4)	0.3	adam	22	73.2	61.7
6	4	128	0.3	(0.4, 0.4)	0.4	adam	14	64	60.8
7	4	128	0.3	(0.4, 0.4)	0.4	rmsprop	16	66	57

Dari percobaan Arsitektur Kompleks LSTM 1, dapat dilihat bahwa penggunaan *spatial dropout* 1D berpengaruh dalam menghasilkan akurasi yang lebih baik. Selain itu juga dapat dilihat jika unit LSTM ditambah, maka baik juga jika *dropout layer*, *dropout unit* dan *recurrent dropout unit* pada layer LSTM juga ikut dinaikkan nilainya. Untuk melihat *loss* antara *train-set* dan *val-set* yang baik, dapat dilihat di Gambar 4.27 s.d 4.31.



Gambar 4. 27 Grafik *loss* kompleks 1 no.1



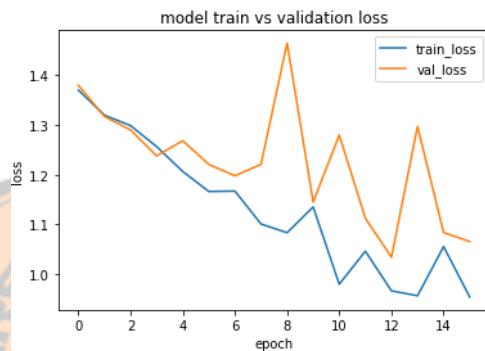
Gambar 4. 28 Grafik *loss* kompleks 1 no.2



Gambar 4. 29 Grafik loss kompleks 1 no.5



Gambar 4. 30 Grafik loss kompleks 1 no.6



Gambar 4. 31 Grafik loss kompleks 1 no.7

Jika dilihat dari kelima gambar grafik *loss* pada Gambar 4.27 s.d 4.31, Nilai *loss* dari *train-set* dan *val-set* terendah ada di konfig no.5 yang juga memiliki akurasi validasi tertinggi dibanding konfig lainnya. Namun, pada grafik *loss* konfig no.6, antara garis *train* dan *val* terlihat hampir bertemu, yang artinya cukup baik karena kemungkinan jika dilanjutkan bisa mencapai *good fitting*, atau juga tidak terlalu mengalami *overfitting* yang besar. Untuk grafik *loss* konfig no.7 yang menggunakan *optimizer* rmsprop juga memperlihatkan garis antara *train* dan *validasi* cukup terlihat dekat, namun terlihat fluktuatif. Oleh karena itu, dapat dikatakan bahwa penggunaan *dropout layer*, *dropout unit*, *recurrent dropout unit*, dan *spatial dropout unit* yang disesuaikan dengan besar jumlah unit LSTM berpengaruh sekali untuk menjaga model tidak mengalami *overfitting* yang terlalu besar dan mendapatkan akurasi yang cukup baik jika

dibandingkan dengan konfig lain selain konfig no.5.

Pada percobaan selanjutnya, akan diuji arsitektur kompleks dengan konfig terbaik di ujicoba sebelumnya yaitu konfig no.5 dan 6, tapi juga menggunakan parameter *return sequence* dengan kondisi *True* dan mencoba fungsi aktivasi untuk *layer LSTM* tersebut.

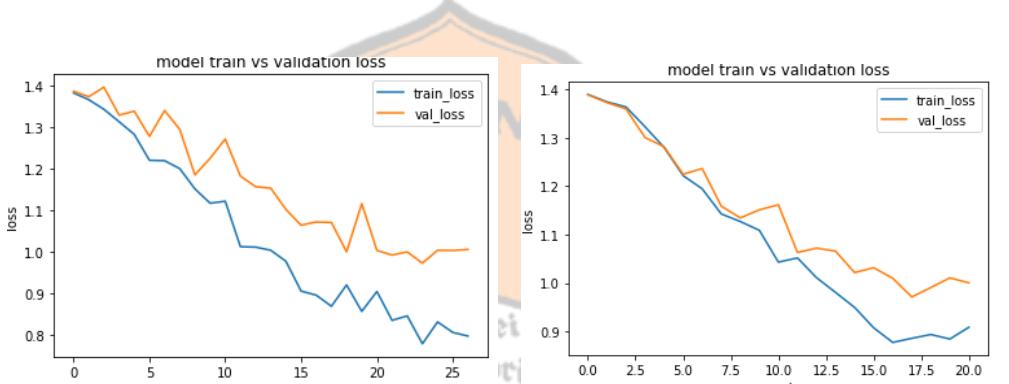
**Tabel 4.9 Pengujian Arsitektur Kompleks 2 dengan 4 Kelas**

No	n Kls	Kon fig LST M	Ret seq	Aktivasi 1	Aktivasi 2	Stop at (ep)	Acc lat up	Acc Val up
1	4	1(5)	True	-	-	28	74.5	49.5
2	4	1(6)	True	-	-	27	67.5	57
3	4	1(5)	True	sigmoid	-	5	22.5	15.9
4	4	1(6)	True	sigmoid	-	4	24.8	15.9
5	4	1(6)	True	sigmoid	sigmoid	6	23.2	15.9
6	4	1(6)	True	tanh	-	21	63.9	58.9
7	4	1(6)	True	tanh	tanh	30	70.3	55.1
8	4	1(6)	True	sigmoid	tanh	8	25.9	23.4
9	4	1(6)	True	tanh	sigmoid	4	25.5	30.8

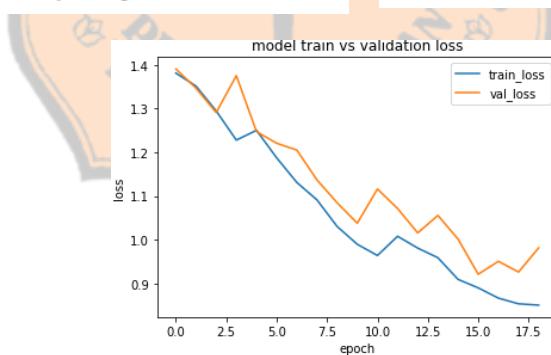
No	n Kls	Kon fig Bi- LST M	Ret seq	Aktivasi 1	Aktivasi 2	Stop at (ep)	Acc lat up	Acc Val up
1	4	1(6)	True	-	-	20	62.8	54.2
2	4	1(6)	True	tanh	-	19	68.5	59.8

Dari Tabel 4.9, dapat dilihat bahwa Bi-LSTM memiliki akurasi validasi yang tinggi dibanding konfig lainnya pada percobaan arsitektur kompleks 2 (menggunakan *return sequences*) dengan menggunakan 1 fungsi aktivasi tanh di *layer* pertama Bi-LSTM. Jika dilihat dari grafik *loss* konfig 2(2), konfig 2(6) LSTM, dan konfig 2(2) Bi-LSTM seperti yang ditunjukkan Gambar 4.32 s.d 4.34, nilai *loss* validasi konfig 2(2) Bi-LSTM lebih rendah dibanding 2 lainnya, namun perbedaannya tidak signifikan dibanding 2 lainnya.



Gambar 4. 34 Grafik *loss* kompleks 2 no.2 LSTM

Gambar 4. 34 Grafik *loss* kompleks 2 no.6 LSTM



Gambar 4. 34 Grafik *loss* kompleks 2 no.2 Bi-LSTM

Selanjutnya akan dilakukan ujicoba menggunakan l2 untuk regularisasi dan setelah didapatkan mana yang lebih baik dari regularisasi, akan dicoba juga menggunakan *class weights* yang sudah didefinisikan saat melakukan upsampling untuk memberikan weights yang berbeda untuk tiap kelas. Pada

percobaan selanjutnya, akan ada 2 bagian yang dicoba yaitu pada ujicoba 1 di arsitektur kompleks yang diberi 12 dan pada ujicoba 2 di arsitektur kompleks yang diberi 12. L2 diberikan pada unit LSTM dengan parameter secara berturut yaitu *kernel regularizer*, *recurrent regularizer*, dan *bias regularizer*.

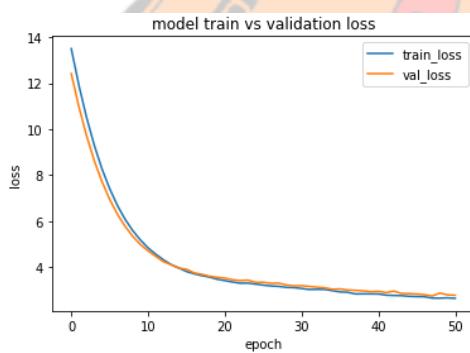
**Tabel 4.10 Pengujian Arsitektur Kompleks 3 dengan 4 Kelas**

No	n Kls	Kon fig LST M	Ker nel reg	Recc reg	Bias reg	Class weigh t	Stop at (ep)	Acc lat up	Acc Val up
1	4	1(5)	0.02	0.02	0.02	-	46	66.6	56.1
2	4	1(6)	0.02	0.02	0.02	-	51	59.9	55.1
3	4	2(2)	0.02	0.02	0.02	-	37	73	56.1
4	4	2(6)	0.02	0.02	0.02	-	41	73.9	62.6
5	4	1(6)	0.02	0.02	0.02	Yes	51	58.3	43.9
6	4	2(6)	0.02	0.02	0.02	Yes	52	74.3	56.1
7	4	1(6)	0.04	0.02	0.02	-	103	59.4	50.5
8	4	1(6)	0.02	0.04	0.02	-	142	61.2	48.6
9	4	1(6)	0.02	0.02	0.04	-	94	61.3	50.5
10	4	2(6)	0.04	0.02	0.02	-	41	72.3	57
11	4	2(6)	0.04	0.02	0.02	Yes	38	68.7	52.3

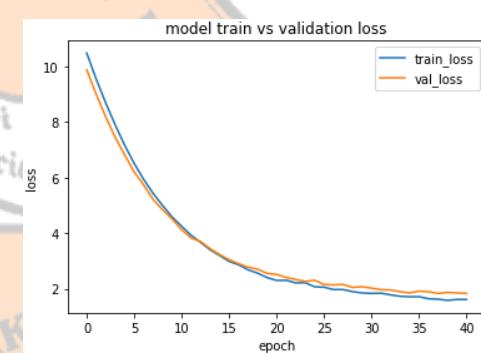
  

No	n Kls	Kon fig Bi- LST M	Ker nel reg	Recc reg	Bias reg	Class weigh t	Stop at (ep)	Acc lat up	Acc Val up
1	4	1(6)	0.02	0.02	0.02	-	103	63.7	60.8
2	4	1(6)	0.04	0.02	0.02	-	150	56.5	52.3
3	4	2(6)	0.02	0.02	0.02	Yes	49	75.5	57
4	4	2(6)	0.04	0.02	0.02	-	42	73.4	59.8

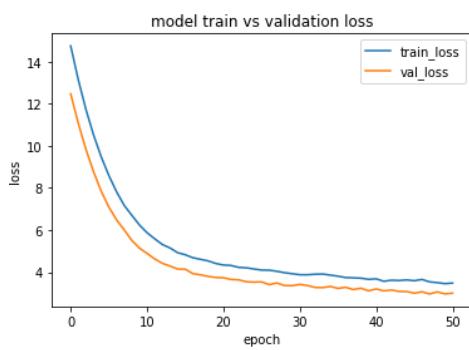
Dari Tabel 4.10, dapat dilihat bahwa penggunaan *class weight* kurang memberikan hasil yang baik dibanding tidak menggunakannya. Hal tersebut juga didukung oleh visualiasi pada Gambar 4.35 s.d. 4.39 yang terlihat bahwa penggunaan *class weight* tidak memberikan *loss val* yang mendekati *loss train*. Selain itu juga dapat dilihat bahwa konfig pada arsitektur LSTM kompleks dengan *return sequence* menggunakan aktivasi tanh 2(6) selalu lebih baik akurasinya dibanding konfig pada arsitektur LSTM kompleks yang hanya menggunakan *dropout layer*, *spatial dropout layer*, *dropout unit*, *reccurent dropout unit* 1(6). Namun begitu, pada arsitektur Bi-LSTM kompleks, konfig 2(6) lebih baik akurasinya dibanding konfig 1(6).



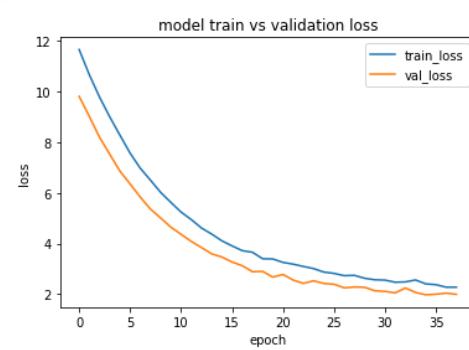
Gambar 4. 35 Grafik *loss* kompleks 3 no.2 LSTM



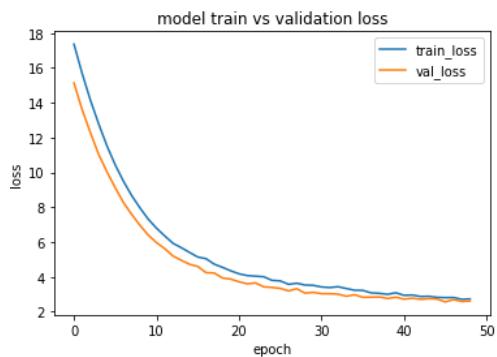
Gambar 4. 36 Grafik *loss* kompleks 3 no.10 LSTM



Gambar 4. 37 Grafik *loss* kompleks 3 no.5 LSTM



Gambar 4. 38 Grafik *loss* kompleks 3 no.11 LSTM



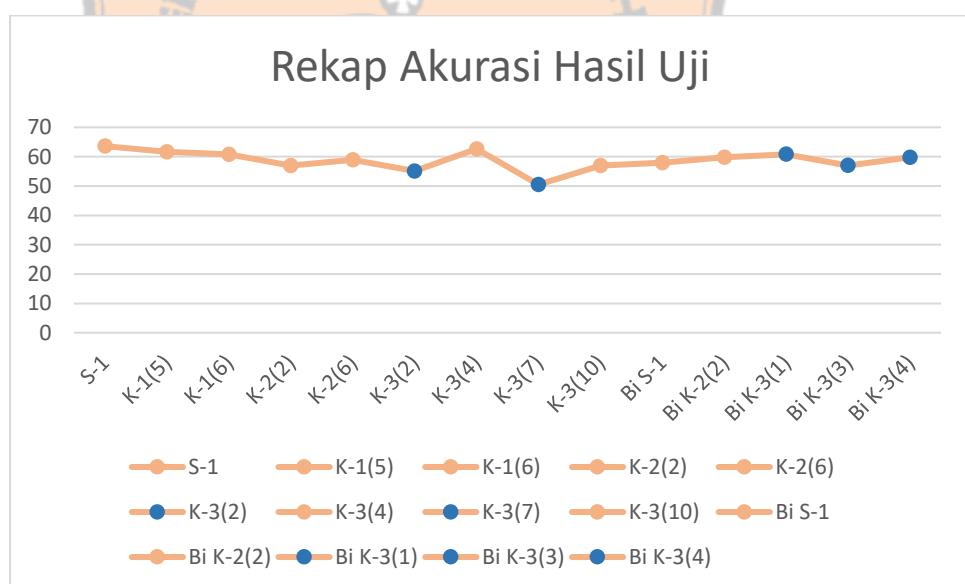
Gambar 4. 39 Grafik loss kompleks 3  
no.3 Bi-LSTM

Berikut pada Tabel 4.11 adalah rekap konfig beserta akurasi dan karakteristik *loss*-nya yang terpilih untuk setiap ujicoba yang dilakukan dari arsitektur sederhana sampai arsitektur kompleks dengan 4 kelas. Kolom ‘Good fit’ pada Tabel 4.11 didapatkan dengan melihat kedekatan garis *loss-train* dan *loss-val* dan tingkat fluktuasinya pada grafis *loss* yang terdapat pada Lampiran 2. Setelah itu, akan dilakukan pemilihan dengan mempertimbangkan antara performa akurasi dan *loss* dari keseluruhan rekap itu untuk mendapatkan beberapa konfig yang final.

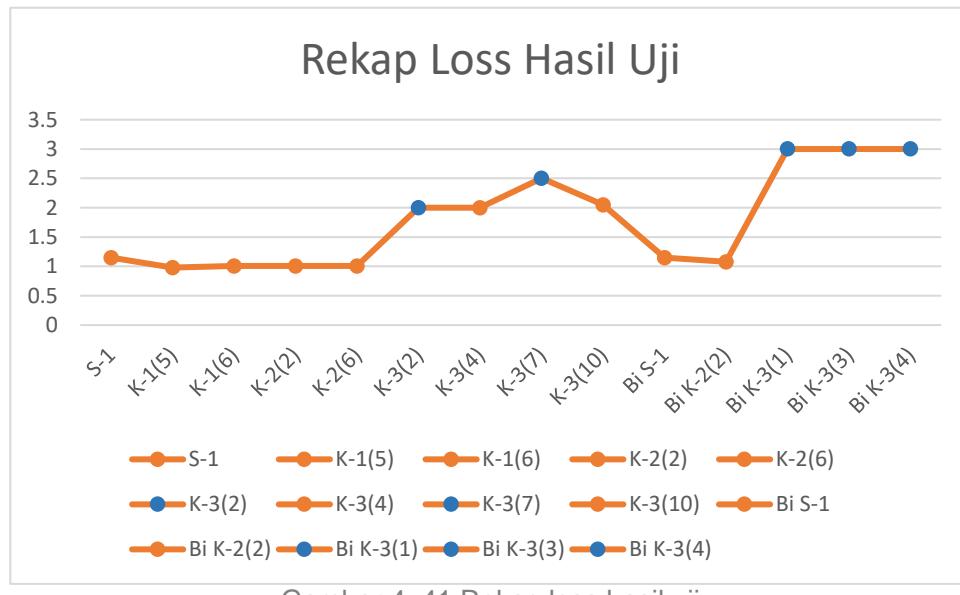
Tabel 4.11 Rekap ujicoba arsitektur sederhana dan kompleks dengan 4 kelas

No	n Kls	Konfig LSTM	Stop at (ep)	Acc lat up	Acc Val up	Good fit	Approx loss val
1	4	S-1	13	70.3	61.7	Tidak	1.15
2	4	K-1(5)	22	73.2	61.7	Tidak	0.98
3	4	K-1(6)	14	64	60.8	Tidak	1.01
4	4	K-2(2)	27	67.5	57	Tidak	1.01
5	4	K-2(6)	21	63.9	58.9	Tidak	1.01
6	4	K-3(2)	51	59.9	55.1	Ya	2

7	4	K-3(4)	41	73.9	62.6	Cukup baik	2
8	4	K-3(7)	103	59.4	50.5	Ya	2.5
9	4	K-3(10)	41	72.3	57	Cukup baik	2.05
No	n Kls	Konfig Bi-LSTM	Stop at (ep)	Acc lat up	Acc Val up	Good fit	Approx loss val
1	4	S-1	16	75.7	57.9	Tidak	1.15
2	4	K-2(2)	19	68.5	59.8	Tidak	1.08
3	4	K-3(1)	103	63.7	60.8	Ya	3
4	4	K-3(3)	49	75.5	57	Ya	3
4	4	K-3(4)	42	73.4	59.8	Ya	3



Gambar 4. 40 Rekap akurasi hasil uji



Gambar 4.41 Rekap loss hasil uji

Dari Gambar 4.40 dan 4.41, terlihat dari 5 titik berwarna biru yang mengindikasikan ujicoba yang menghasilkan ‘*good fit*’ bahwa semua konfig arsitektur Bi-LSTM yang menghasilkan akurasi validasi terbaik mengungguli 2 arsitektur LSTM yang menghasilkan akurasi validasi terbaik. Namun begitu, dapat juga terlihat bahwa nilai *loss* terkecil di antara 5 poin biru tersebut ada di titik pertama dari kiri yaitu konfig 3(2) LSTM dengan nilai sekitar 2, sedangkan untuk ketiga konfig Bi-LSTM yang menghasilkan akurasi terbaik memiliki nilai *loss* yang seragam yaitu sebesar 3. Walaupun begitu, ke-5 poin tersebut sudah merupakan *good fit*. Perbedaan akurasi validasi terbaik Bi-LSTM dan LSTM yaitu sebesar 1.9 dengan perbedaan loss validasinya sebesar 1, sehingga dapat dikatakan bahwa model Bi-LSTM mengungguli LSTM dalam hal kestabilan dan penjagaan agar tidak terlalu *overfitting* dan akurasi yang lebih baik.

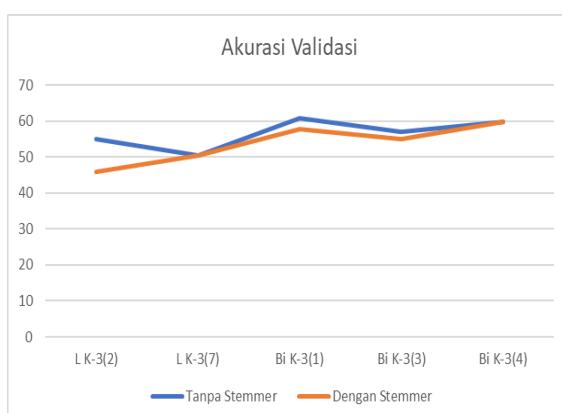
#### 4.2.2.3 Pengujian dengan Menggunakan Sastrawi sebagai Stemmer

Dari pengujian arsitektur, didapatkan beberapa arsitektur yang memiliki akurasi cukup optimal (jika dibandingkan yang

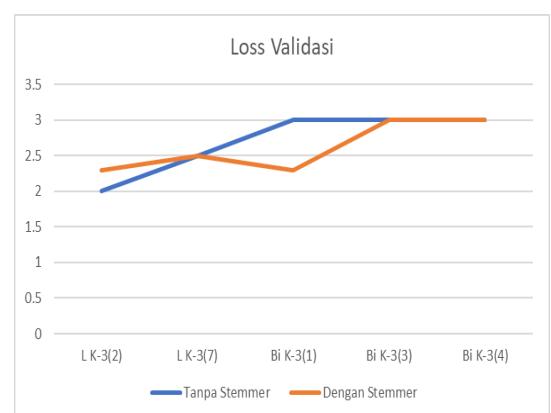
lain) dengan mempertimbangkan lossnya juga. Namun begitu, angka akurasi masih berada di sekitar 50-60. Penulis akan melakukan pengujian dalam hal *preprocessing* yang sebelumnya tidak menggunakan stemming karena FastText sendiri mampu untuk mengenali afiks dan prefiks namun mencoba untuk melihat hasil jika menggunakan dengan metode *stemming* dari modul Sastrawi.

**Tabel 4.12 Ujicoba dengan Stemmer Sastrawi**

No	n Kls	Konfig LSTM	Stop at (ep)	Acc lat up	Acc Val up	Good fit	Approx loss val
1	4	K-3(2)	57	61.2	45.8	Ya	2.3
2	4	K-3(7)	74	48.6	50.5	Ya	2.5
No	n Kls	Konfig Bi-LSTM	Stop at (ep)	Acc lat up	Acc Val up	Good fit	Approx loss val
1	4	K-3(1)	145	65.3	57.9	Ya	2.3
2	4	K-3(3)	48	73.4	55.1	Cukup baik	3
3	4	K-3(4)	65	77.2	59.8	Ya	3



Gambar 4. 43 Akurasi *val-set* dan tanpa *stemmer* dan dengan *stemmer*



Gambar 4. 42 *Loss val-set* dan tanpa *stemmer* dan dengan *stemmer*

Dari Gambar 4.41 dan 4.42, terlihat bahwa menggunakan *stemmer* tidak melampaui performa tanpa *stemmer* untuk akurasi validasi, dan untuk *loss* validasi juga tidak melampaui titik terendah milik percobaan tanpa *stemmer* yaitu dengan aproksimasi nilai 2.

Selain itu dapat juga dilihat bahwa akurasi validasi model Bi-LSTM dengan *stemmer* juga mengungguli akurasi validasi model LSTM kecuali untuk nilai *loss* validasi. Walaupun dengan *stemmer* nilai *loss* validasi terkecil terdapat di poin konfig 3(1) Bi-LSTM, namun untuk 2 konfig Bi-LSTM lainnya memiliki nilai yang lebih besar dibandingkan LSTM. Sedangkan perbandingan antara model Bi-LSTM dan LSTM untuk tanpa *stemmer* sudah dijelaskan sebelumnya di paragraf bawah Gambar 4.41.

Meskipun demikian, perbedaan nilai *loss* validasi antara LSTM dan Bi-LSTM dengan *stemmer* tidak begitu besar yaitu berada di *range* 0-0.7 dan untuk perbedaan akurasi validasi terbaik antara LSTM dan Bi-LSTM yaitu sebesar 9.3. Oleh karena itu dapat dikatakan bahwa dengan *stemmer*, model Bi-LSTM memberikan performa yang lebih baik dibanding LSTM.

## BAB V

### PENUTUP

#### 5.1 Kesimpulan

Berdasarkan hasil penelitian menggunakan LSTM (*Long Short-Term Memory*) dan Bi-LSTM (*Bidirectional Long Short Term Memory*) untuk mengklasifikasi suasana pada teks potongan cerita berbahasa Indonesia, diperoleh kesimpulan sebagai berikut.

1. Metode LSTM dan Bi-LSTM dapat melakukan klasifikasi untuk teks potongan cerita berbahasa Indonesia
2. Akurasi terbaik dari klasifikasi teks potongan cerita berbahasa Indonesia berjumlah 630 didapatkan dengan mereduksi jumlah kelas label menjadi 4 yang masing-masing kelas memiliki jumlah data (bahagia: 128, sedih: 143, jenaka:85, menegangkan:174)
3. Akurasi terbaik dari klasifikasi teks potongan cerita berbahasa Indonesia dengan 4 kelas dan dengan rasio *train-val* 0.8-0.2 adalah sebagai berikut.
  - a. 55.1% untuk klasifikasi dengan data yang telah dilakukan *upsample* dan dengan arsitektur LSTM yang menggunakan konfigurasi kompleks ketiga no.2 yaitu dengan konfigurasi *layer* (jumlah unit =128, *dropout layer* = 0.3, *spatial dropout layer 1D* = 0.4, *dropout unit* dan *reccurrent dropout unit* = 0.4), menggunakan regularisasi 12 dengan porsi yang sama semua untuk *kernel reg*, *reccurrent reg*, dan *bias reg* yaitu sebesar 0.02, tidak menggunakan *class weight*, *adam* sebagai *optimzer*, *batch size* = 128.
  - b. 50.5% untuk klasifikasi dengan data yang telah dilakukan *upsample* dan dengan arsitektur LSTM kompleks ketiga no.7 yang dominan sama seperti di poin a namun *kernel reg* pada regularisasi 12 diubah menjadi lebih besar dibanding 2 lainnya yaitu sebesar 0.04.
  - c. 60.8% untuk klasifikasi dengan data yang telah dilakukan *upsample* dan dengan arsitektur Bi-LSTM kompleks ketiga no.1

yang dominan sama seperti poin a namun perbedaannya hanya poin a merupakan LSTM, dan poin c adalah Bi-LSTM.

- d. 57% untuk klasifikasi dengan data yang telah dilakukan *upsample* dan dengan arsitektur Bi-LSTM yang menggunakan konfigurasi kompleks ketiga no.3 yang dominan sama seperti poin c, namun dengan 3-stack Bi-LSTM yang 2 pertama mengembalikan sekuens dengan fungsi aktivasi 1 yaitu *tanh* pada *stack* teratas Bi-LSTM, terdapat 2 *dense layer* dengan jumlah unit 64 di sebelum dan sesudah 3-stack Bi-LSTM, dan menggunakan *class weight*.
  - e. 59.8% untuk klasifikasi dengan data yang telah dilakukan *upsample* dan dengan arsitektur Bi-LSTM yang menggunakan konfigurasi kompleks ketiga no.4 yang dominan sama seeperti poin d namun *kernel reg* pada regularisasi 12 dibuat lebih besar yaitu 0.04, dan tidak menggunakan *class weight*.
4. Arsitektur Bi-LSTM menghasilkan akurasi lebih baik dibanding LSTM dalam mengklasifikasi teks potongan cerita berbahasa Indonesia baik tanpa *stemmer* maupun dengan *stemmer*.
  5. Dengan menggunakan fastText tanpa *stemmer*, performa klasifikasi sudah lebih baik dibanding dengan menggunakan *stemmer*.

## 5.2 Saran

Berdasarkan penelitian klasifikasi suasana pada cerita pendek berbahasa Indonesia menggunakan metode LSTM dan Bi-LSTM dengan *word embedding* fastText, penulis memberikan saran sebagai berikut.

1. Dikarenakan perangkat lunak adalah berbasis web, diharapkan bahwa *file uploader* dapat menyesuaikan lokasi penyimpanan *user* yang mengunjungi perangkat lunak ini.
2. Dilakukan penelitian selanjutnya dengan topik yang sama namun *dataset* cerita pendek berbahasa Indonesia dikumpulkan lagi baik hanya 4 kelas maupun tanpa mereduksi kelas dengan harapan adanya peningkatan performa.

## Daftar Pustaka

- [1] Nofriani, Dina. 2018. *Analisis Latar Dalam Novel Menggapai Mentari Karya Anastasia Elisa Herman* di <http://repo.stkip-pgri-sumbar.ac.id/id/eprint/249/1/11080328%20DINA%20NOFRIANI%20ok%20Flip.pdf> (diakses Mei 2021)
- [2] Colah. 2015. *Understanding LSTM Networks* di <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> (diakses Mei 2021)
- [3] Nurdin, A., Aji, B. A. S., Bustamin, A., & Abidin, Z. (2020). *Perbandingan Kinerja Word Embedding Word2Vec, Glove, dan Fasttext Pada Klasifikasi Teks*. Jurnal TEKNOKOMPAK, 14(2), 74–79.
- [4] Sari, W. K., Rini, D. P., Malik, R. F., & Azhar, I. S. B. (2020). *Klasifikasi Teks Multilabel pada Artikel Berita Menggunakan Long Short-Term Memory dengan Word2Vec*. Resti, 1(10), 276–285.
- [5] Abdillah,Jiddy. 2020. *Klasifikasi Emosi pada Lirik Lagu menggunakan Metode Bidirectional LSTM dengan Pembobotan GloVe Word Representation* di <https://openlibrary.telkomuniversity.ac.id/pustaka/files/164556/dp/klasifikasi-emosi-pada-lirik-lagu-menggunakan-metode-bidirectional-lstm-dengan-pembobotan-glove-word-representation.pdf> (diakses Mei 2021)
- [6] Han, Jiawei. 2012. *Data Mining Concepts and Techniques Third Edition*. USA: Elsevier
- [7] Ganesan, Kavita. Text Preprocessing for Machine Learning & NLP di <https://kavita-ganesan.com/text-preprocessing-tutorial/#.Yea81N9BzMU> (diakses Januari 2022)
- [8] Bojanowski, P. et al. (2017) ‘Enriching Word Vectors with Subword Information’, Transactions of the Association for Computational Linguistics, 5, pp. 135–146. doi: 10.1162/tacl\_a\_00051.
- [9] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.
- [10] Duchi, J. C., Bartlett, P. L., & Wainwright, M. J. (2012). Randomized smoothing for (parallel) stochastic optimization. *Proceedings of the IEEE Conference on Decision and Control*, 12, 5442–5444. <https://doi.org/10.1109/CDC.2012.6426698>
- [11] Gandhi, Rohith. 2018. *A Look at Gradient Descent and RMSprop Optimizers*

di <https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b> (diakses Mei 2021)

- [12] Yuan, Jing & Tian, Ying. (2019). An Intelligent Fault Diagnosis Method Using GRU Neural Network towards Sequential Data in Dynamic Processes. *Processes*. 7. 152. 10.3390/pr7030152.
- [13] Chaudhary, Mukesh. 2020. *Comparing Sigmoid function with others activation functions and Importance ReLU in Hidden Layer of NN* di <https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5> (diakses Mei 2021)
- [14] Novindasari, Ida. 2020. Mengapa Diperlukan Regularisasi pada Model Neural Network? di <https://idanovinda.medium.com/mengapa-diperlukan-regularisasi-pada-model-neural-network-d622ed98f9a8> (diakses Mei 2021)
- [15] Srivastava, Nitish, et al. 2014. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*
- [16] Khandelwal, Renu. 2018. *L1 and L2 Regularization* di <https://medium.datadriveninvestor.com/l1-l2-regularization-7f1b4fe948f2> (diakses Mei 2021)
- [17] Gomez, Raul. 2018. Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names di [https://gombru.github.io/2018/05/23/cross\\_entropy\\_loss/](https://gombru.github.io/2018/05/23/cross_entropy_loss/) (diakses Mei 2021)
- [18] Lee, Ceshine. 2017. Understanding Bidirectional RNN in PyTorch di <https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66> (diakses Mei 2021)

## Daftar Istilah/Glosarium

1	<i>API</i>	<i>Application Programming Interface</i> , merupakan <i>interface</i> yang dapat menghubungkan satu aplikasi dengan aplikasi lainnya.
2	<i>Artificial Neural Network</i>	Jaringan saraf tiruan, adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan sistem saraf manusia.
3	<i>Backpropagation</i>	Algoritme yang banyak digunakan untuk melatih jaringan saraf maju umpan.
3	<i>Backpropagation through time</i>	Teknik berbasis gradien untuk melatih beberapa jenis jaringan saraf berulang ( <i>recurrent neural network</i> ).
4	<i>Backward pass</i>	Proses menghitung perubahan bobot (pembelajaran <i>de facto</i> ), menggunakan algoritma penurunan gradien (atau yang serupa). Komputasi dilakukan dari lapisan terakhir, mundur ke lapisan pertama.
5	<i>Batch</i>	Sekumpulan hal dalam 1 waktu.
6	<i>Batch size</i>	<i>Hyperparameter</i> yang menentukan jumlah sampel untuk dikerjakan sebelum memperbarui parameter model internal.
7	<i>Bidirectional</i>	2 arah.
8	<i>Cell state</i>	Kemampuan memori jangka panjang yang menyimpan dan memuat informasi dari kejadian-kejadian sebelumnya.
9	<i>Class weight</i>	Pemberian bobot terhadap kelas-kelas target dengan mempertimbangkan distribusi kelas yang miring sehingga dataset untuk

		pembuatan model diharapkan tidak bias.
10	<i>Confusion matrix</i>	Pengukuran performa untuk masalah klasifikasi <i>machine learning</i> dimana keluaran dapat berupa dua kelas atau lebih yang dimana untuk kolom jejeran kelas secara horizontal adalah hasil prediksi, dan baris dari kolom kelas secara vertikal adalah nilai aktual.
11	<i>Convolutional Neural Network</i>	Kelas jaringan saraf tiruan, yang paling umum diterapkan untuk menganalisis citra visual
12	<i>Corpus</i>	Kumpulan teks dokumen.
13	<i>Dataframe</i>	Struktur data berlabel 2 dimensi dengan kolom yang berpotensi dari tipe data yang berbeda.
14	<i>Deep Learning</i>	Cabang pembelajaran mesin yang menggunakan jaringan saraf dengan banyak lapisan.
15	<i>Dense layer</i>	Dikenal juga dengan <i>fully connected layer</i> . Model tradisional <i>neural network</i> yang berfungsi untuk melakukan klasifikasi sesuai dengan <i>class</i> pada <i>output</i> . <i>Dense layer</i> memiliki satu <i>input</i> dan memiliki <i>output</i> yang jumlahnya sesuai dengan jumlah <i>class</i> yang akan diklasifikasikan/diprediksi.
16	<i>Domain-specific corpus</i>	Kumpulan teks dokumen yang spesifik ke suatu <i>domain/asal/area</i> yang telah ditentukan.
17	<i>Dummy</i>	Data tiruan yang digunakan untuk pengujian aplikasi/pembentukan model.
18	<i>Early stopping</i>	Bentuk regularisasi yang menghentikan

		proses <i>training</i> ketika metrik yang diobservasi sudah stagnan dalam beberapa <i>epoch</i> ke depan yang sudah ditentukan, tujuannya yaitu untuk menghindari <i>overfitting</i> saat pelatihan.
19	<i>Epoch</i>	Ketika seluruh <i>dataset</i> sudah melalui proses <i>training</i> pada <i>neural netwok</i> sampai dikembalikan ke awal untuk sekali putaran.
20	<i>F1/F1-score</i>	Dikenal juga dengan sebutan <i>F-score</i> atau <i>F-measure</i> . Merupakan ukuran akurasi suatu tes yang mengkombinasikan <i>recall</i> dan <i>precision</i> . Ukuran yang menampilkan timbal balik antara <i>recall</i> dan <i>precision</i> yang merupakan bobot <i>harmonic mean</i> dan <i>recall</i> dan <i>precision</i> .
21	<i>fastText</i>	Perpustakaan/ <i>lib</i> untuk mempelajari penyisipan kata dan klasifikasi teks yang dibuat oleh lab Penelitian AI Facebook.
22	<i>Feature-engineering</i>	Tindakan mengubah pengamatan mentah menjadi fitur yang diinginkan menggunakan pendekatan statistik atau pembelajaran mesin.
23	<i>Feature extraction</i>	Proses mengubah data mentah menjadi fitur numerik yang dapat diproses sambil mempertahankan informasi dalam kumpulan data asli
24	<i>Feed-forward</i>	Algoritma untuk menghitung vektor keluaran dari vektor masukan.
25	<i>Forget gate</i>	Salah 1 <i>gate</i> atau <i>layer sigmoid</i> di dalam sel LSTM/Bi-LSTM yang berfungsi untuk menentukan input diterima atau dilupakan.

26	<i>Generalization error</i>	Ukuran seberapa akurat suatu algoritme mampu memprediksi nilai hasil untuk data yang sebelumnya tidak terlihat.
27	<i>Gensim</i>	Pustaka sumber terbuka/ <i>open source lib</i> untuk pemodelan topik tanpa pengawasan dan pemrosesan bahasa alami, menggunakan pembelajaran mesin statistik modern.
28	<i>GloVe</i>	Singkatan dari <i>Global Vector</i> yang digunakan untuk representasi kata. Ini adalah algoritma pembelajaran tanpa pengawasan ( <i>unsupervised</i> ) yang dikembangkan oleh Stanford untuk menghasilkan <i>word embeddings</i> dengan menggabungkan matriks kemunculan kata-kata global dari korpus.
29	<i>Gradient optimizer</i>	Pendekatan optimasi umum dan sederhana yang secara iteratif memperbarui parameter untuk menaikkan (turun dalam kasus minimalisasi) gradien fungsi tujuan.
30	<i>GUI</i>	Jenis antarmuka pengguna yang menggunakan metode interaksi pada peranti elektronik secara grafis antara pengguna dan komputer.
31	<i>Hashing</i>	Fungsi apapun yang dapat digunakan untuk memetakan data dengan ukuran arbitrer ke nilai ukuran tetap.
32	<i>Hidden layer</i>	Lapisan di antara lapisan <i>input</i> dan lapisan <i>output</i> , di mana <i>neuron</i> buatan mengambil satu <i>set input</i> berbobot dan menghasilkan <i>output</i> melalui fungsi aktivasi.
33	<i>Hyperparameter</i>	Konfigurasi yang berada di luar model dan

		yang nilainya tidak dapat diperkirakan dari data.
34	<i>Image processing</i>	Salah satu cabang informatika yang berkutat pada usaha untuk melakukan transformasi suatu citra menjadi citra lain dengan menggunakan teknik tertentu.
35	<i>Input gate</i>	Salah 1 <i>gate</i> di dalam sel LSTM/Bi-LSTM yang berkerja dengan menerima <i>input</i> kondisi sekarang dan <i>input</i> dari <i>hidden state</i> sel LSTM/Bi-LSTM sebelumnya yang bercabang 2 masuk ke <i>layer sigmoid</i> dan <i>layer tanh</i> yang kemudian membentuk calon <i>cell state</i> baru setelah operasi selanjutnya.
36	<i>Input shape</i>	Bentuk atau dimensi format <i>input</i> secara keseluruhan.
37	<i>Keras</i>	Perpustakaan perangkat lunak sumber terbuka/ <i>open source lib</i> yang menyediakan antarmuka Python untuk jaringan saraf tiruan.
38	<i>Konvergen</i>	Memusat atau tidak menyebar.
39	<i>Language model</i>	Distribusi probabilitas atas urutan kata-kata.
40	<i>Learning rate</i>	Parameter penyetelan dalam algoritme pengoptimalan yang menentukan ukuran langkah pada setiap iterasi sambil bergerak menuju nilai minimum fungsi kerugian ( <i>loss function</i> ).
41	<i>Lemmatization</i>	Proses pengelompokan bersama bentuk infleksi kata sehingga dapat dianalisis sebagai satu item, diidentifikasi dengan kata lemma, atau bentuk kamus.

42	<i>Listing</i>	Daftar program adalah daftar baris kode komputer atau data digital.
43	<i>Loss</i>	Kesalahan prediksi pada <i>neural network</i> .
44	<i>Lowercasing</i>	Mengubah semua karakter dalam teks dokumen menjadi huruf kecil yang membantu dalam tahap praproses dan pada tahap selanjutnya dalam aplikasi <i>natural language processing</i> .
45	<i>LSTM</i>	Arsitektur jaringan saraf tiruan berulang yang digunakan dalam bidang <i>deep learning</i> yang memiliki karakteristik utama yaitu mampu mengingat informasi penting untuk waktu yang lama.
46	<i>Machine learning</i>	Disiplin ilmu yang mencakup perancangan dan pengembangan algoritme yang memungkinkan komputer untuk mengembangkan perilaku berdasarkan data empiris, seperti dari sensor data basis data.
47	<i>N-gram</i>	Urutan yang berdekatan dari n <i>item</i> dari sampel teks atau ucapan yang diberikan. <i>Item</i> dapat berupa fonem, suku kata, huruf, kata atau pasangan basa sesuai dengan aplikasi.
48	<i>Natural Language Processing</i>	Cabang ilmu komputer, linguistik, dan kecerdasan buatan yang mengkaji interaksi antara komputer dan bahasa manusia, khususnya cara memprogram komputer untuk mengolah data bahasa alami dalam jumlah besar.
49	<i>Neuron</i>	Fungsi matematika yang memodelkan fungsi <i>neuron</i> biologis.

50	<i>Node</i>	Unit komputasi yang memiliki satu atau lebih koneksi <i>input</i> berbobot, fungsi transfer yang menggabungkan <i>input</i> dalam beberapa cara, dan koneksi <i>output</i> .
51	<i>Noise removal</i>	Istilah penghapusan karakter digit dan potongan teks yang dapat mengganggu analisis teks <i>dataset</i> .
52	<i>Normalization</i>	Proses normalisasi dengan rumus-rumus normalisasi seperti contohnya <i>MinMax</i> . Sedangkan normalisasi teks adalah proses mengubah teks menjadi bentuk kanonik tunggal yang mungkin belum pernah ada sebelumnya.
53	<i>One hot encoding</i>	Salah satu metode <i>encoding</i> . Metode ini merepresentasikan data bertipe kategori sebagai vektor biner yang bernilai <i>integer</i> , 0 dan 1, dimana semua elemen akan bernilai 0 kecuali satu elemen yang bernilai 1, yaitu elemen yang memiliki nilai kategori tersebut.
54	<i>Output gate</i>	<i>Gate</i> atau <i>layer sigmoid</i> dalam LSTM/Bi-LSTM yang menentukan nilai <i>hidden state</i> berikutnya.
55	<i>Output layer</i>	Lapisan terakhir <i>neuron</i> yang menghasilkan keluaran tertentu untuk program.
56	<i>Overfitting</i>	Suatu keadaan dimana data yang digunakan untuk pelatihan itu adalah yang "terbaik".
57	<i>Padding</i>	Penambahan data pada awal, tengah, atau akhir.
58	<i>Pixel</i>	Unsur gambar atau representasi sebuah titik terkecil dalam sebuah gambar grafis yang

		dihitung per inci.
59	<i>Preprocessing</i>	Manipulasi atau penghapusan data sebelum digunakan untuk memastikan atau meningkatkan kinerja, dan merupakan langkah penting dalam proses penambangan data.
60	<i>Pretrained model</i>	Model yang dibuat oleh orang lain yang dapat langsung digunakan atau mungkin dilanjutkan pelatihannya.
61	<i>Pseudocode</i>	Deskripsi tingkat tinggi informal dan ringkas atas algoritme pemrograman komputer yang menggunakan konvensi struktural atas suatu bahasa pemrograman, dan ditujukan untuk dibaca oleh manusia dan bukan oleh mesin.
62	<i>Punctuation</i>	Tanda baca.
63	<i>Recurrent Neural Network</i>	Kelas jaringan saraf tiruan di mana koneksi antara node membentuk grafik berarah atau tidak berarah sepanjang urutan temporal.
64	<i>Sekuens</i>	Rangkaian/rentetan
65	<i>Sklearn</i>	Perpustakaan pembelajaran mesin perangkat lunak gratis/ <i>open source machine learning lib</i> untuk bahasa pemrograman Python.
66	<i>Sparse gradient</i>	Ketika <i>network</i> tidak menerima sinyal yang cukup kuat untuk menyesuaikan bobotnya, contohnya gradien yang menghilang ( <i>vanishing gradient</i> ).
67	<i>Speech recognition</i>	Suatu pengembangan teknik dan sistem yang memungkinkan komputer untuk menerima masukan berupa kata yang diucapkan.
68	<i>Stemming</i>	Pemotongan imbuhan pada kata berimbuhan

		yang dijalankan dengan algoritme tertentu.
69	<i>Stochastic Gradient Descent</i>	Metode iteratif untuk mengoptimalkan fungsi objektif dengan properti kehalusan yang sesuai. Ini dapat dianggap sebagai pendekatan stokastik dari pengoptimalan penurunan gradien, karena ia menggantikan gradien aktual dengan perkiraannya.
70	<i>Stopword(s) removal</i>	Penghapusan informasi tingkat rendah dari teks dokumen agar memberikan lebih banyak fokus pada informasi penting.
71	<i>Stratified</i>	Metode pengambilan sampel dari suatu populasi yang dapat dipartisi menjadi subpopulasi.
72	<i>Subword</i>	Potongan kata, bagian dari suatu kata.
73	<i>Tokenizing</i>	Cara memisahkan sepotong teks menjadi <i>unit</i> yang lebih kecil yang disebut <i>token</i> .
74	<i>Upsample</i>	Prosedur di mana titik data yang dihasilkan secara sintetis (sesuai dengan kelas minoritas) disuntikkan ke dalam <i>dataset</i> .
75	<i>User interface</i>	Titik akses di mana pengguna berinteraksi dengan desain.
76	<i>Web scraping</i>	Kegiatan yang dilakukan untuk mengambil data tertentu secara semi-terstruktur dari sebuah halaman situs <i>web</i> .
77	<i>Weight</i>	Bobot/koefisien dari variabel <i>input</i> .
78	<i>Word2Vec</i>	Teknik untuk pemrosesan bahasa alami yang diterbitkan pada tahun 2013. Algoritma <i>word2vec</i> menggunakan model jaringan saraf untuk mempelajari asosiasi kata dari kumpulan teks yang besar.

79	<i>Word Embedding</i>	Istilah yang digunakan untuk representasi kata untuk analisis teks, biasanya dalam bentuk vektor bernilai nyata yang mengkodekan makna kata sedemikian rupa sehingga kata-kata yang lebih dekat dalam ruang vektor diharapkan.
80	<i>Zero-centered</i>	Saat rata-rata (rata-rata) data terletak pada nol.





# LAMPIRAN

## LAMPIRAN I: PENGUJIAN PERANGKAT LUNAK

Test case	Deskripsi	Prosedur Pengujian	Masukan (input)	Keluaran yang diharapkan	Hasil yang didapat	Catatan Proses Pengembangan
Upload dataset (halaman training)	1. Menguji dengan file masukan ekstensi .xlsx	1. Jalankan program 2. Klik halaman ‘training’ 3. Klik tombol ‘Browse files’	Dataset cerpen.csv	Data pada ‘Dataset cerpen.csv’ ditampilkan pada tabel perangkat lunak	Data pada ‘Dataset cerpen.csv’ ditampilkan pada tabel perangkat lunak	Tidak ada perbaikan
	2. Menguji dengan file masukan ekstensi .csv		Dataset cerpen.xlsx	Data pada ‘Dataset cerpen.xlsx’ ditampilkan pada tabel perangkat lunak	Data pada ‘Dataset cerpen.xlsx’ ditampilkan pada tabel perangkat lunak	Tidak ada perbaikan

Konfig dan tuning arsitektur LSTM dan Bi-LSTM	1. Menguji elemen UI pada konfig sederhana	<p>1. Klik halaman ‘training’</p> <p>2. Arahkan pandangan pada sidebar sebelah kiri dari pandangan pengguna</p> <p>3. Pilih arsitektur dengan menekan dropdown</p> <p>4. Pilih konfig sederhana/kompleks dengan menekan radio button berkaitannya</p> <p>. Isi field untuk tiap konfig</p>	<p>Input user untuk tiap elemen UI yaitu: select box (pemilihan arsitektur), radio button (pemilihan konfigurasi sederhana), pengisian text field unit model dan Dropout</p> <p>layer, expander hyperparameter, text field dan select box pada expander</p> <p>hyperparameter (batch size, epoch, optimizer), select box rasio train-test,</p>	<p>Mendapatkan value dari elemen UI: select box (pemilihan arsitektur), radio button (pemilihan konfigurasi sederhana), pengisian text field unit model dan Dropout</p> <p>layer, text field dan select box pada expander</p> <p>hyperparameter (batch size, epoch, optimizer), select box rasio train-test,</p>	<p>Mendapatkan value dari elemen UI: select box (pemilihan arsitektur), radio button (pemilihan konfigurasi sederhana), pengisian text field unit model dan Dropout</p> <p>layer, text field dan select box pada expander</p> <p>hyperparameter (batch size, epoch, optimizer), select box rasio train-test,</p>	Tidak ada perbaikan
---	--	--	--	--	--	---------------------

		select box rasio train-test, checkbox upsample	checkbox upsample	checkbox upsample	
2. Menguji elemen UI pada konfig kompleks		Input user untuk tiap elemen UI yaitu: select box (pemilihan arsitektur), radio button (pemilihan konfigurasi kompleks), expander layer, expander model param, expander hyperparameter, pengisian text	Mendapatkan value dari elemen UI: select box (pemilihan arsitektur), radio button (pemilihan konfigurasi kompleks), pengisian text	Mendapatkan value dari elemen UI: select box (pemilihan arsitektur), radio button (pemilihan konfigurasi kompleks), pengisian text	Tidak ada perbaikan

			field Dropout dan Spatial Dropout 1D pada expander layer, checkbox (return sequence), dropdown aktivasi 1 dan aktivasi 2, pengisian text field (neuron unit, dropout unit, reccurent dropout unit), radio button regularizer pada expander model param, pengisian text field (kernel reg, reccuret reg, dan bias reg) jika memilih pilihan 'Ya' button regularizer, text field dan select	(return sequence), dropdown aktivasi 1 dan aktivasi 2, pengisian text field (neuron unit, dropout unit, reccurent dropout unit), radio button regularizer pada expander model param, pengisian text field (kernel reg, reccuret reg, dan bias reg) jika memilih pilihan 'Ya' button regularizer, text field dan select	(return sequence), dropdown aktivasi 1 dan aktivasi 2, pengisian text field (neuron unit, dropout unit, reccurent dropout unit), radio button regularizer pada expander model param, pengisian text field (kernel reg, reccuret reg, dan bias reg) jika memilih pilihan 'Ya' button regularizer, text field dan select	
--	--	--	---	--	--	--

			memilih pilihan 'Ya' button regularizer, text field dan select box pada expander hyperparameter (batch size, epoch, optimizer), select box rasio train-test, checkbox upsample	box pada expander hyperparameter (batch size, epoch, optimizer), select box rasio train-test, checkbox upsample	box pada expander hyperparameter (batch size, epoch, optimizer), select box rasio train-test, checkbox upsample	
Proses pelatihan dan simpan model	1. Menguji jika hanya melakukan pelatihan	1. Klik halaman 'training' 2. Upload dataset terlebih dahulu seperti di test case 1	1. Dataset dengan format yang sesuai 2. Pilihan arsitektur dan konfigurasi model	1. Detail mengenai hasil pelatihan 2. Pilihan arsitektur, confusion matrix, classification	1. Detail mengenai hasil pelatihan (arsitektur, confusion matrix, classification)	Tidak ada perbaikan
	2. Menguji jika	3. Pilih arsitektur				

	<p>melakukan pelatihan sekaligus simpan data</p>	<p>dan konfigurasi seperti di test case 2</p> <p>4. Masukkan nama model di field yang tersedia</p> <p>5. Klik radio button dengan pilihan “OFF”, “HANYA TRAIN”, “TRAIN DAN SIMPAN”</p>	<p>3. Input user dari elemen UI text field untuk nama model</p> <p>4. Input user dari elemen UI radio button antara “HANYA TRAIN” atau “TRAIN DAN SIMPAN”</p>	<p>report, grafik akurasi dan loss)</p> <p>2. Model tersimpan di folder besar program dengan folder baru sesuai dengan nama yang diinput user</p>	<p>report, grafik akurasi dan loss)</p> <p>2. Model tersimpan di folder besar program dengan folder baru sesuai dengan nama yang diinput user</p>	
Load model untuk halaman ‘Tentang Model’	Menguji jika detail tentang model yang telah disimpan tertampil pada halaman	<p>1. Klik halaman ‘Tentang Model’</p> <p>2. Klik ‘Browse files’</p> <p>3. Pilih kedua model yang telah disimpan dengan ekstensi ‘.h5’ dan</p>	Input file model dengan ekstensi ‘.h5’ dan ‘.pkl’	<p>Menampilkan dictionary akurasi dan loss tiap epoch,</p> <p>menampilkan grafik akurasi dan loss train-val set pada pelatihan</p>	<p>Menampilkan dictionary akurasi dan loss tiap epoch,</p> <p>menampilkan grafik akurasi dan loss train-val set pada pelatihan</p>	Tidak ada perbaikan

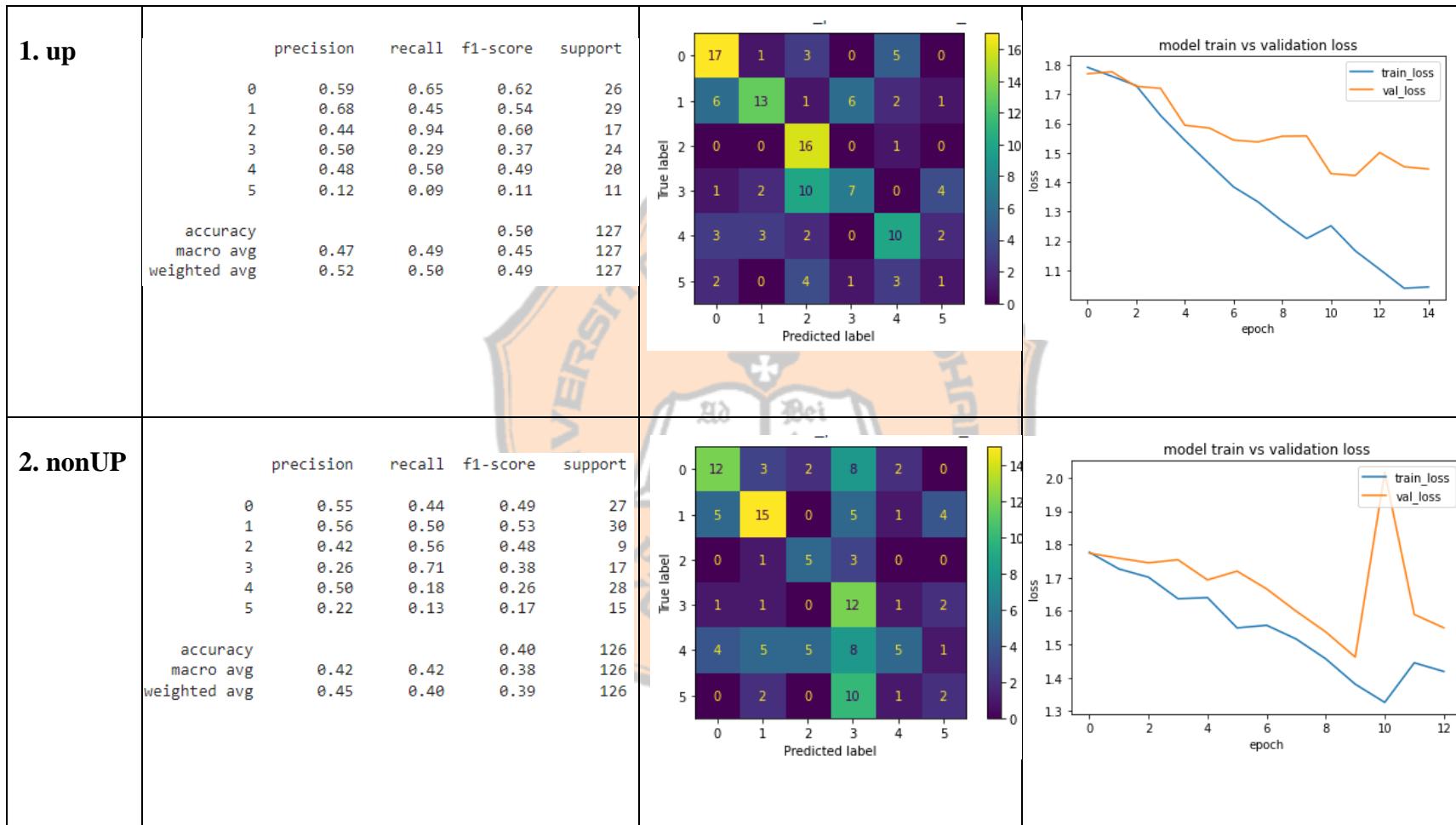
		‘.pkl’ sekaligus		sebelumnya yang dialami model itu, ringkasan model yang dibentuk (model summary)	sebelumnya yang dialami model itu, ringkasan model yang dibentuk (model summary)	
Prediksi atau penggunaan	1. Menguji load model pada halaman ‘Penggunaan’	1. Klik halaman ‘Penggunaan’ 2. Upload model dengan ekstensi ‘.h5’ di file uploader yang bertuliskan ‘Browse files’	Input model yang sudah dibuat berekstensi ‘.h5’	Model dengan ekstensi ‘.h5’ terload yang indikasinya adalah terlihat label nama file di bawah file uploader		Tidak ada perbaikan
	2. Menguji input berbentuk teks	3. Pilih jenis input data ‘Teks’ atau ‘Excel/csv’  4. Jika memilih ‘Teks’ maka setelah mengisi text area,	1. Input potongan cerita tunggal atau banyak (jika banyak dipisahkan dengan enter 2	1. Tabel dengan kolom ‘potongan_cerita’ tertampil  2. Tabel probabilitas untuk	1. Tabel dengan kolom ‘potongan_cerita’ tertampil  2. Tabel probabilitas untuk	

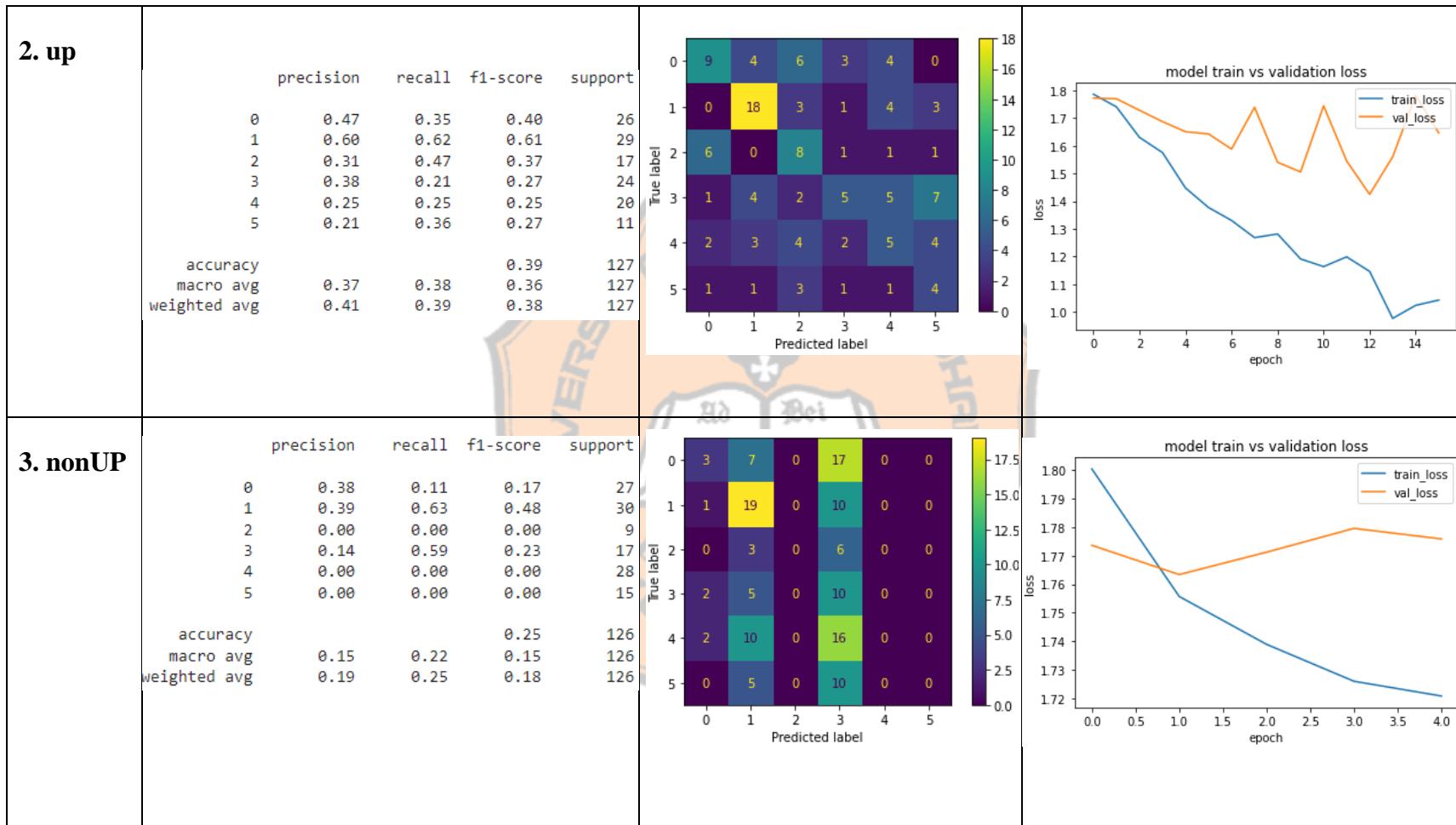
		klik ‘Ctrl+Enter’ 5. Klik checkbox ‘Uji’	kali) 2. Input dari elemen UI checkbox ‘Uji’	tiap kelas, dan hasil akhir prediksi (kelas terprediksi) tertampil	tiap kelas, dan hasil akhir prediksi (kelas terprediksi) tertampil	
	3. Menguji input berbentuk dataset excel atau csv tanpa fitur target (label)		1. Input dataset dengan ekstensi ‘.xls’ atau ‘.xlsx’ atau ‘.csv’ dengan format kolom yaitu ‘id’, ‘potongan_cerita’ 2. Input dari elemen UI checkbox ‘Uji’	1. Tabel dengan kolom ‘id’ ‘potongan_cerita’ tertampil 2. Tabel probabilitas untuk tiap kelas, dan hasil akhir prediksi (kelas terprediksi) tertampil	1. Tabel dengan kolom ‘id’ ‘potongan_cerita’ tertampil 2. Tabel probabilitas untuk tiap kelas, dan hasil akhir prediksi (kelas terprediksi) tertampil	
	4. Menguji input berbentuk		1. Input dataset dengan ekstensi ‘.xls’ atau ‘.xlsx’	1. Tabel dengan kolom ‘id’ ‘potongan_cerita’,	1. Tabel dengan kolom ‘id’ ‘potongan_cerita’,	

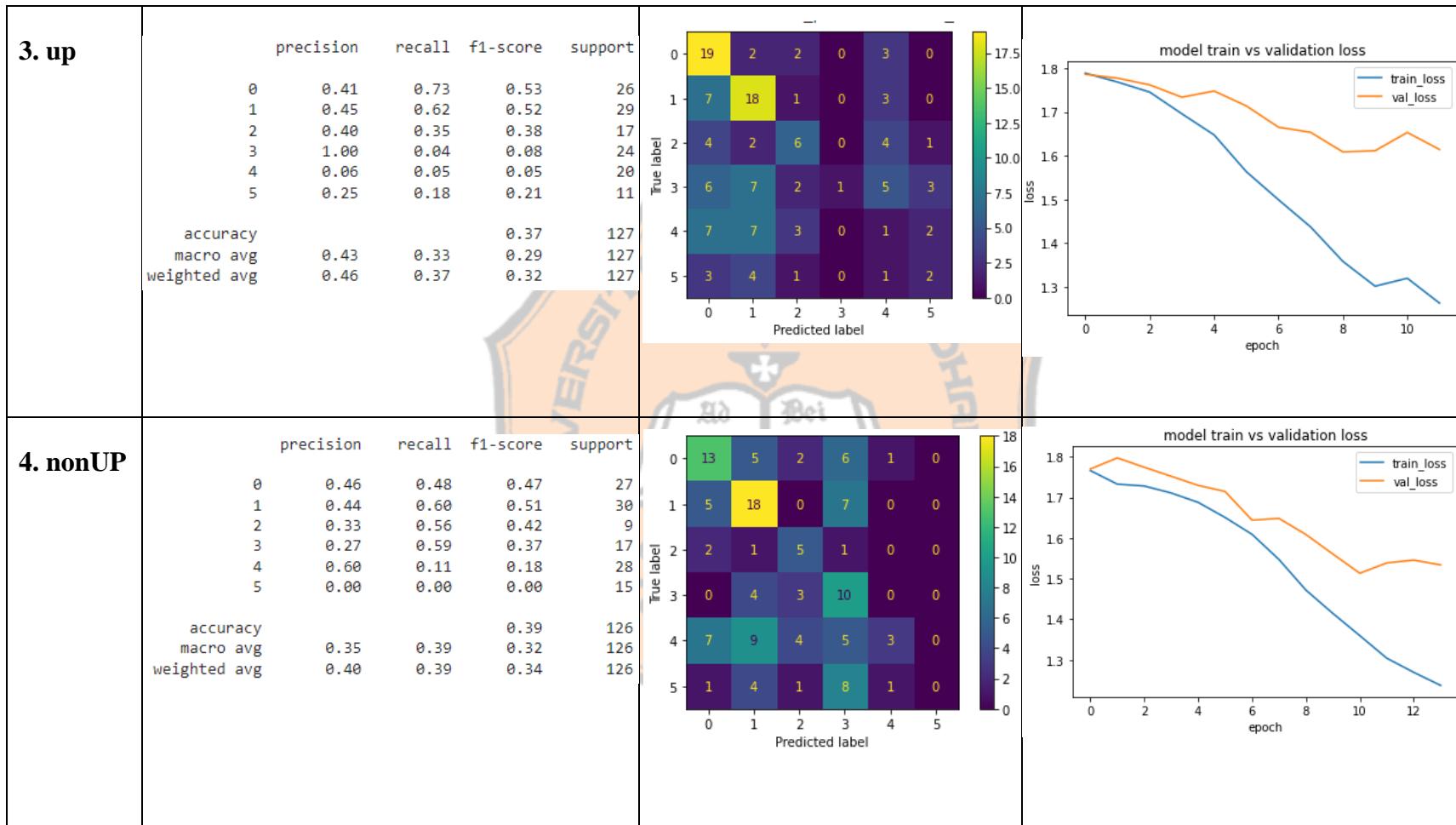
	dataset excel atau csv dengan fitur target (label)	atau ‘.csv’ dengan format kolom yaitu ‘id’, ‘potongan_cerita’, ‘suasana’	dan ‘suasana’ tertampil 2. Tabel probabilitas untuk tiap kelas, dan hasil akhir prediksi (kelas terprediksi) tertampil	dan ‘suasana’ tertampil 2. Tabel probabilitas untuk tiap kelas, dan hasil akhir prediksi (kelas terprediksi) tertampil	3. Laporan klasifikasi dan confusion matrix tertampil	
--	--	--	---	---	---	--

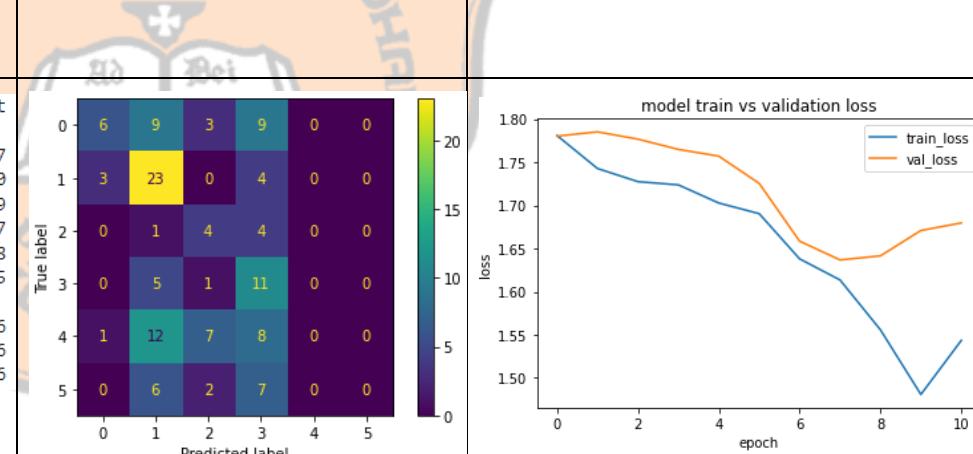
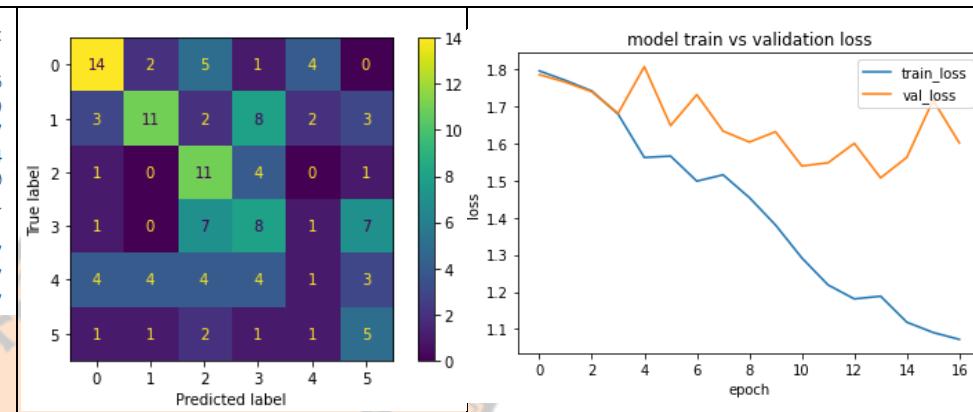
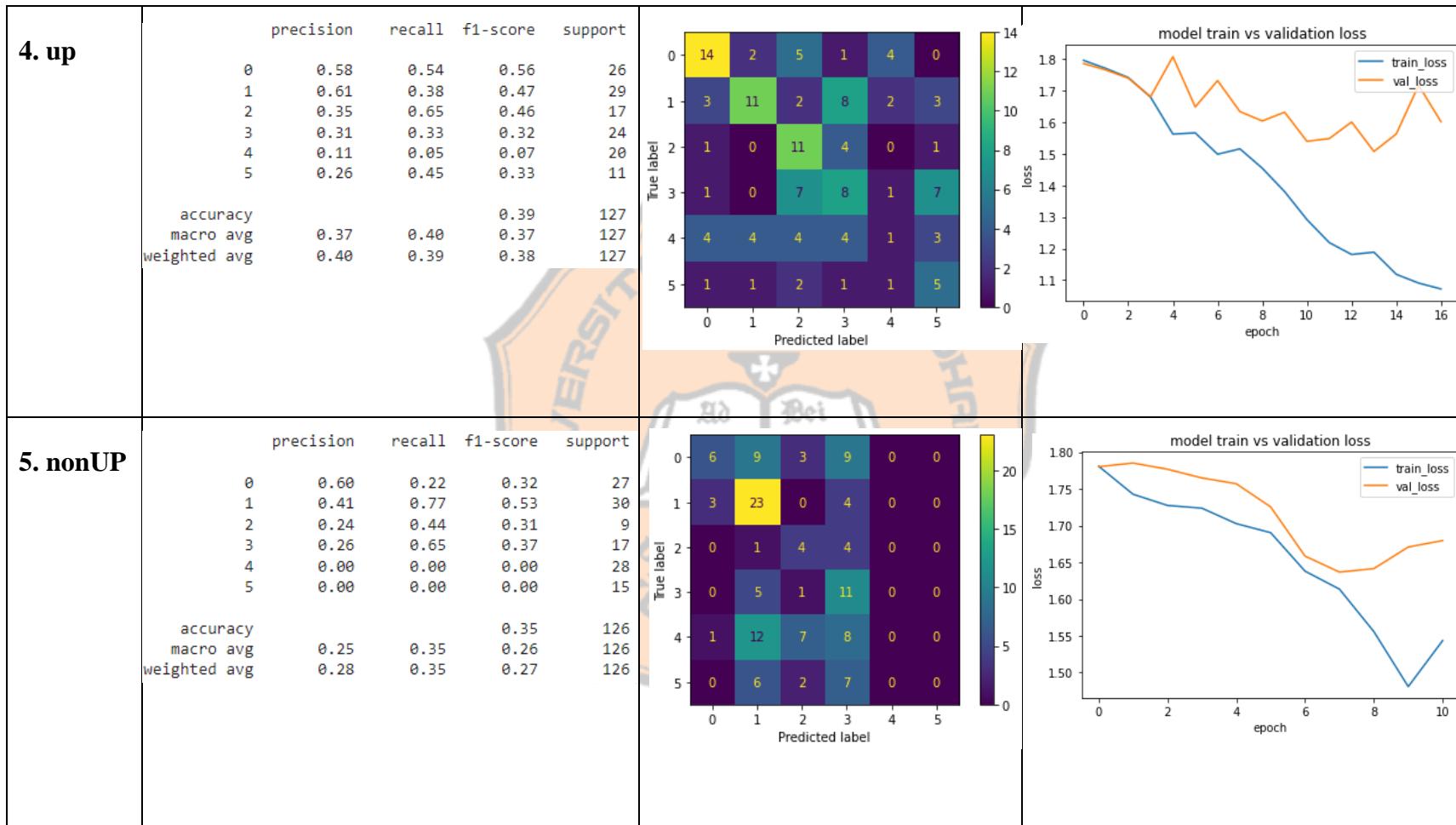
## LAMPIRAN 2: LAPORAN KLASIFIKASI, CONF MATRIX, GRAFIK LOSS

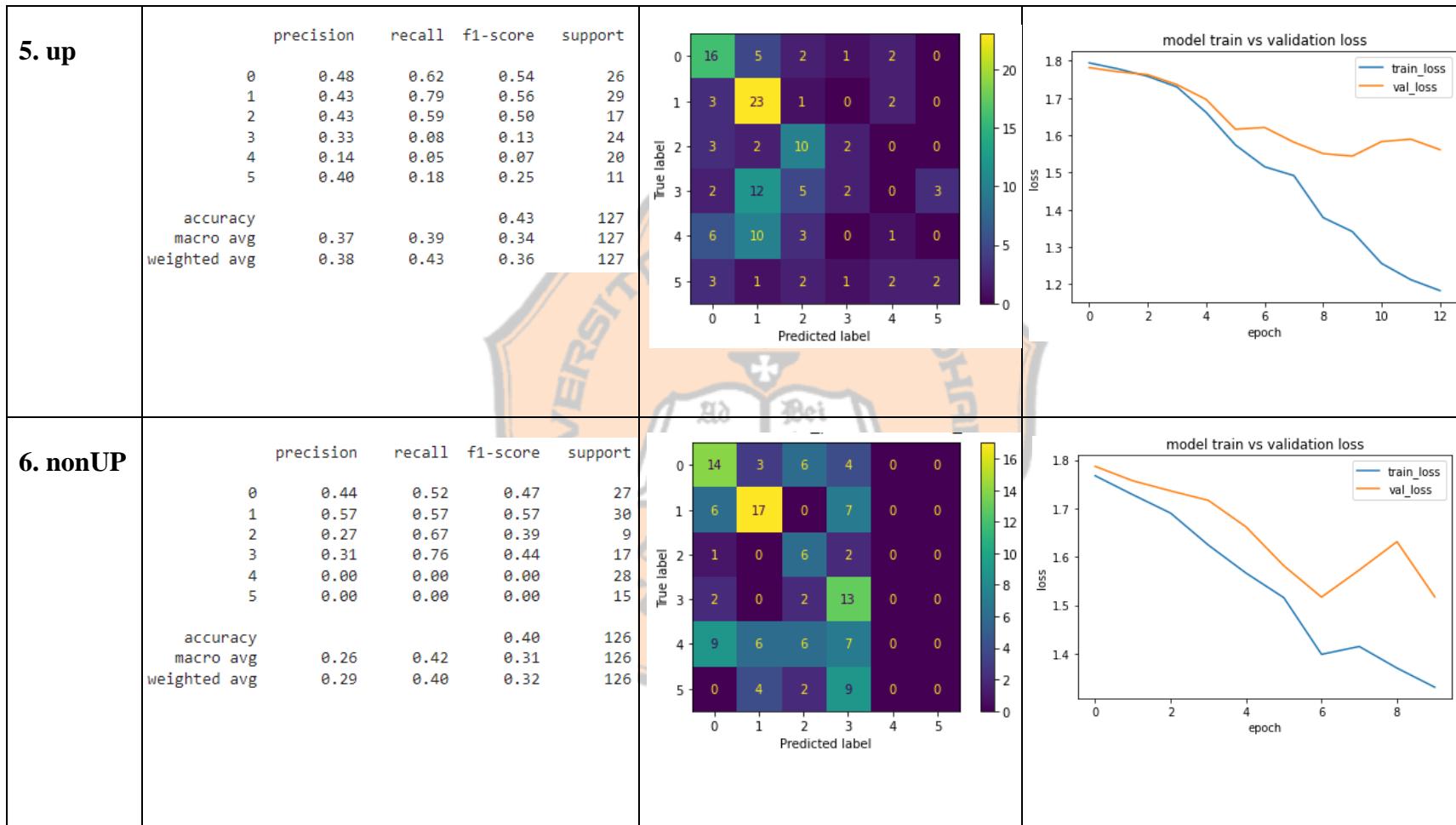
4.2.2.1 Pengujian dengan Dataset Potongan Cerita Arsitektur Sederhana																																																																																																																																		
6 Kelas LSTM	Report		Confusion Matrix		Grafik train-val Loss																																																																																																																													
1. nonUP	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.44</td> <td>0.44</td> <td>0.44</td> <td>27</td> </tr> <tr> <td>1</td> <td>0.48</td> <td>0.70</td> <td>0.57</td> <td>30</td> </tr> <tr> <td>2</td> <td>0.50</td> <td>0.33</td> <td>0.40</td> <td>9</td> </tr> <tr> <td>3</td> <td>0.39</td> <td>0.53</td> <td>0.45</td> <td>17</td> </tr> <tr> <td>4</td> <td>0.25</td> <td>0.21</td> <td>0.23</td> <td>28</td> </tr> <tr> <td>5</td> <td>0.00</td> <td>0.00</td> <td>0.00</td> <td>15</td> </tr> <tr> <td>accuracy</td> <td></td> <td></td> <td>0.40</td> <td>126</td> </tr> <tr> <td>macro avg</td> <td>0.34</td> <td>0.37</td> <td>0.35</td> <td>126</td> </tr> <tr> <td>weighted avg</td> <td>0.35</td> <td>0.40</td> <td>0.37</td> <td>126</td> </tr> </tbody> </table>			precision	recall	f1-score	support	0	0.44	0.44	0.44	27	1	0.48	0.70	0.57	30	2	0.50	0.33	0.40	9	3	0.39	0.53	0.45	17	4	0.25	0.21	0.23	28	5	0.00	0.00	0.00	15	accuracy			0.40	126	macro avg	0.34	0.37	0.35	126	weighted avg	0.35	0.40	0.37	126	<p>Confusion Matrix Data:</p> <table border="1"> <thead> <tr> <th>True label \ Predicted label</th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> </tr> </thead> <tbody> <tr> <th>0</th> <td>12</td> <td>5</td> <td>0</td> <td>1</td> <td>9</td> <td>0</td> </tr> <tr> <th>1</th> <td>3</td> <td>21</td> <td>0</td> <td>3</td> <td>2</td> <td>1</td> </tr> <tr> <th>2</th> <td>1</td> <td>0</td> <td>3</td> <td>1</td> <td>4</td> <td>0</td> </tr> <tr> <th>3</th> <td>1</td> <td>3</td> <td>1</td> <td>9</td> <td>2</td> <td>1</td> </tr> <tr> <th>4</th> <td>10</td> <td>10</td> <td>1</td> <td>1</td> <td>6</td> <td>0</td> </tr> <tr> <th>5</th> <td>0</td> <td>5</td> <td>1</td> <td>8</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	True label \ Predicted label	0	1	2	3	4	5	0	12	5	0	1	9	0	1	3	21	0	3	2	1	2	1	0	3	1	4	0	3	1	3	1	9	2	1	4	10	10	1	1	6	0	5	0	5	1	8	1	0	<p>model train vs validation loss</p> <table border="1"> <thead> <tr> <th>epoch</th> <th>train_loss</th> <th>val_loss</th> </tr> </thead> <tbody> <tr><td>0.0</td><td>1.75</td><td>1.78</td></tr> <tr><td>2.5</td><td>1.68</td><td>1.75</td></tr> <tr><td>5.0</td><td>1.55</td><td>1.72</td></tr> <tr><td>7.5</td><td>1.45</td><td>1.65</td></tr> <tr><td>10.0</td><td>1.35</td><td>1.52</td></tr> <tr><td>12.5</td><td>1.28</td><td>1.50</td></tr> <tr><td>15.0</td><td>1.30</td><td>1.45</td></tr> <tr><td>17.5</td><td>1.22</td><td>1.58</td></tr> </tbody> </table>	epoch	train_loss	val_loss	0.0	1.75	1.78	2.5	1.68	1.75	5.0	1.55	1.72	7.5	1.45	1.65	10.0	1.35	1.52	12.5	1.28	1.50	15.0	1.30	1.45	17.5	1.22	1.58
	precision	recall	f1-score	support																																																																																																																														
0	0.44	0.44	0.44	27																																																																																																																														
1	0.48	0.70	0.57	30																																																																																																																														
2	0.50	0.33	0.40	9																																																																																																																														
3	0.39	0.53	0.45	17																																																																																																																														
4	0.25	0.21	0.23	28																																																																																																																														
5	0.00	0.00	0.00	15																																																																																																																														
accuracy			0.40	126																																																																																																																														
macro avg	0.34	0.37	0.35	126																																																																																																																														
weighted avg	0.35	0.40	0.37	126																																																																																																																														
True label \ Predicted label	0	1	2	3	4	5																																																																																																																												
0	12	5	0	1	9	0																																																																																																																												
1	3	21	0	3	2	1																																																																																																																												
2	1	0	3	1	4	0																																																																																																																												
3	1	3	1	9	2	1																																																																																																																												
4	10	10	1	1	6	0																																																																																																																												
5	0	5	1	8	1	0																																																																																																																												
epoch	train_loss	val_loss																																																																																																																																
0.0	1.75	1.78																																																																																																																																
2.5	1.68	1.75																																																																																																																																
5.0	1.55	1.72																																																																																																																																
7.5	1.45	1.65																																																																																																																																
10.0	1.35	1.52																																																																																																																																
12.5	1.28	1.50																																																																																																																																
15.0	1.30	1.45																																																																																																																																
17.5	1.22	1.58																																																																																																																																

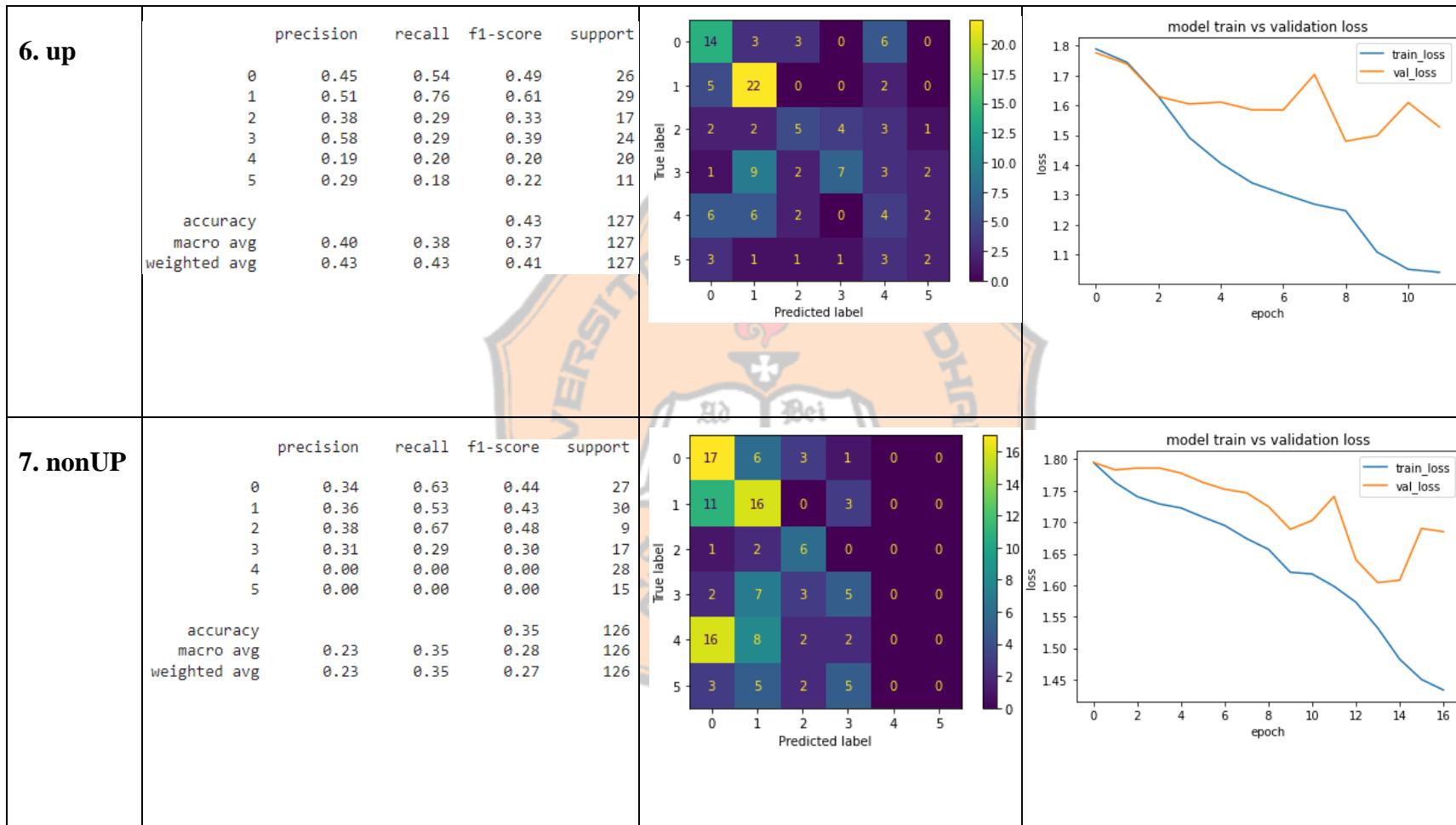


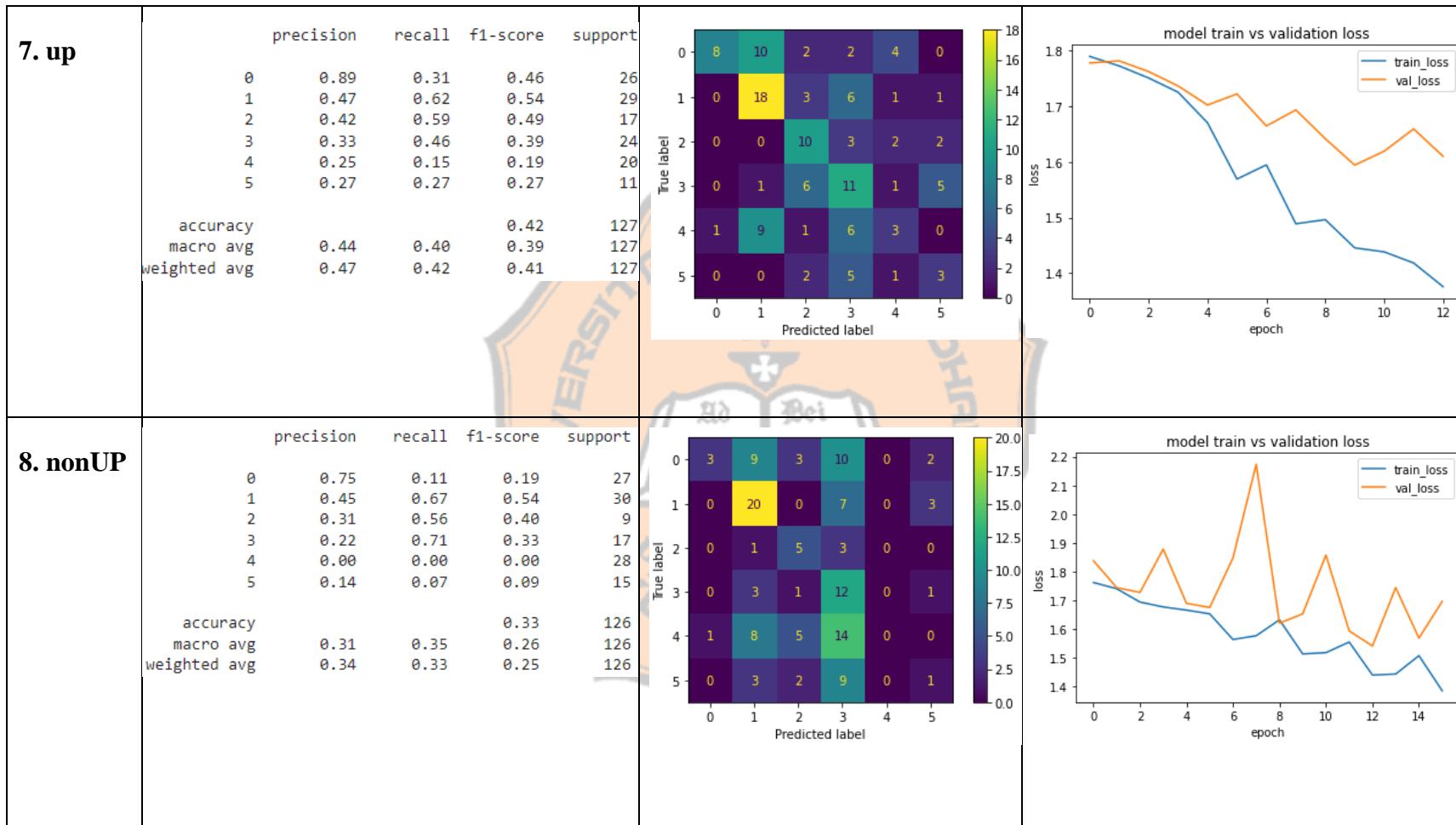


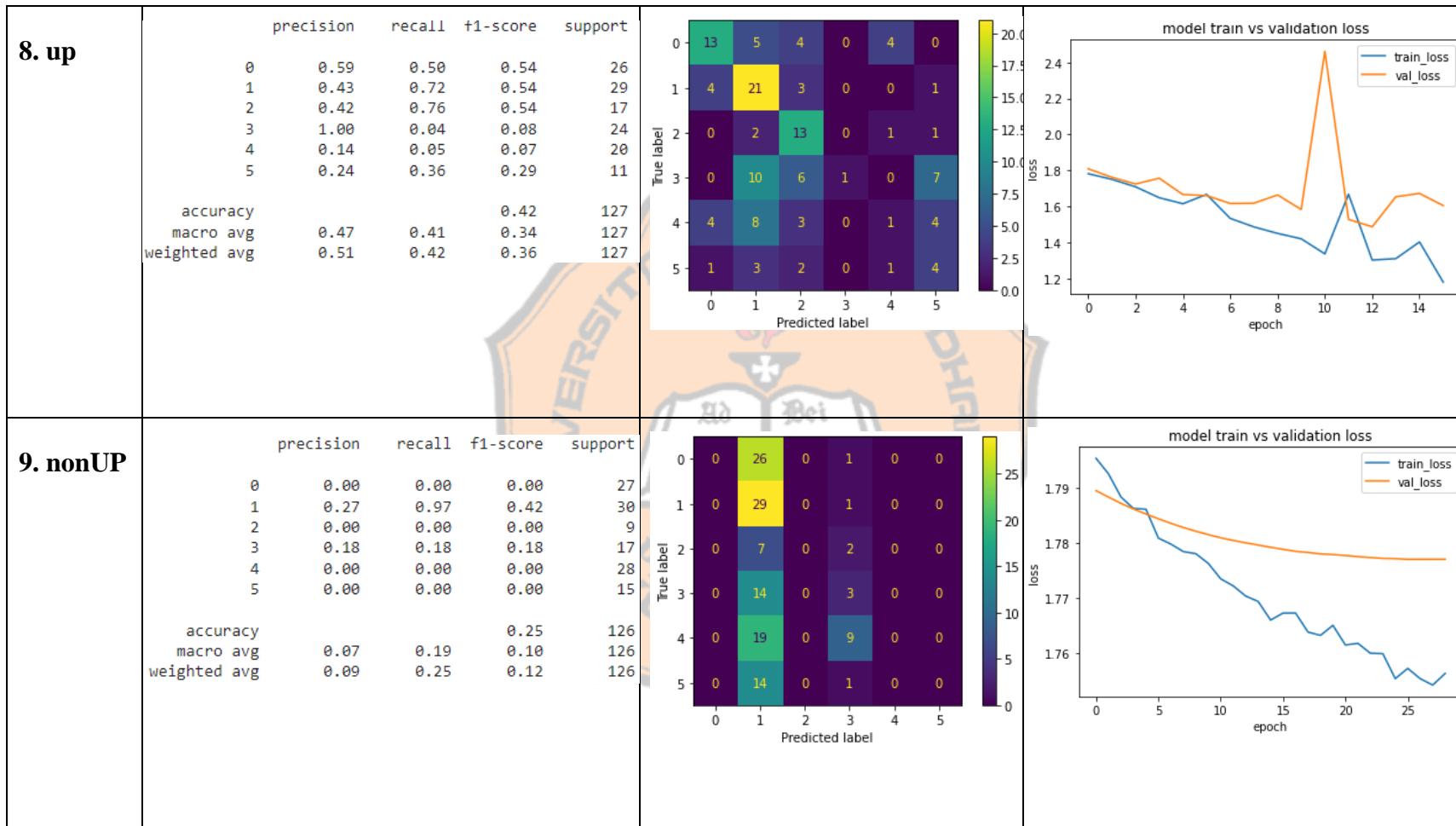


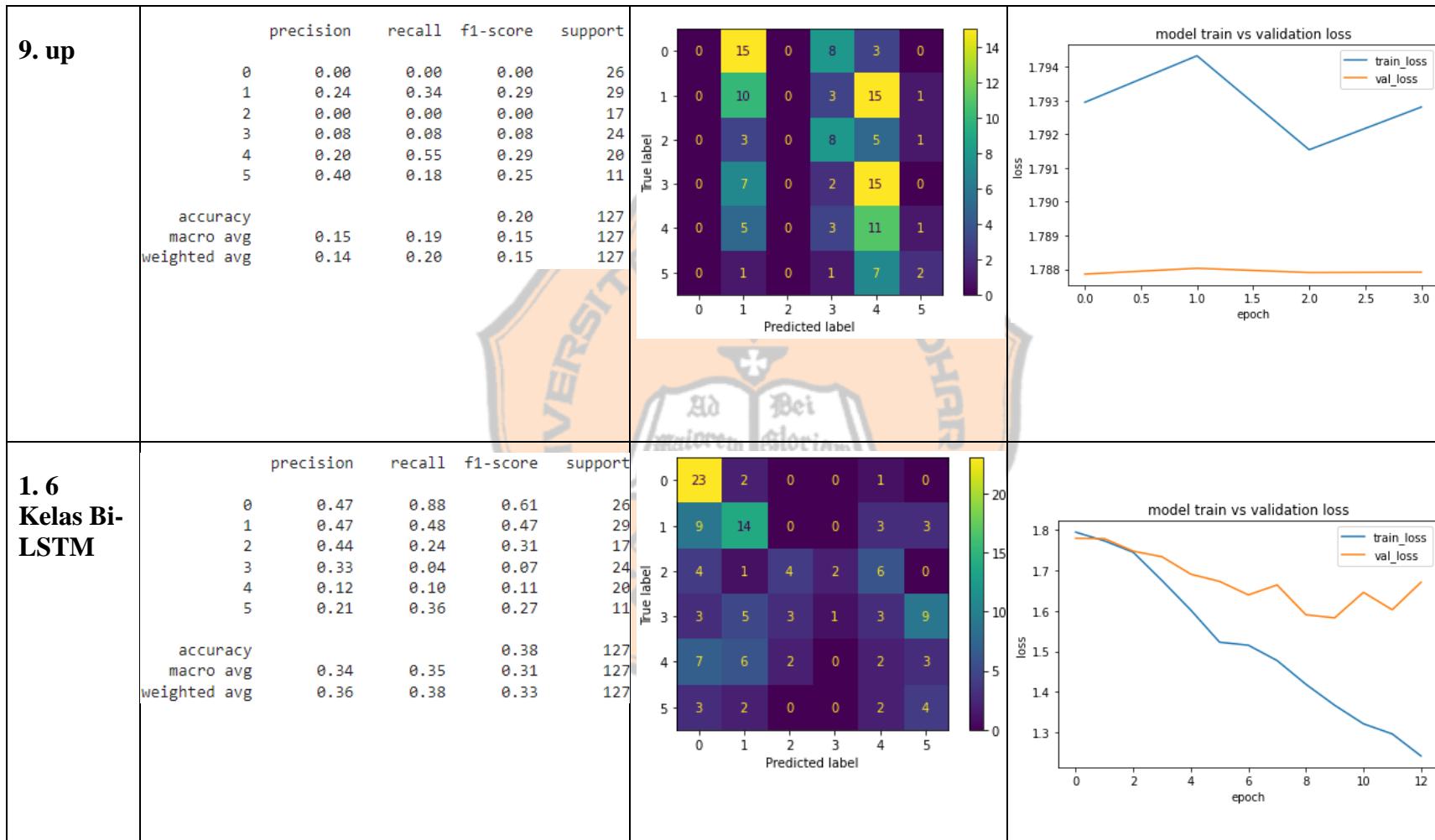


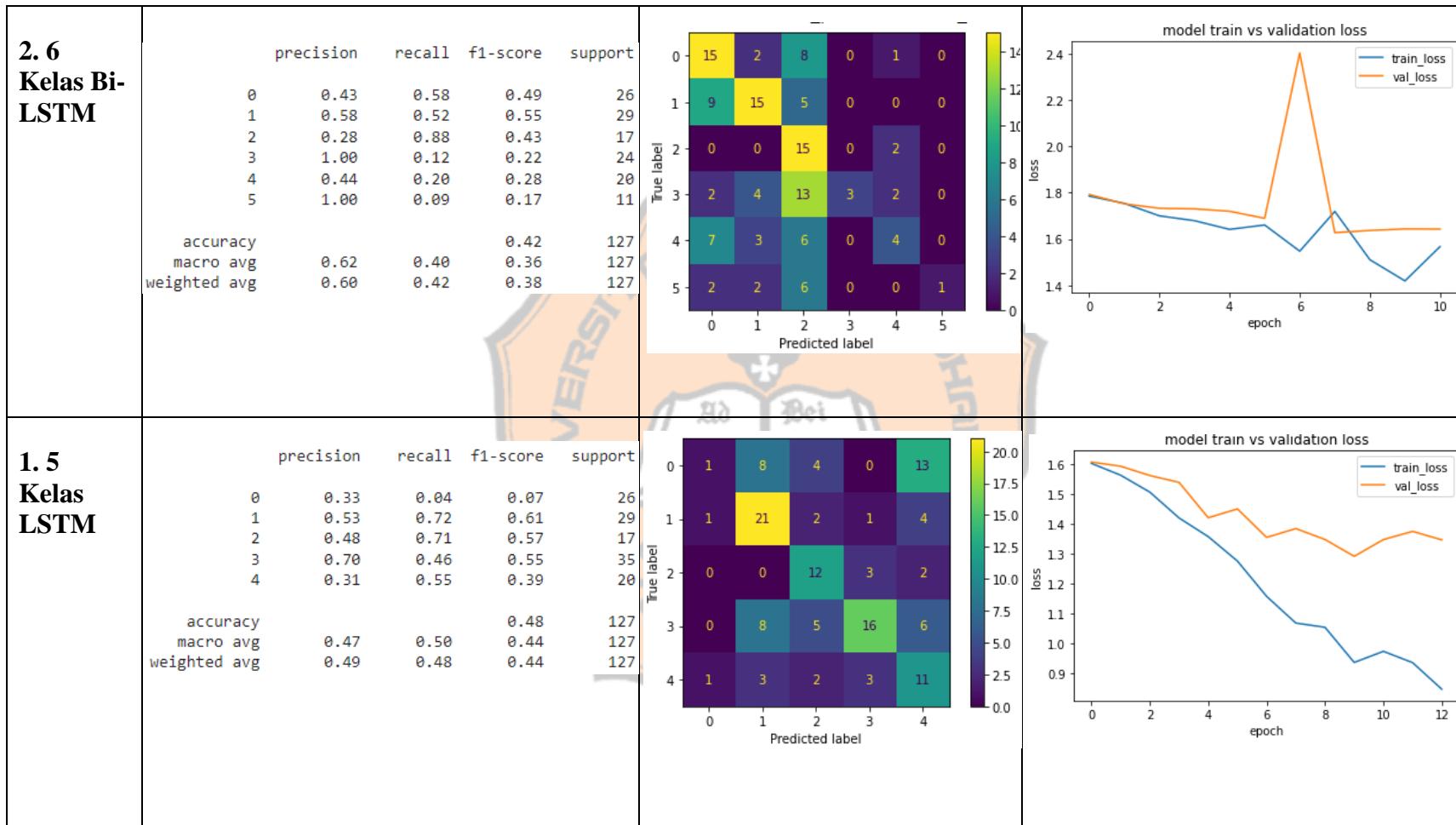


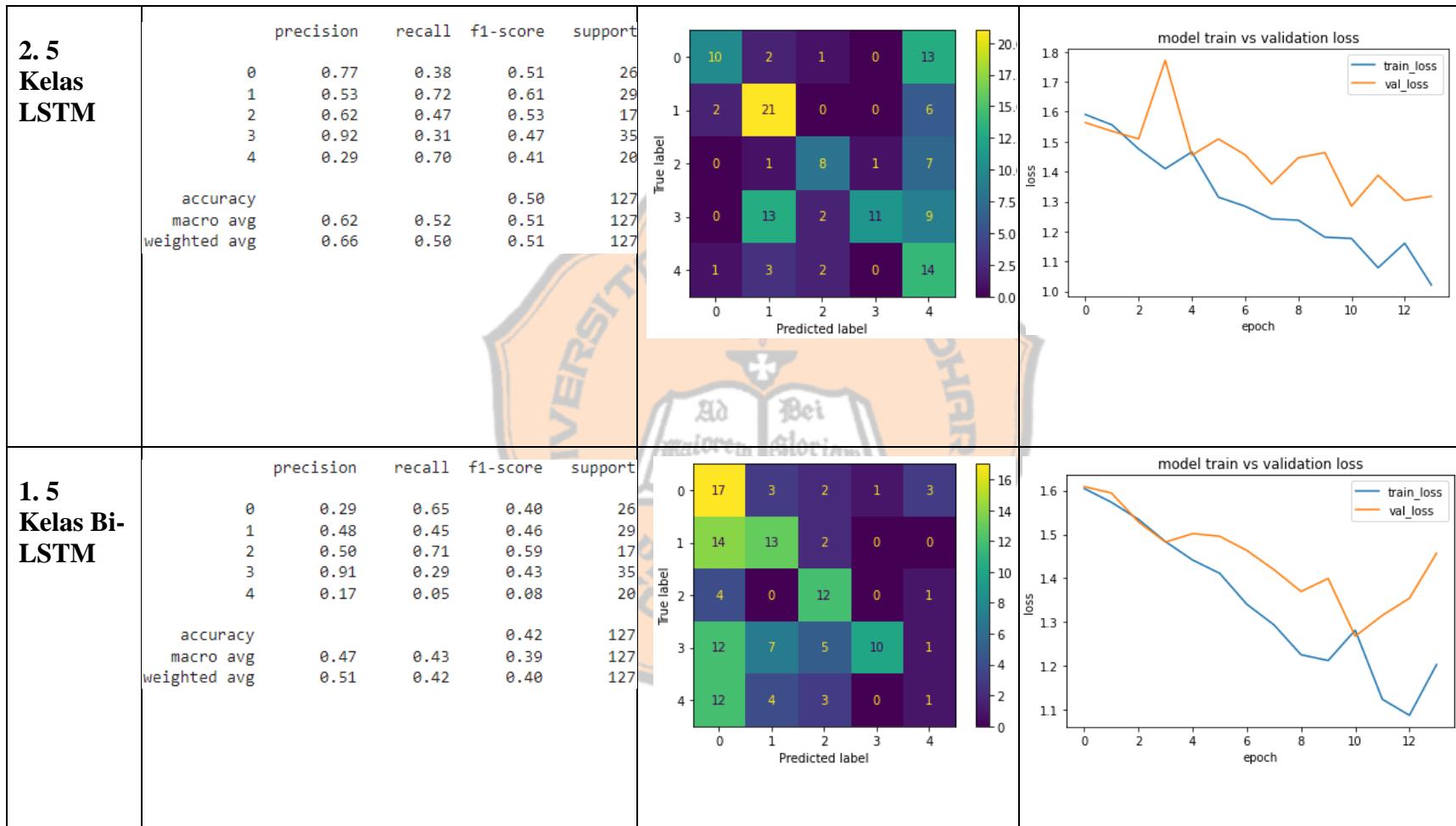


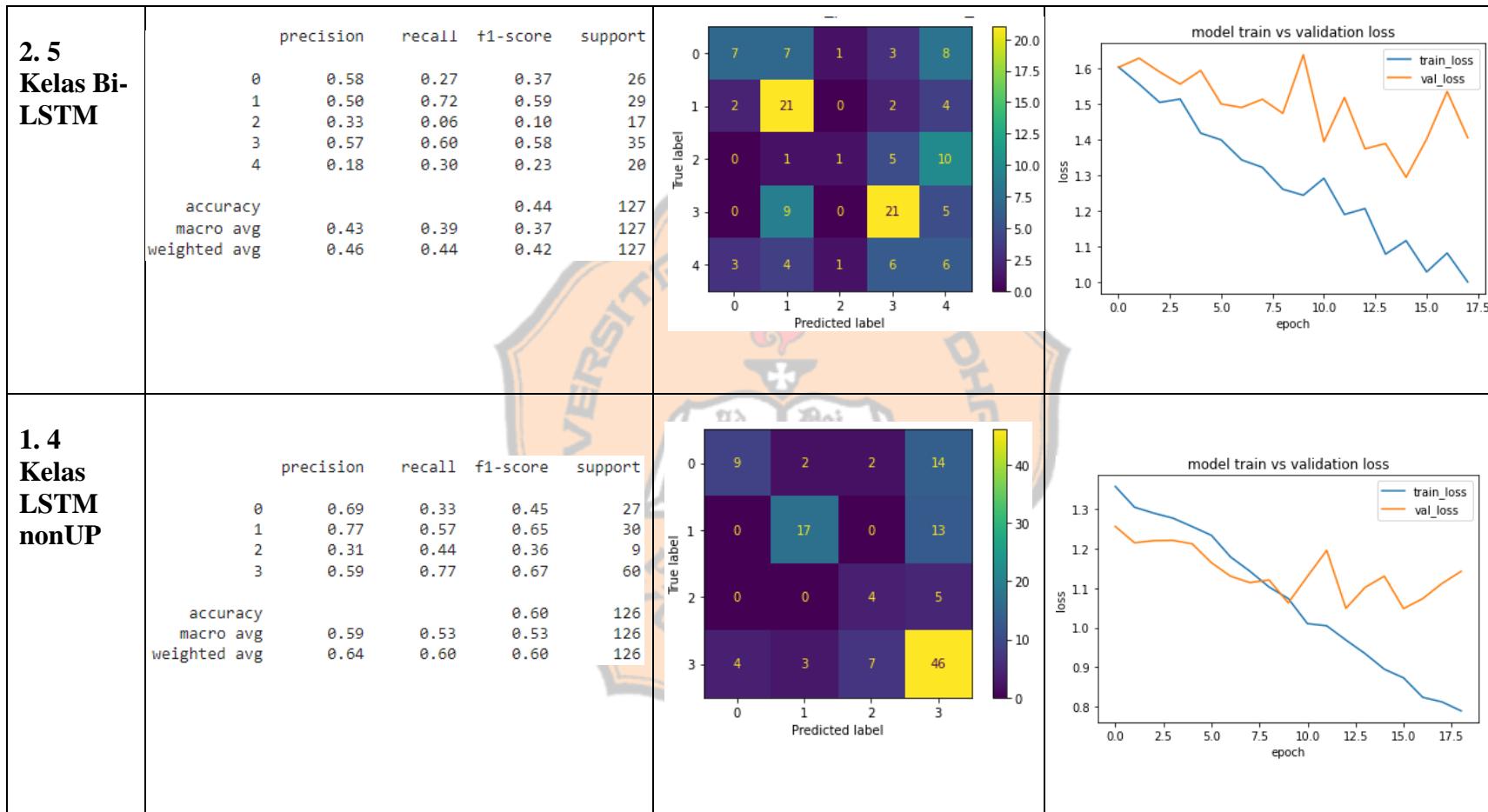


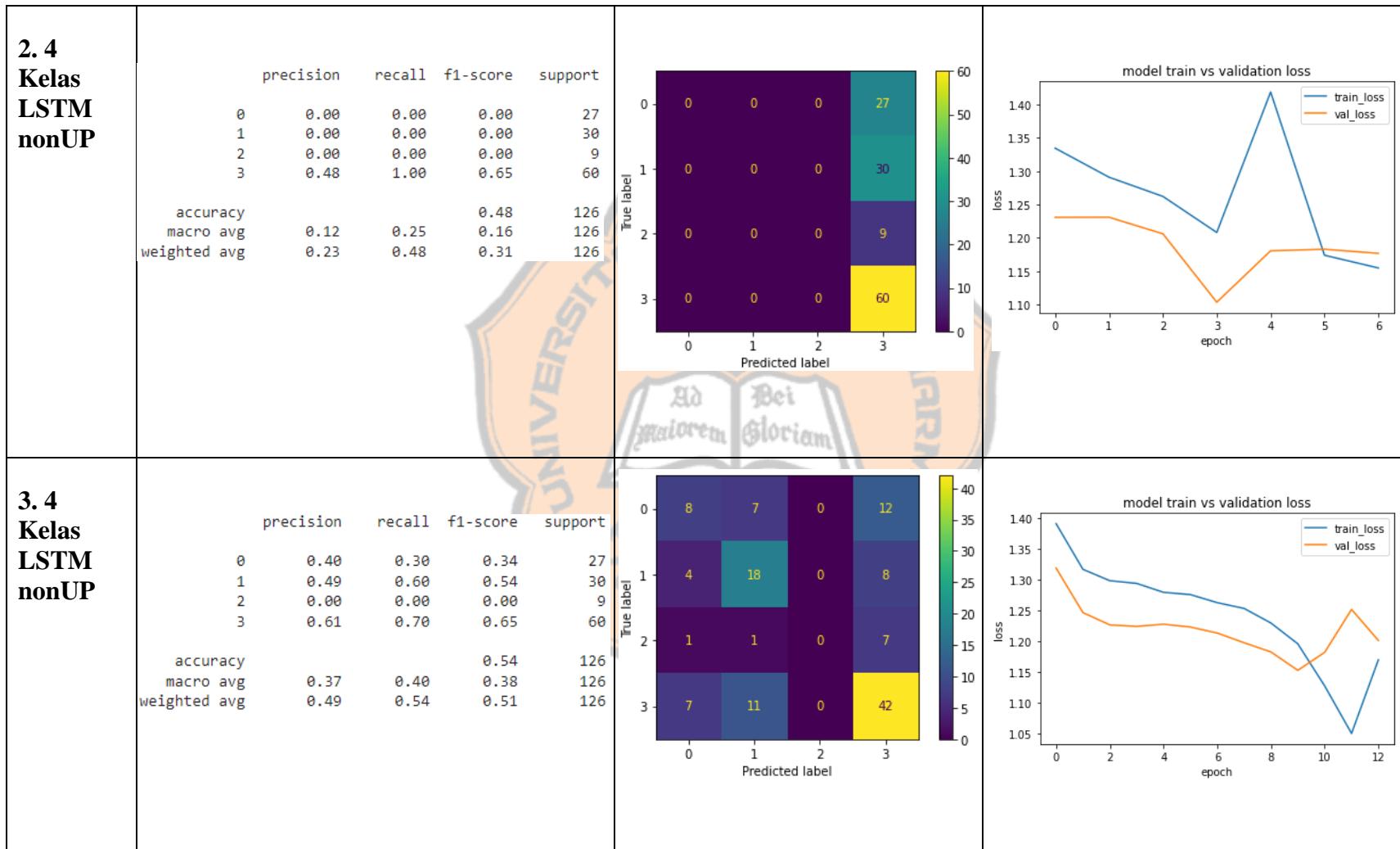


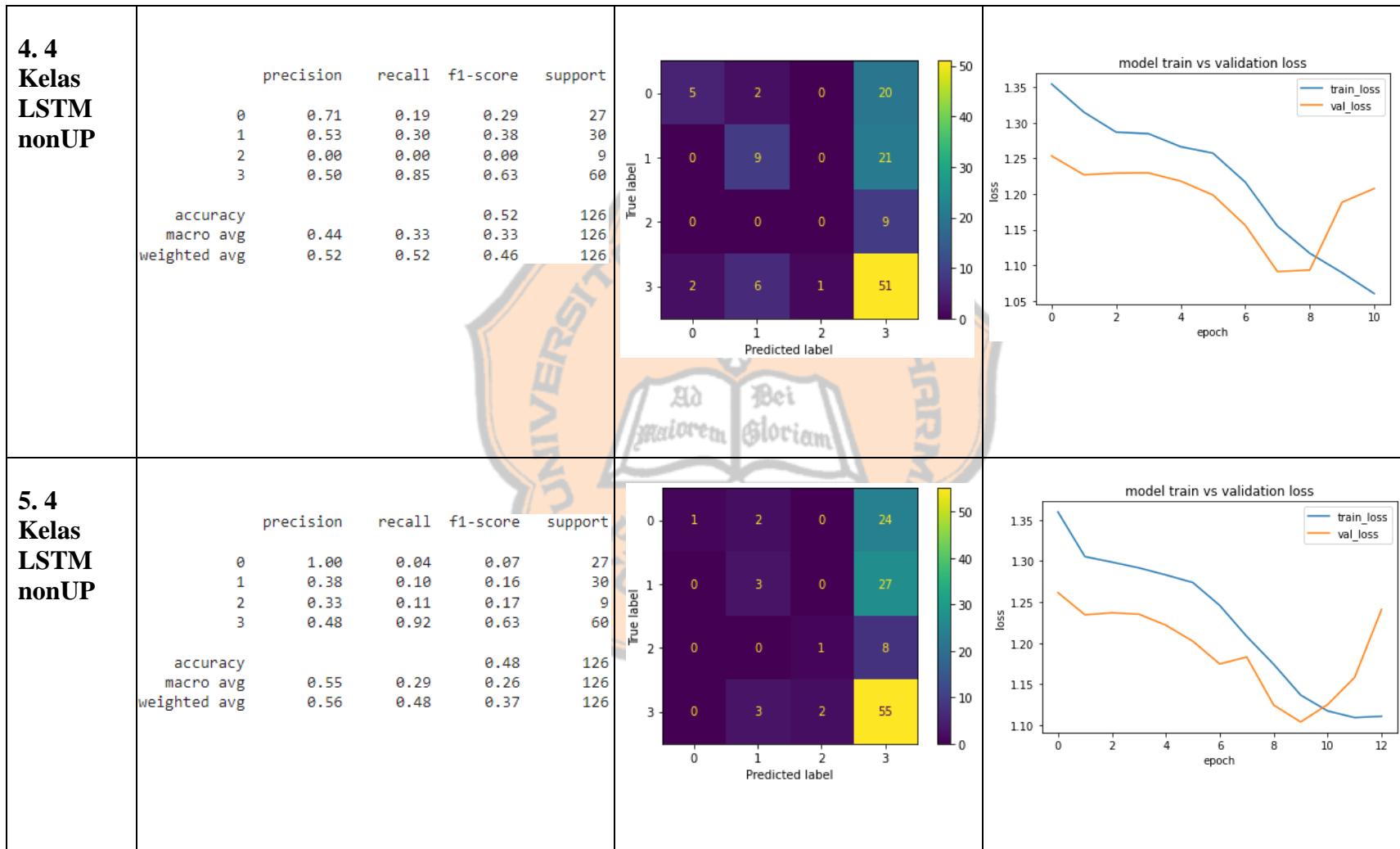


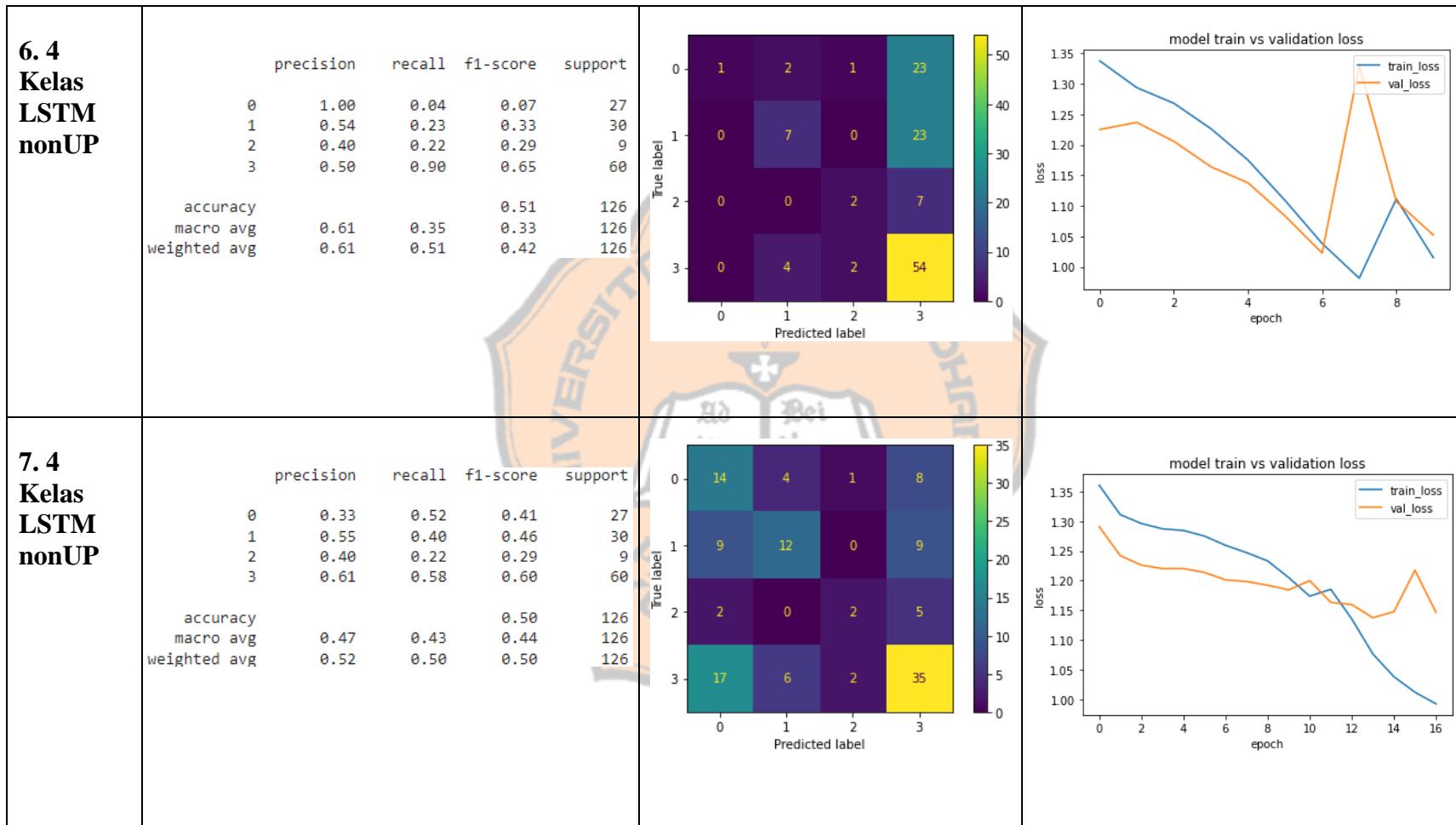


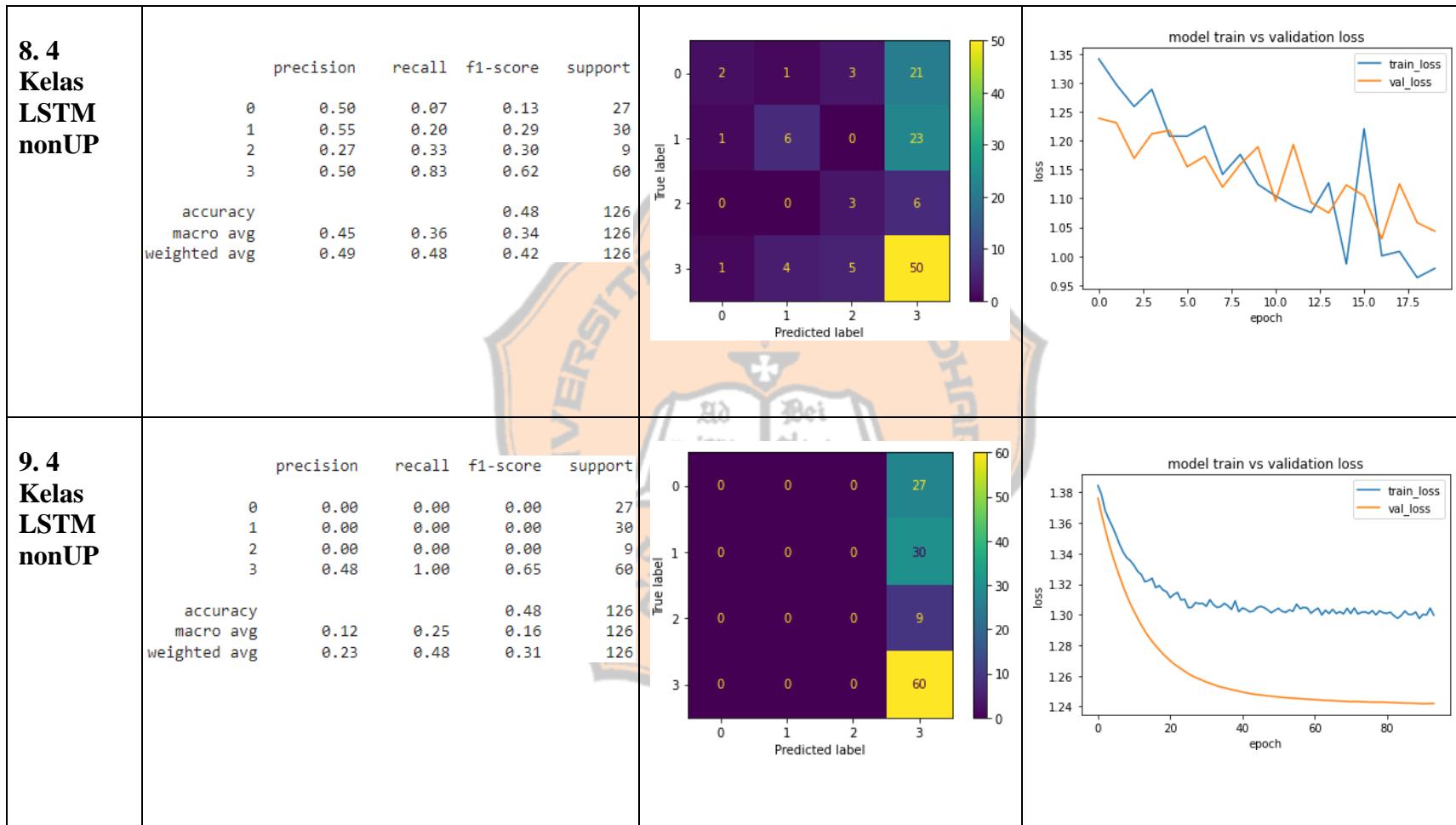


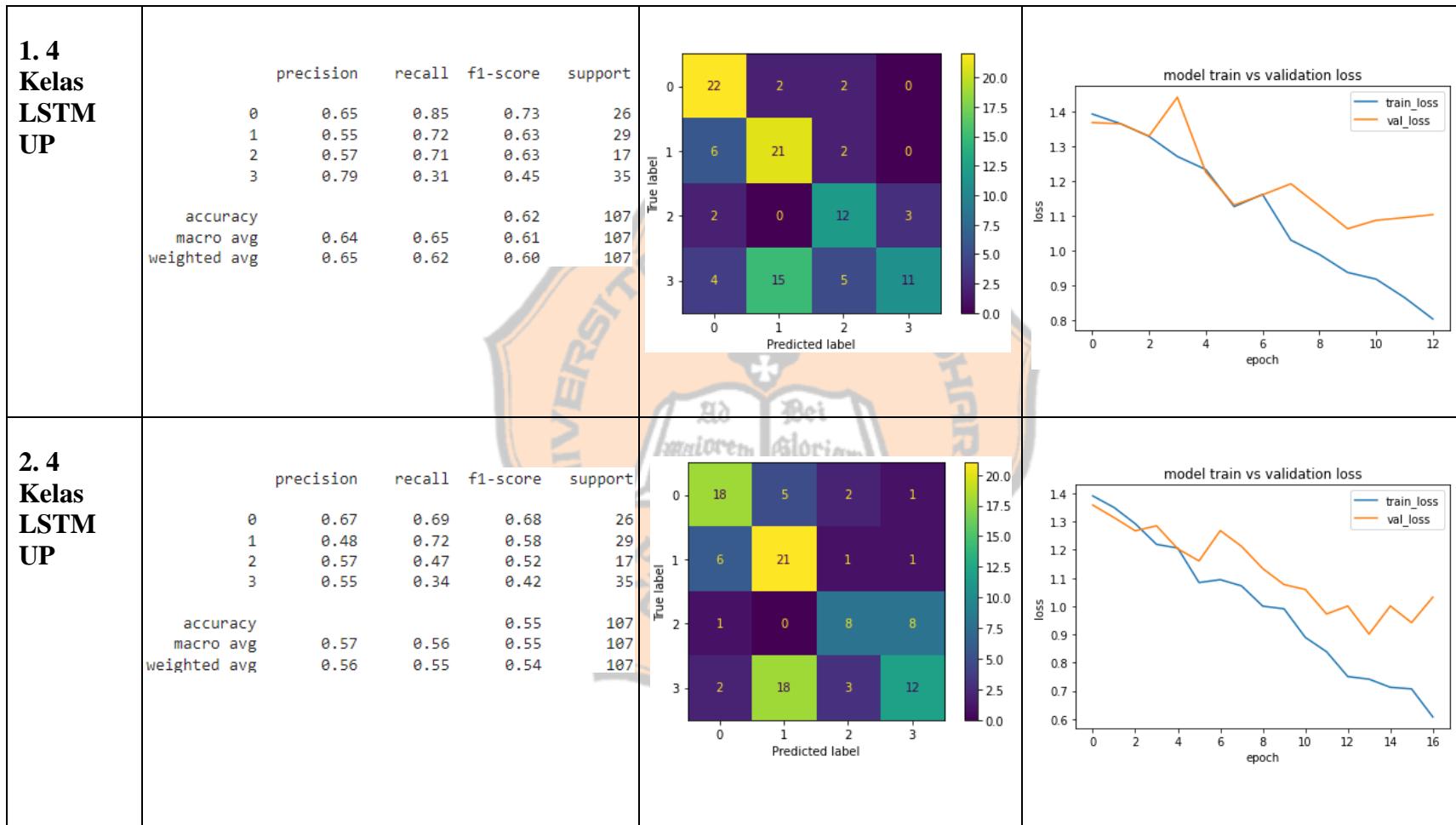


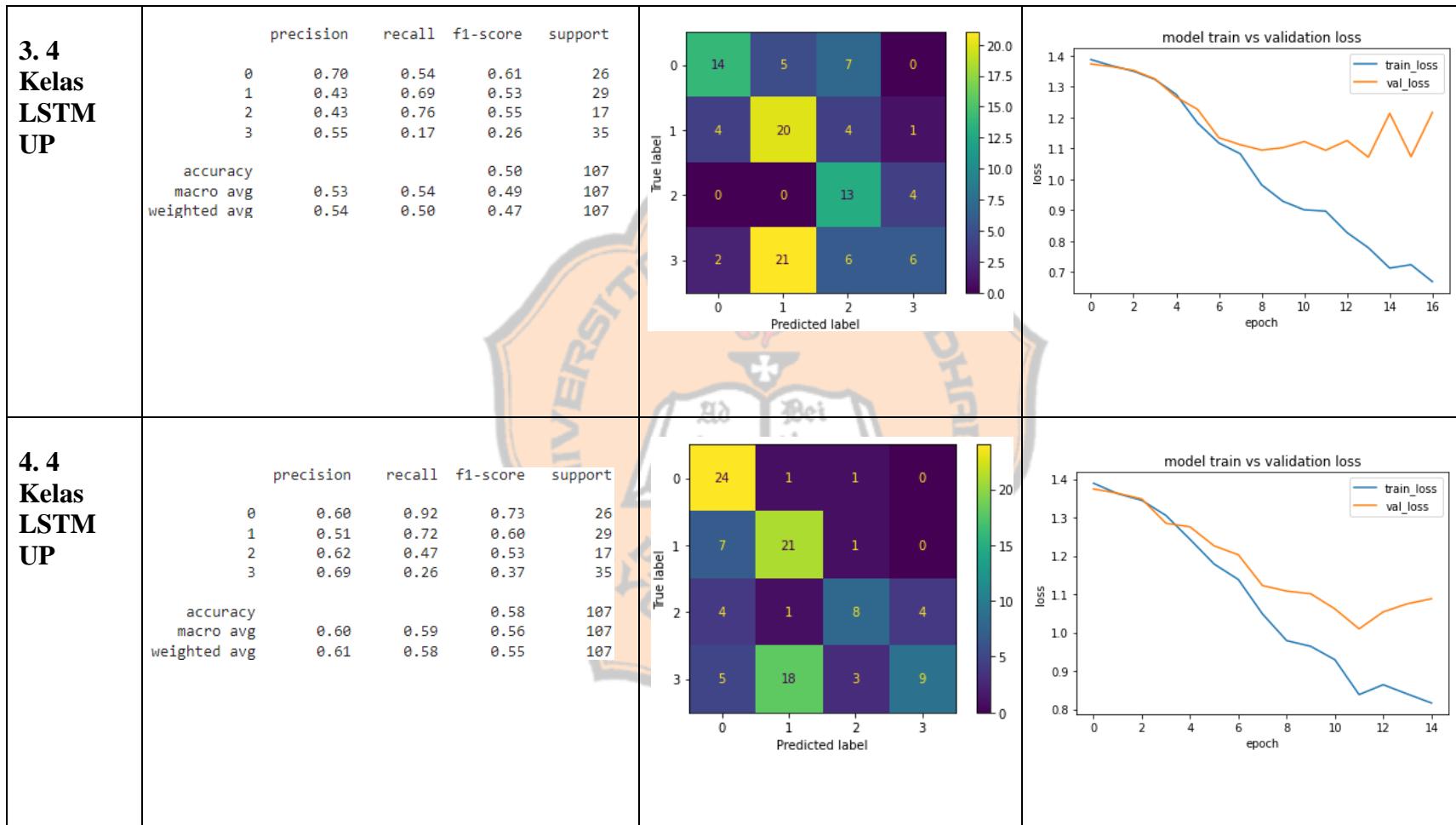


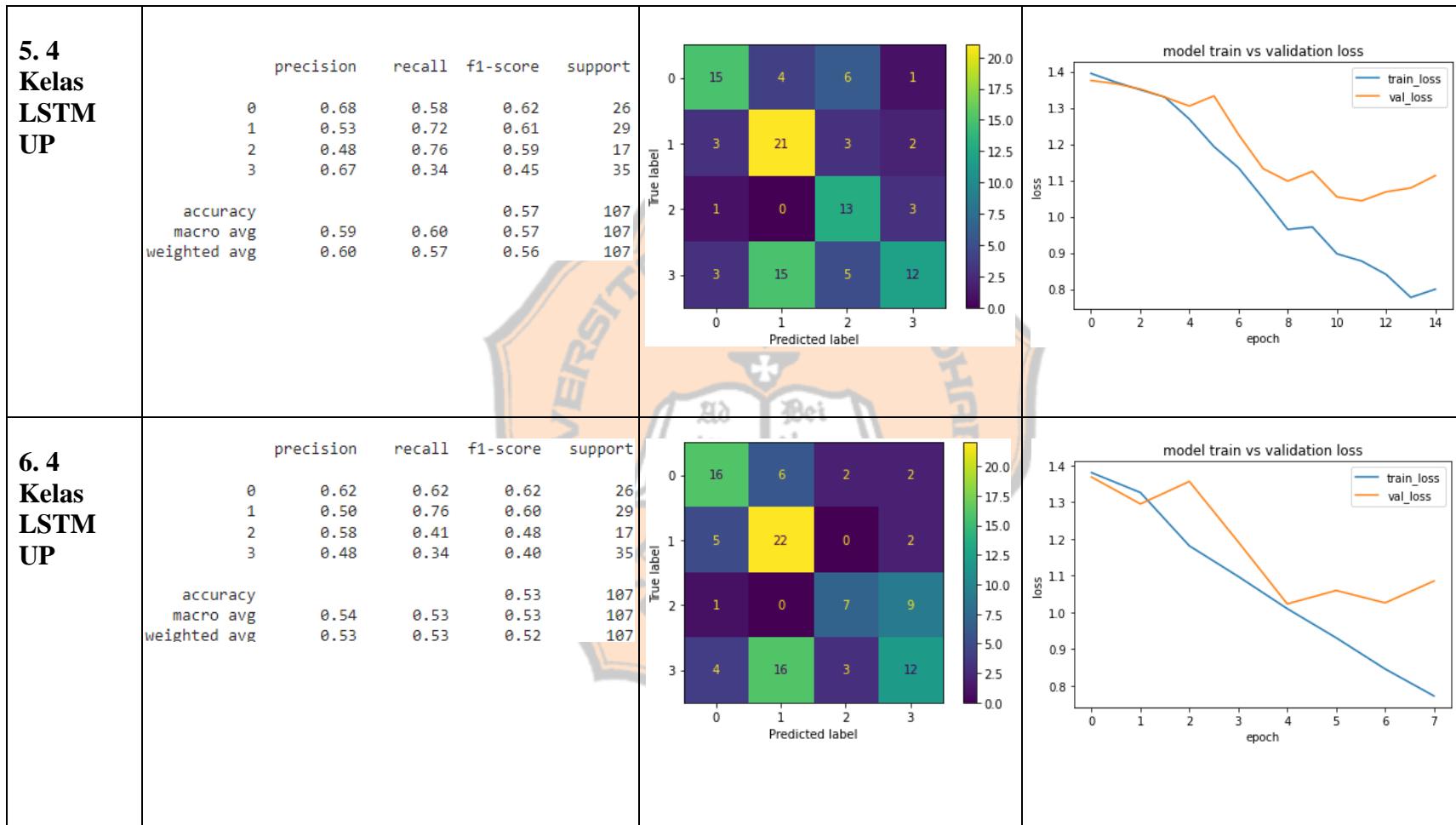


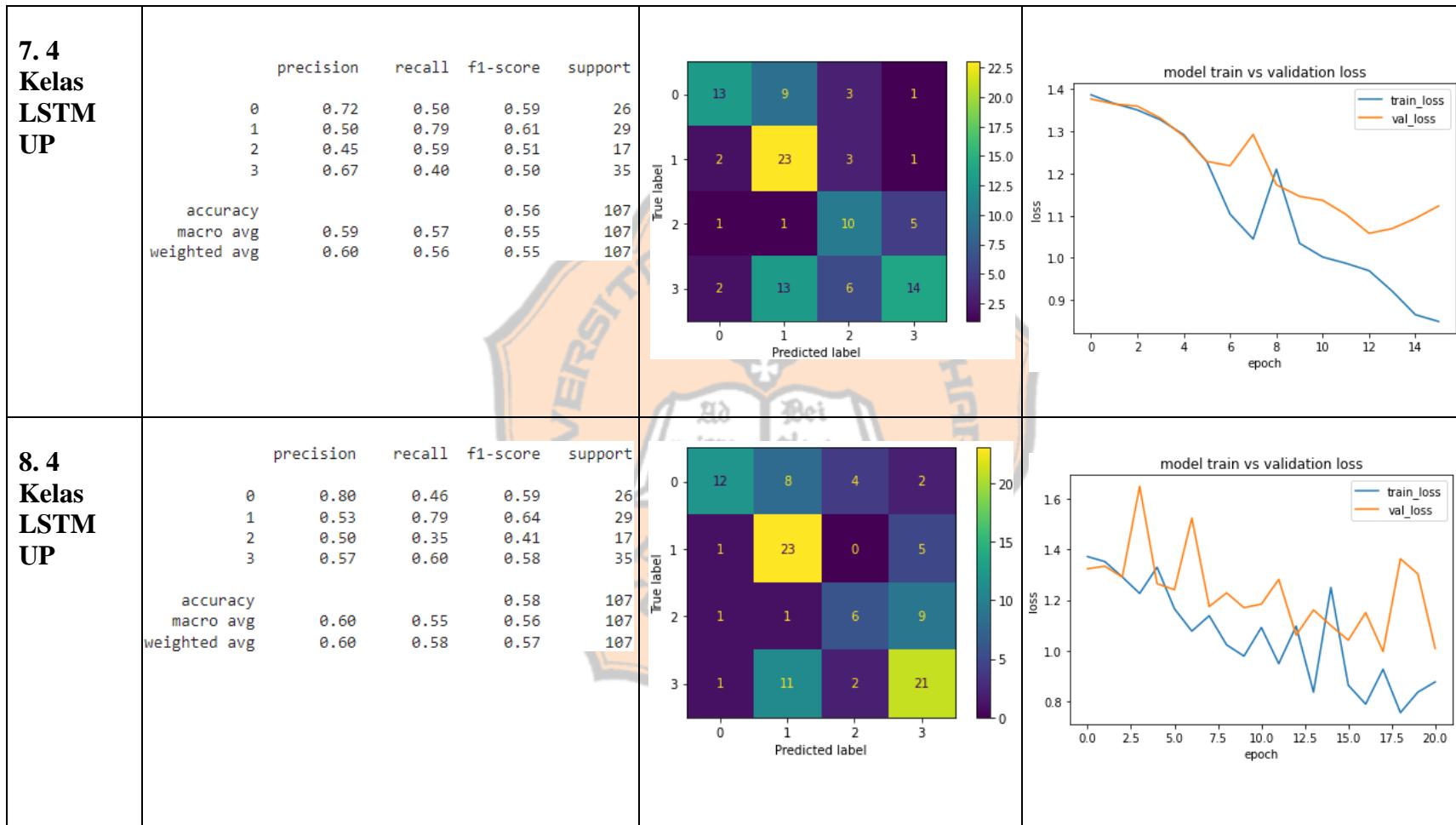


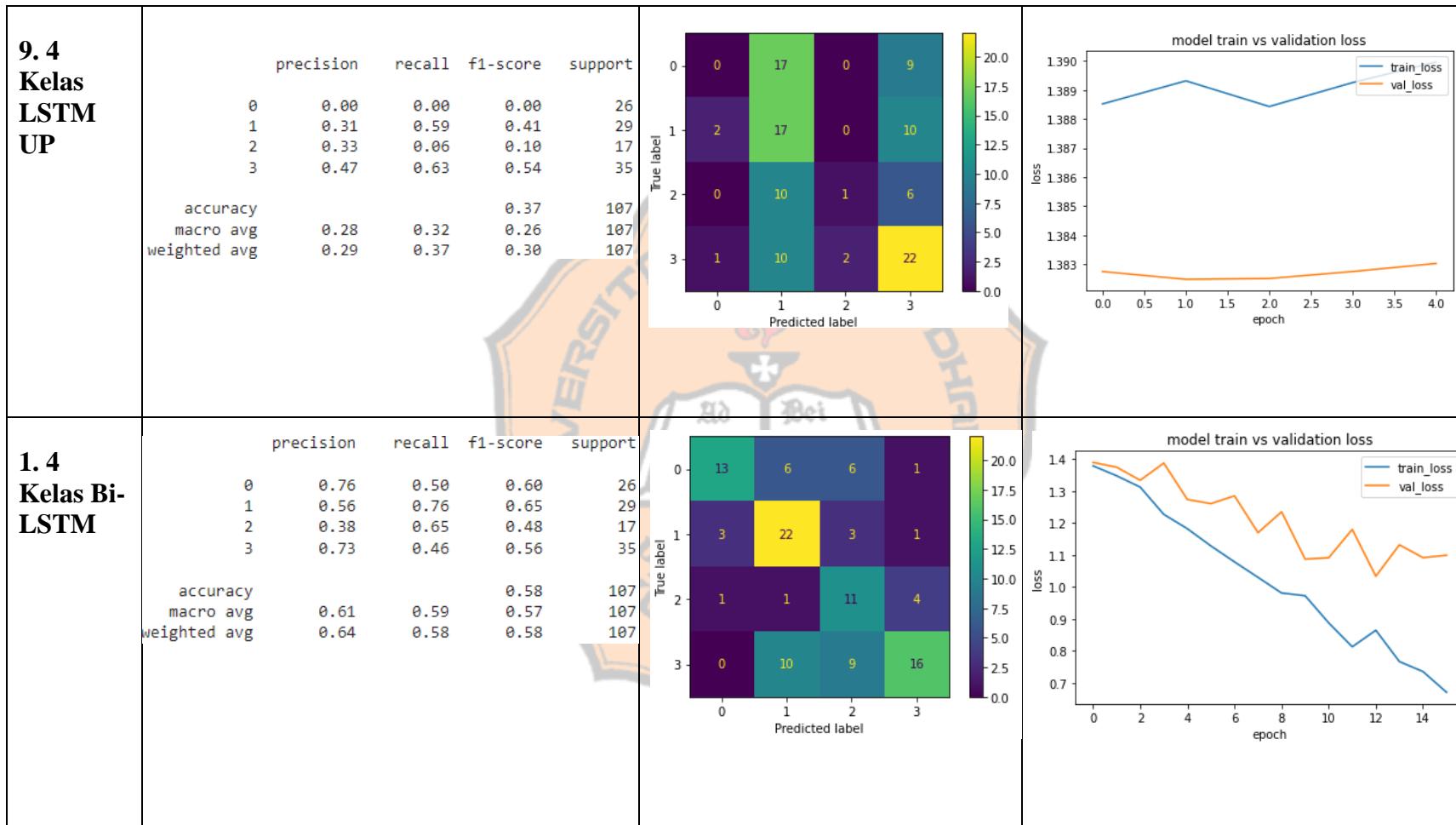






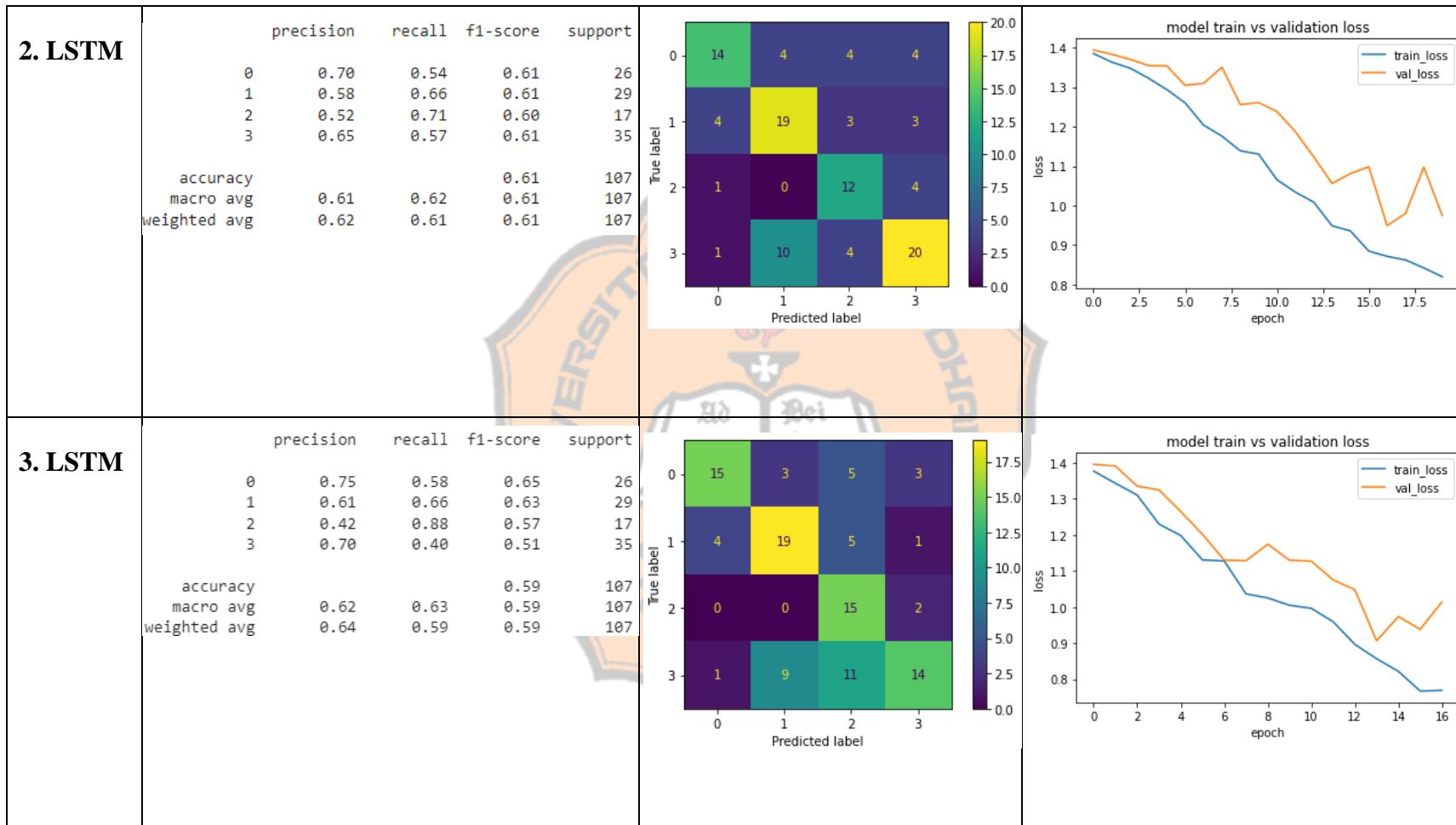


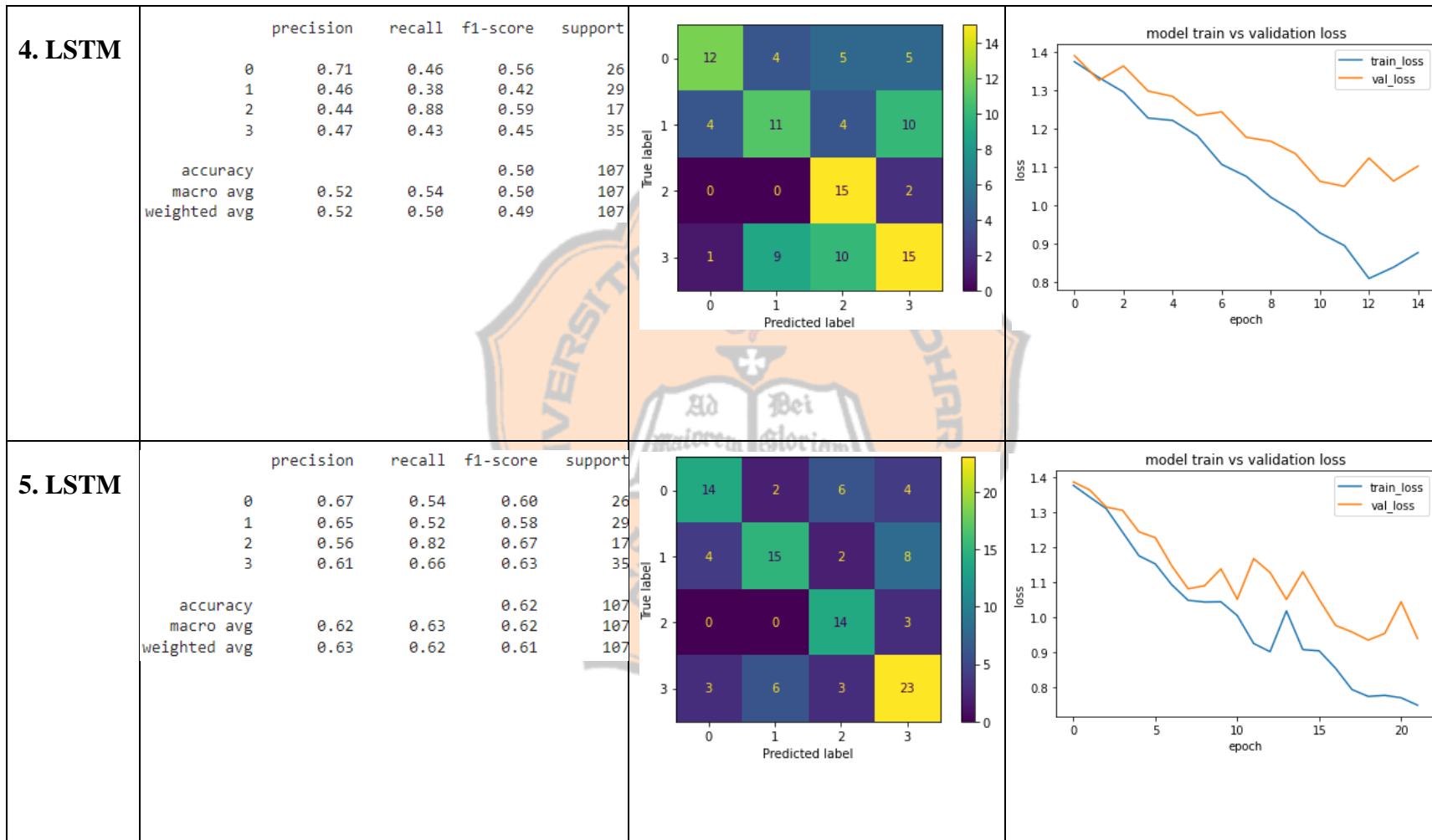




<b>2.4 Kelas Bi-LSTM</b>	precision	recall	f1-score	support	<table border="1"> <thead> <tr> <th></th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> </tr> </thead> <tbody> <tr> <th>0</th> <td>8</td> <td>15</td> <td>3</td> <td>0</td> </tr> <tr> <th>1</th> <td>1</td> <td>25</td> <td>3</td> <td>0</td> </tr> <tr> <th>2</th> <td>1</td> <td>5</td> <td>10</td> <td>1</td> </tr> <tr> <th>3</th> <td>0</td> <td>20</td> <td>4</td> <td>11</td> </tr> </tbody> </table>		0	1	2	3	0	8	15	3	0	1	1	25	3	0	2	1	5	10	1	3	0	20	4	11	<table border="1"> <thead> <tr> <th>epoch</th> <th>train_loss</th> <th>val_loss</th> </tr> </thead> <tbody> <tr><td>0.0</td><td>14.0</td><td>14.0</td></tr> <tr><td>2.5</td><td>12.5</td><td>14.5</td></tr> <tr><td>5.0</td><td>11.5</td><td>13.0</td></tr> <tr><td>7.5</td><td>10.5</td><td>12.5</td></tr> <tr><td>10.0</td><td>10.0</td><td>11.5</td></tr> <tr><td>12.5</td><td>10.5</td><td>12.5</td></tr> <tr><td>15.0</td><td>9.5</td><td>10.5</td></tr> <tr><td>17.5</td><td>9.5</td><td>12.0</td></tr> </tbody> </table>	epoch	train_loss	val_loss	0.0	14.0	14.0	2.5	12.5	14.5	5.0	11.5	13.0	7.5	10.5	12.5	10.0	10.0	11.5	12.5	10.5	12.5	15.0	9.5	10.5	17.5	9.5	12.0
	0	1	2	3																																																						
0	8	15	3	0																																																						
1	1	25	3	0																																																						
2	1	5	10	1																																																						
3	0	20	4	11																																																						
epoch	train_loss	val_loss																																																								
0.0	14.0	14.0																																																								
2.5	12.5	14.5																																																								
5.0	11.5	13.0																																																								
7.5	10.5	12.5																																																								
10.0	10.0	11.5																																																								
12.5	10.5	12.5																																																								
15.0	9.5	10.5																																																								
17.5	9.5	12.0																																																								
accuracy		0.50	107																																																							
<b>1. LSTM</b>	macro avg	0.52	0.50	107	<table border="1"> <thead> <tr> <th></th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> </tr> </thead> <tbody> <tr> <th>0</th> <td>7</td> <td>11</td> <td>6</td> <td>2</td> </tr> <tr> <th>1</th> <td>2</td> <td>24</td> <td>2</td> <td>1</td> </tr> <tr> <th>2</th> <td>2</td> <td>0</td> <td>10</td> <td>5</td> </tr> <tr> <th>3</th> <td>1</td> <td>11</td> <td>4</td> <td>19</td> </tr> </tbody> </table>		0	1	2	3	0	7	11	6	2	1	2	24	2	1	2	2	0	10	5	3	1	11	4	19	<table border="1"> <thead> <tr> <th>epoch</th> <th>train_loss</th> <th>val_loss</th> </tr> </thead> <tbody> <tr><td>0.0</td><td>14.0</td><td>14.0</td></tr> <tr><td>2.5</td><td>13.5</td><td>13.5</td></tr> <tr><td>5.0</td><td>12.5</td><td>13.0</td></tr> <tr><td>7.5</td><td>10.5</td><td>12.0</td></tr> <tr><td>10.0</td><td>10.0</td><td>10.5</td></tr> <tr><td>12.5</td><td>7.5</td><td>10.5</td></tr> <tr><td>15.0</td><td>7.0</td><td>9.5</td></tr> <tr><td>17.5</td><td>6.5</td><td>11.5</td></tr> </tbody> </table>	epoch	train_loss	val_loss	0.0	14.0	14.0	2.5	13.5	13.5	5.0	12.5	13.0	7.5	10.5	12.0	10.0	10.0	10.5	12.5	7.5	10.5	15.0	7.0	9.5	17.5	6.5	11.5
	0	1	2	3																																																						
0	7	11	6	2																																																						
1	2	24	2	1																																																						
2	2	0	10	5																																																						
3	1	11	4	19																																																						
epoch	train_loss	val_loss																																																								
0.0	14.0	14.0																																																								
2.5	13.5	13.5																																																								
5.0	12.5	13.0																																																								
7.5	10.5	12.0																																																								
10.0	10.0	10.5																																																								
12.5	7.5	10.5																																																								
15.0	7.0	9.5																																																								
17.5	6.5	11.5																																																								
weighted avg	0.56	0.54	107																																																							
accuracy		0.56	107																																																							
macro avg	0.56	0.53	107																																																							
weighted avg	0.56	0.54	107																																																							

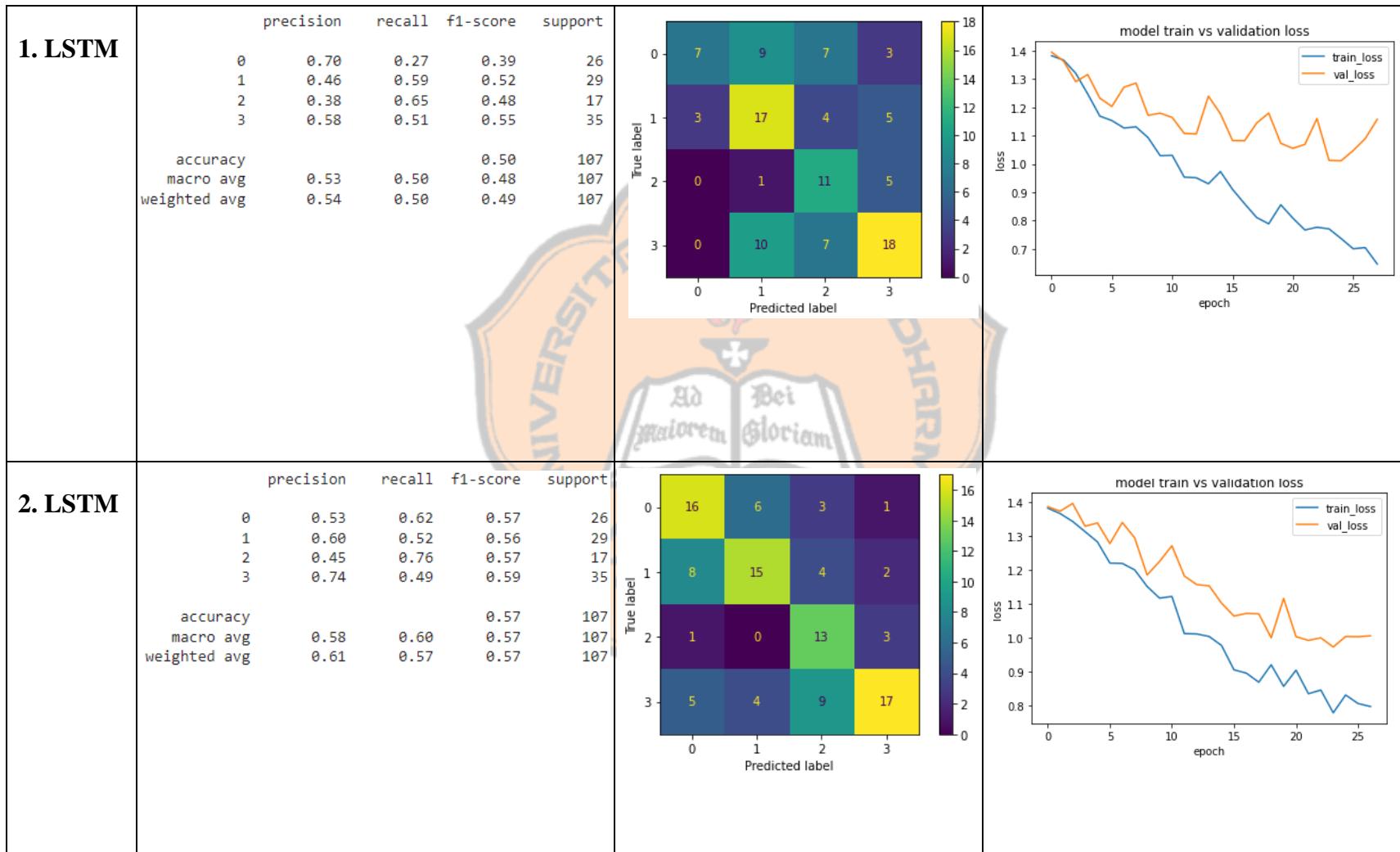
Tabel 4.8 Pengujian Arsitektur Kompleks 1 dengan 4 Kelas

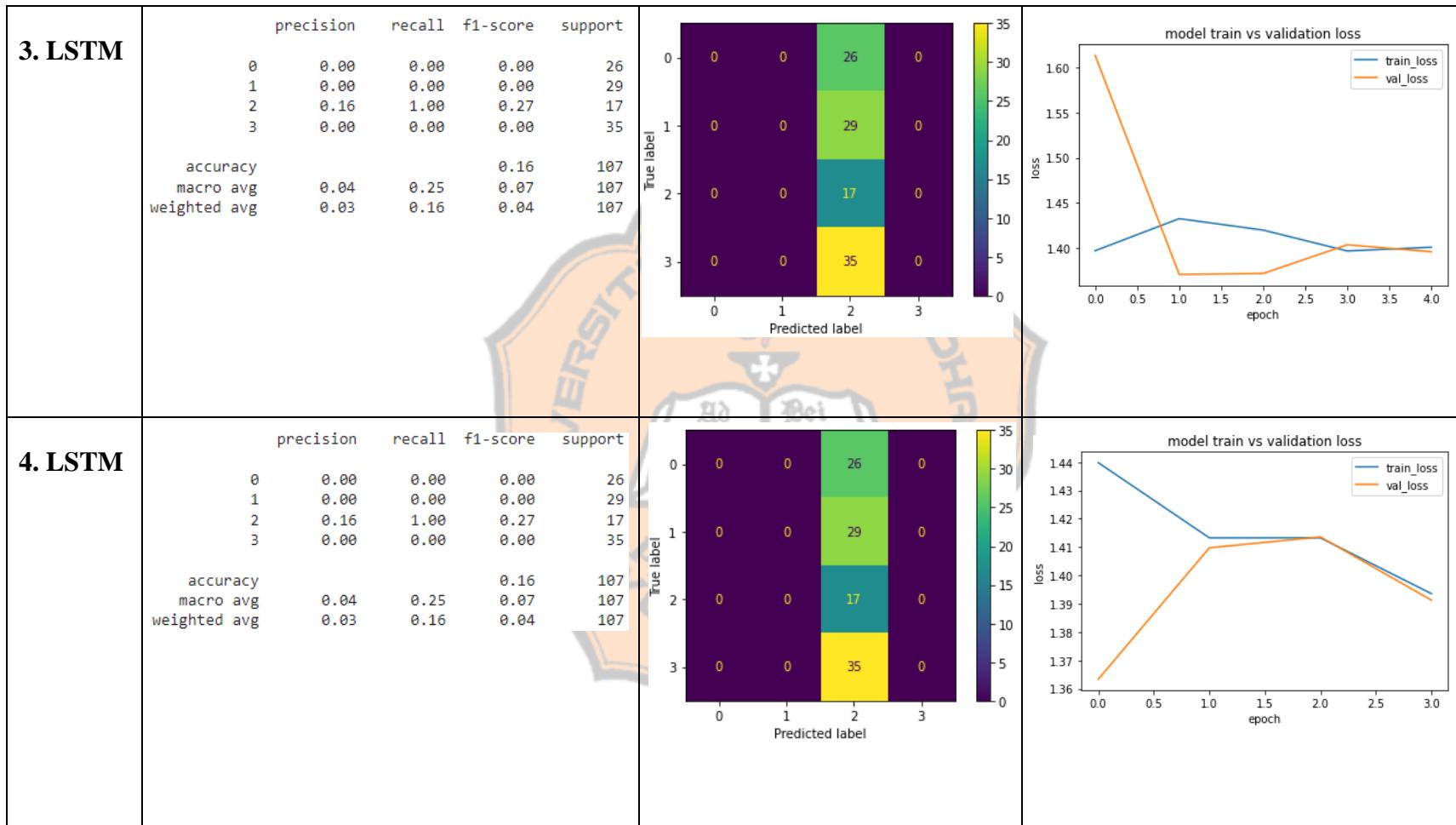


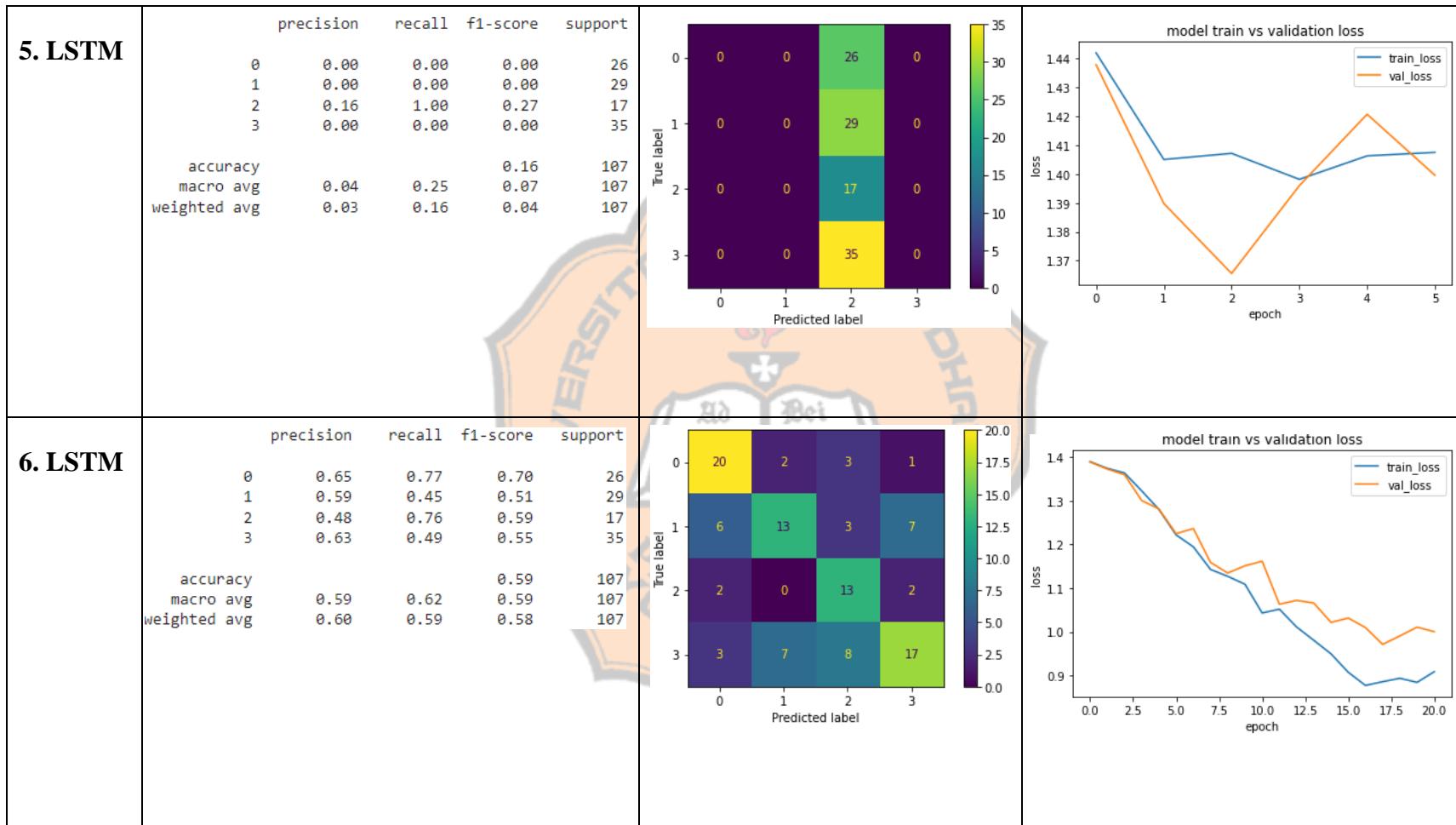


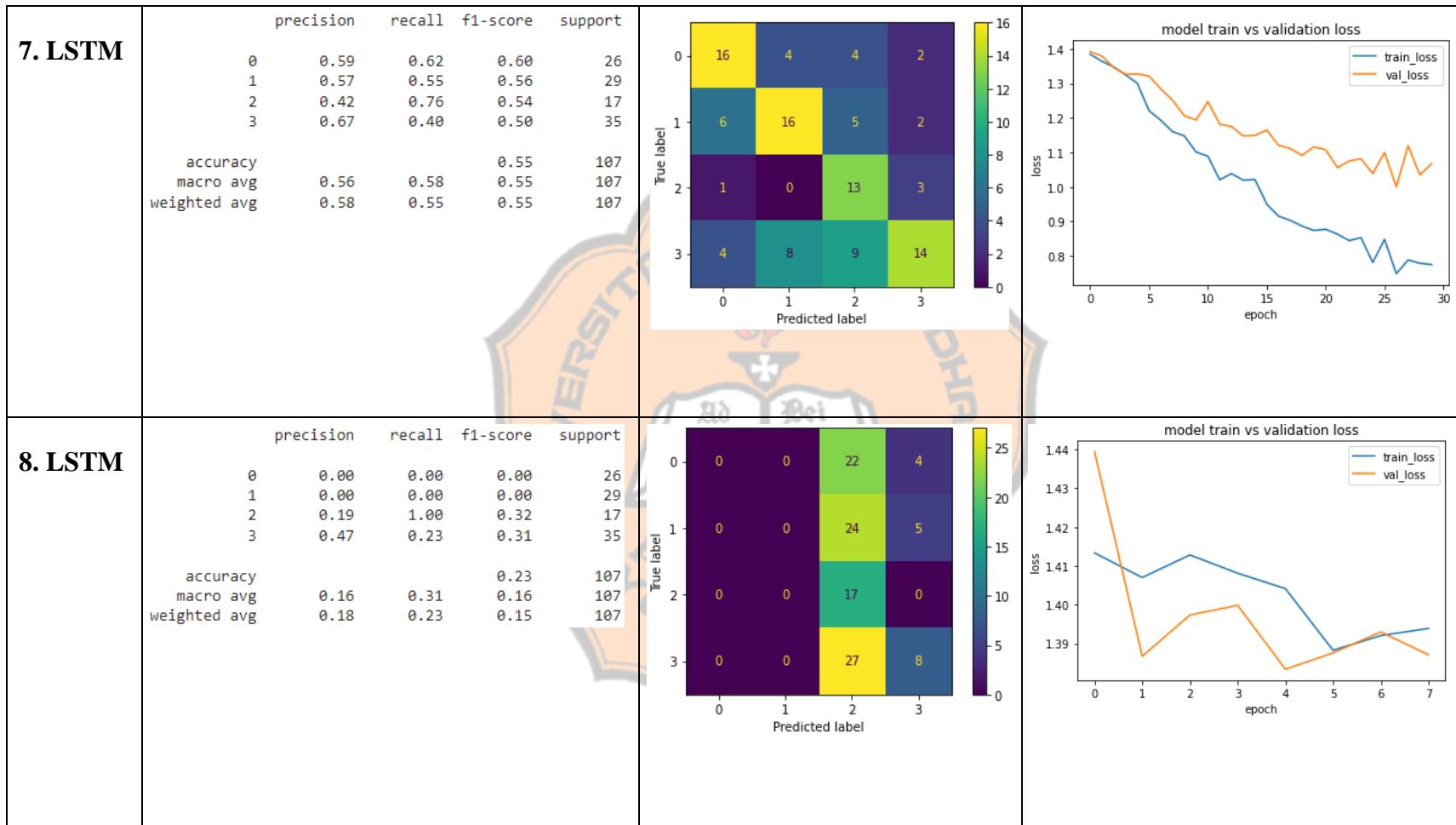
<b>6. LSTM</b>	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.56</td><td>0.69</td><td>0.62</td><td>26</td></tr> <tr><td>1</td><td>0.62</td><td>0.62</td><td>0.62</td><td>29</td></tr> <tr><td>2</td><td>0.58</td><td>0.65</td><td>0.61</td><td>17</td></tr> <tr><td>3</td><td>0.67</td><td>0.51</td><td>0.58</td><td>35</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.61</td><td>107</td></tr> <tr><td>macro avg</td><td>0.61</td><td>0.62</td><td>0.61</td><td>107</td></tr> <tr><td>weighted avg</td><td>0.61</td><td>0.61</td><td>0.61</td><td>107</td></tr> </tbody> </table>		precision	recall	f1-score	support	0	0.56	0.69	0.62	26	1	0.62	0.62	0.62	29	2	0.58	0.65	0.61	17	3	0.67	0.51	0.58	35	accuracy			0.61	107	macro avg	0.61	0.62	0.61	107	weighted avg	0.61	0.61	0.61	107	<table border="1"> <thead> <tr> <th>True label \ Predicted label</th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> </tr> </thead> <tbody> <tr><td>0</td><td>18</td><td>3</td><td>2</td><td>3</td></tr> <tr><td>1</td><td>6</td><td>18</td><td>2</td><td>3</td></tr> <tr><td>2</td><td>2</td><td>1</td><td>11</td><td>3</td></tr> <tr><td>3</td><td>6</td><td>7</td><td>4</td><td>18</td></tr> </tbody> </table>	True label \ Predicted label	0	1	2	3	0	18	3	2	3	1	6	18	2	3	2	2	1	11	3	3	6	7	4	18	<p>model train vs validation loss</p> <p>loss</p> <p>epoch</p>
	precision	recall	f1-score	support																																																																
0	0.56	0.69	0.62	26																																																																
1	0.62	0.62	0.62	29																																																																
2	0.58	0.65	0.61	17																																																																
3	0.67	0.51	0.58	35																																																																
accuracy			0.61	107																																																																
macro avg	0.61	0.62	0.61	107																																																																
weighted avg	0.61	0.61	0.61	107																																																																
True label \ Predicted label	0	1	2	3																																																																
0	18	3	2	3																																																																
1	6	18	2	3																																																																
2	2	1	11	3																																																																
3	6	7	4	18																																																																
<b>7. LSTM RMSPro p</b>	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.80</td><td>0.46</td><td>0.59</td><td>26</td></tr> <tr><td>1</td><td>0.47</td><td>0.90</td><td>0.62</td><td>29</td></tr> <tr><td>2</td><td>0.52</td><td>0.82</td><td>0.64</td><td>17</td></tr> <tr><td>3</td><td>0.90</td><td>0.26</td><td>0.40</td><td>35</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.57</td><td>107</td></tr> <tr><td>macro avg</td><td>0.67</td><td>0.61</td><td>0.56</td><td>107</td></tr> <tr><td>weighted avg</td><td>0.70</td><td>0.57</td><td>0.54</td><td>107</td></tr> </tbody> </table>		precision	recall	f1-score	support	0	0.80	0.46	0.59	26	1	0.47	0.90	0.62	29	2	0.52	0.82	0.64	17	3	0.90	0.26	0.40	35	accuracy			0.57	107	macro avg	0.67	0.61	0.56	107	weighted avg	0.70	0.57	0.54	107	<table border="1"> <thead> <tr> <th>True label \ Predicted label</th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> </tr> </thead> <tbody> <tr><td>0</td><td>12</td><td>10</td><td>3</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>26</td><td>3</td><td>0</td></tr> <tr><td>2</td><td>1</td><td>2</td><td>14</td><td>0</td></tr> <tr><td>3</td><td>2</td><td>17</td><td>7</td><td>9</td></tr> </tbody> </table>	True label \ Predicted label	0	1	2	3	0	12	10	3	1	1	0	26	3	0	2	1	2	14	0	3	2	17	7	9	<p>model train vs validation loss</p> <p>loss</p> <p>epoch</p>
	precision	recall	f1-score	support																																																																
0	0.80	0.46	0.59	26																																																																
1	0.47	0.90	0.62	29																																																																
2	0.52	0.82	0.64	17																																																																
3	0.90	0.26	0.40	35																																																																
accuracy			0.57	107																																																																
macro avg	0.67	0.61	0.56	107																																																																
weighted avg	0.70	0.57	0.54	107																																																																
True label \ Predicted label	0	1	2	3																																																																
0	12	10	3	1																																																																
1	0	26	3	0																																																																
2	1	2	14	0																																																																
3	2	17	7	9																																																																

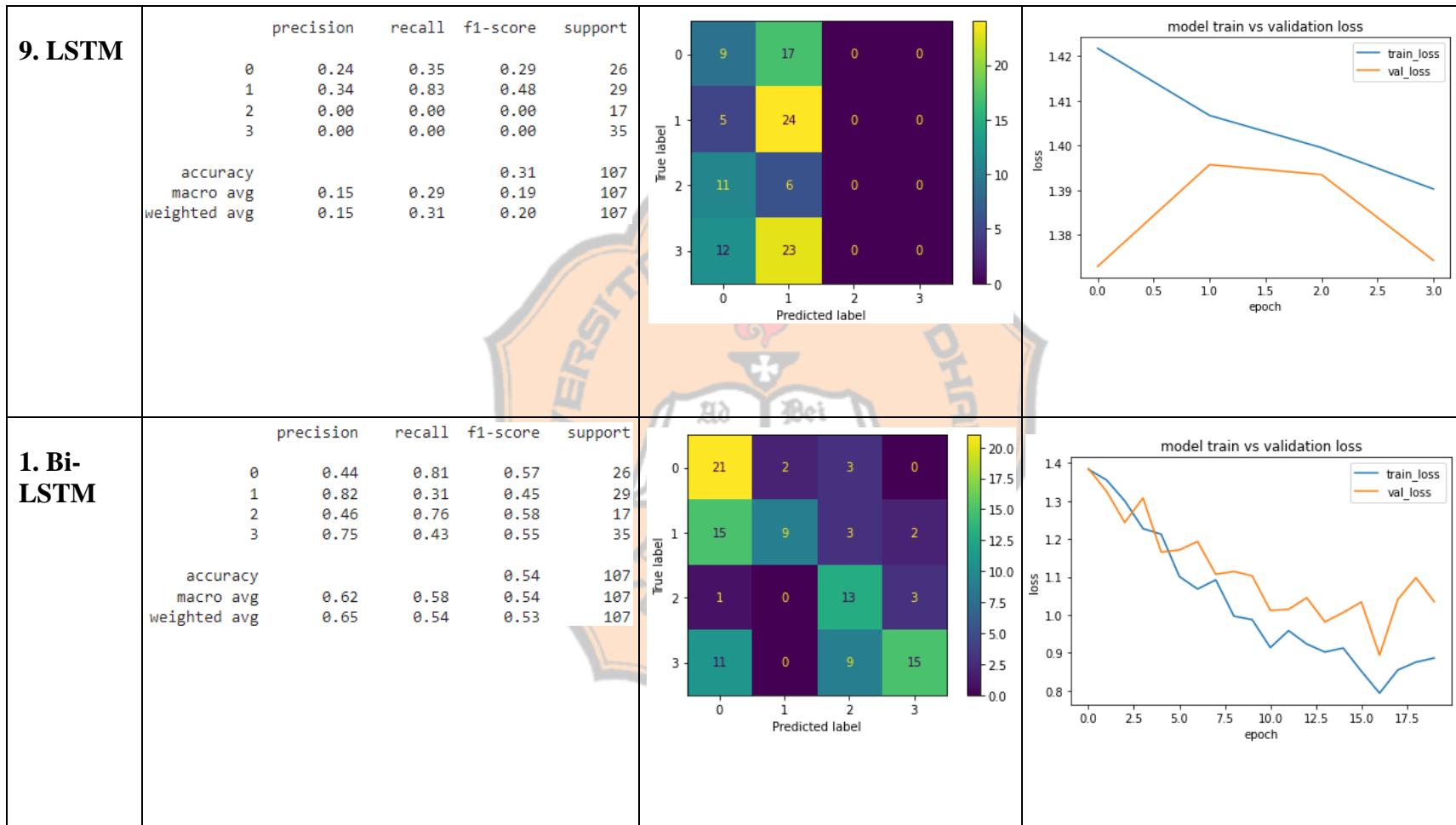
Tabel 4.9 Pengujian Arsitektur Kompleks LSTM 2 dengan 4 Kelas

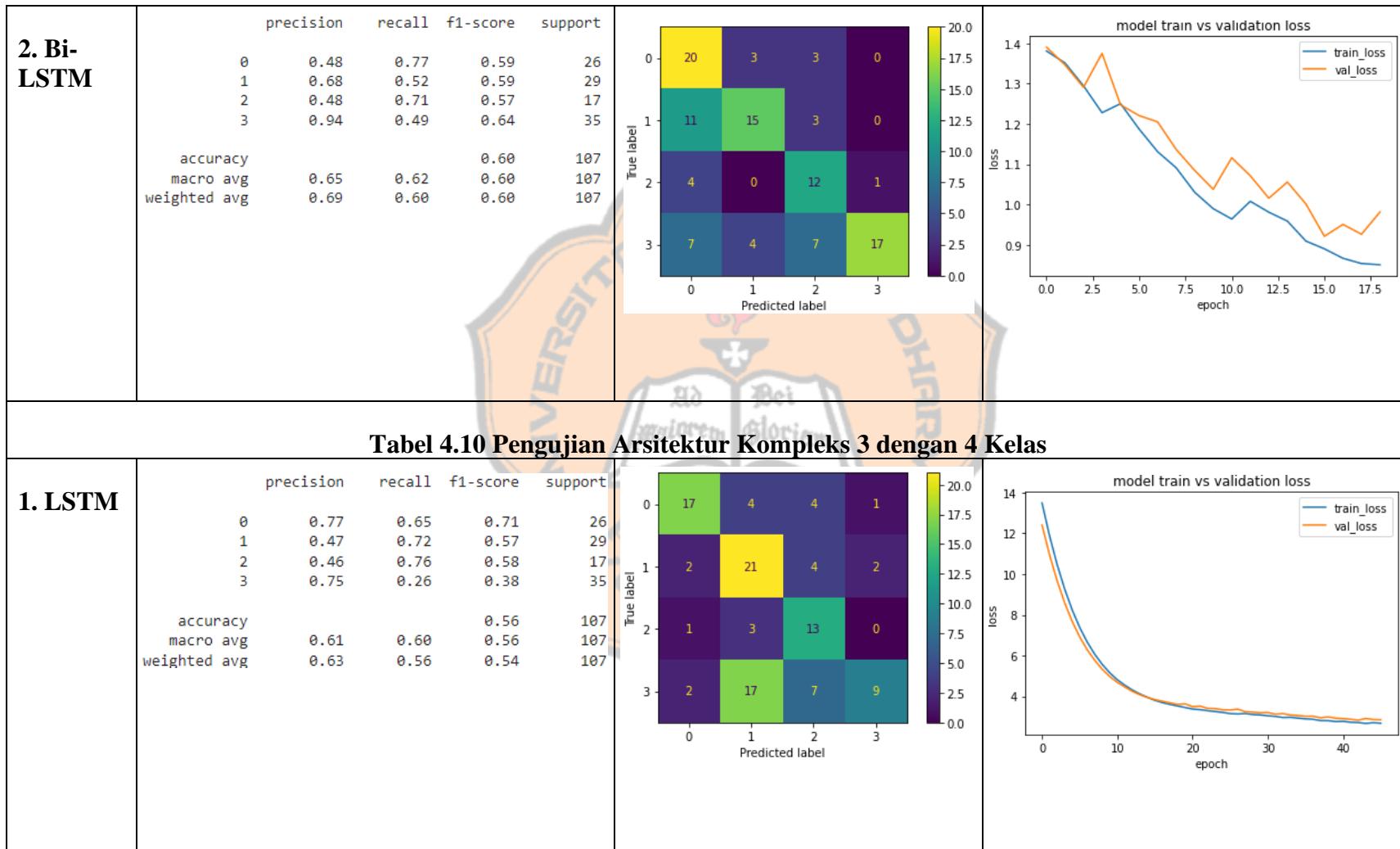




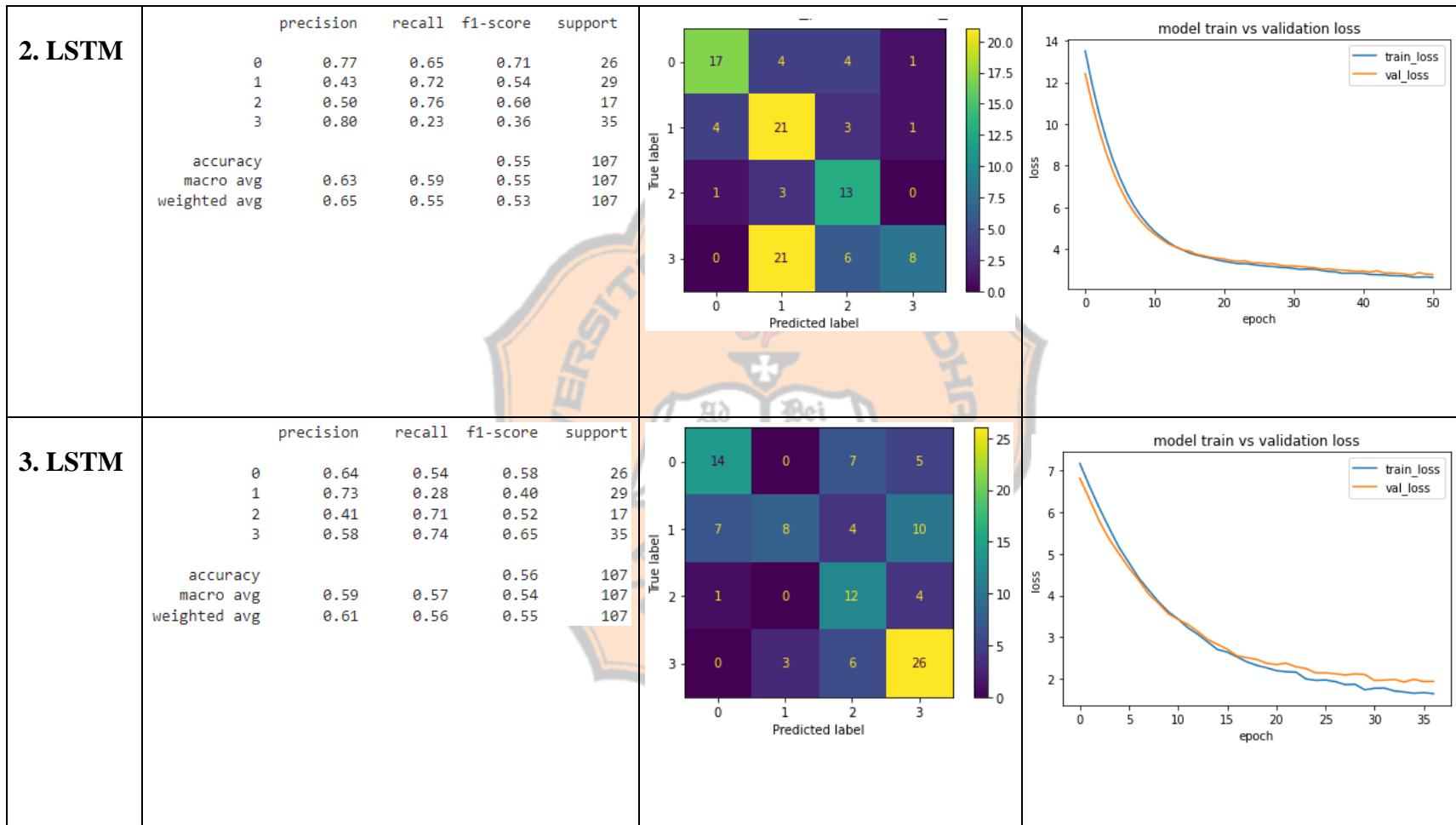


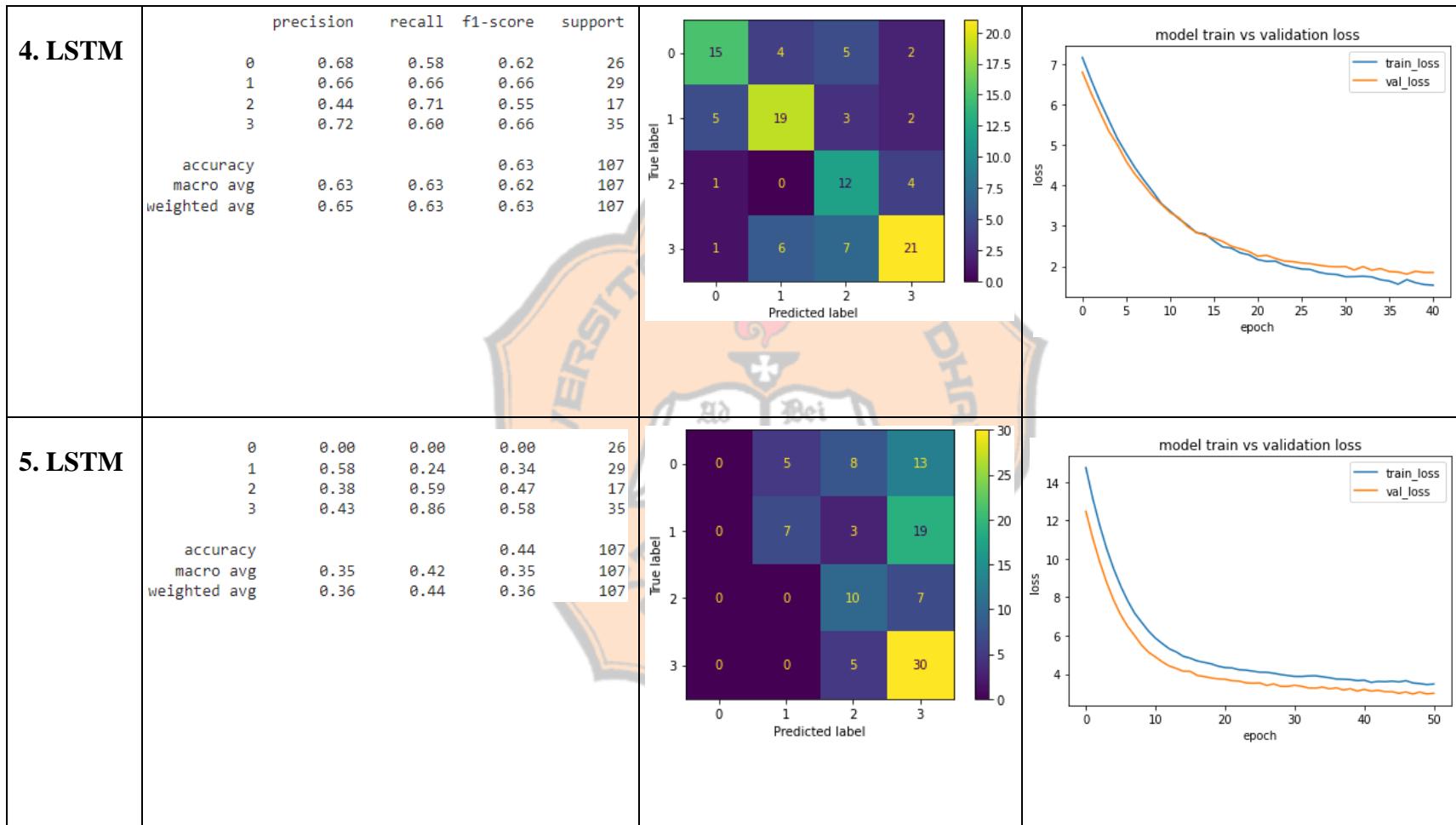


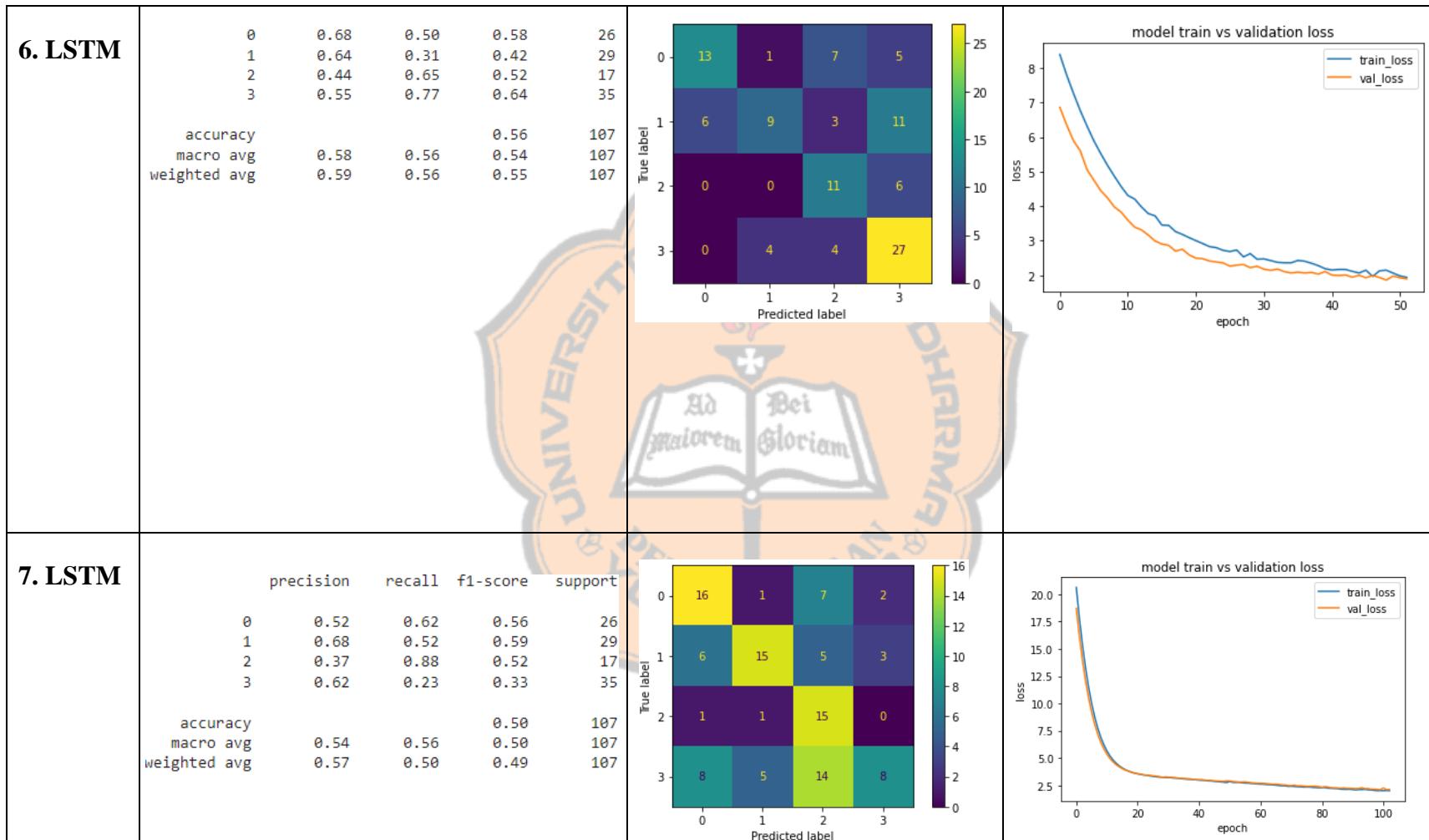


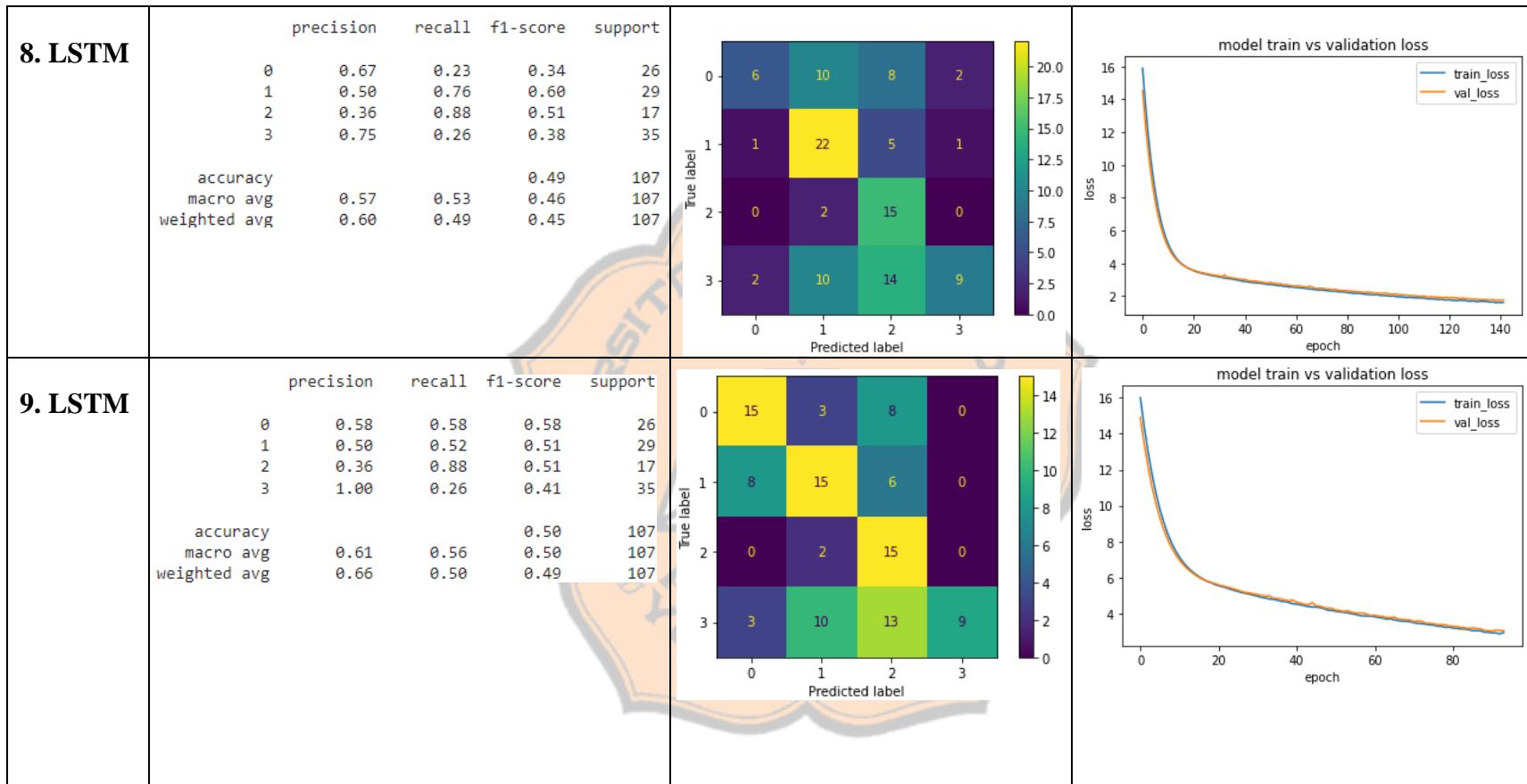


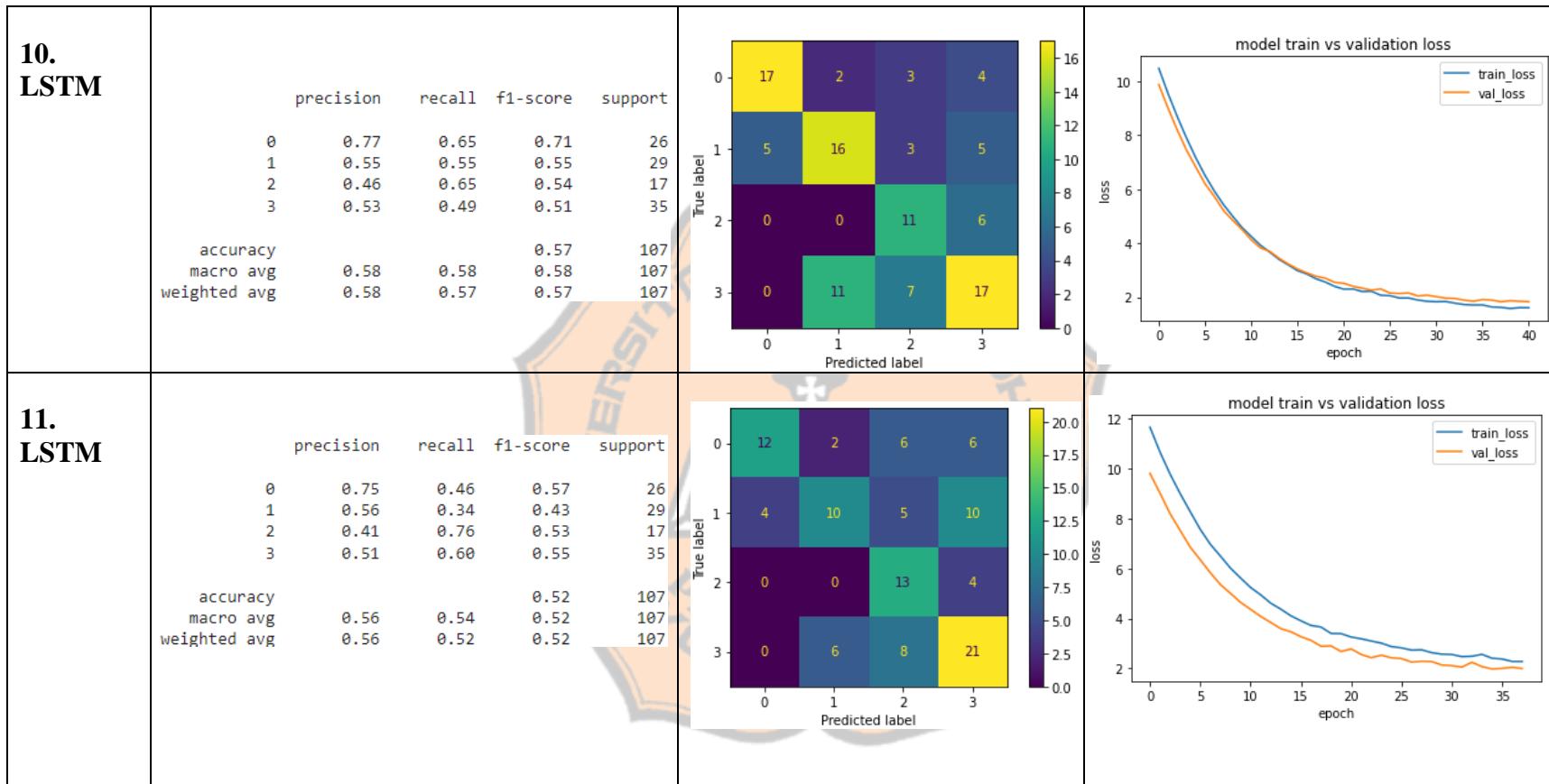
Tabel 4.10 Pengujian Arsitektur Kompleks 3 dengan 4 Kelas

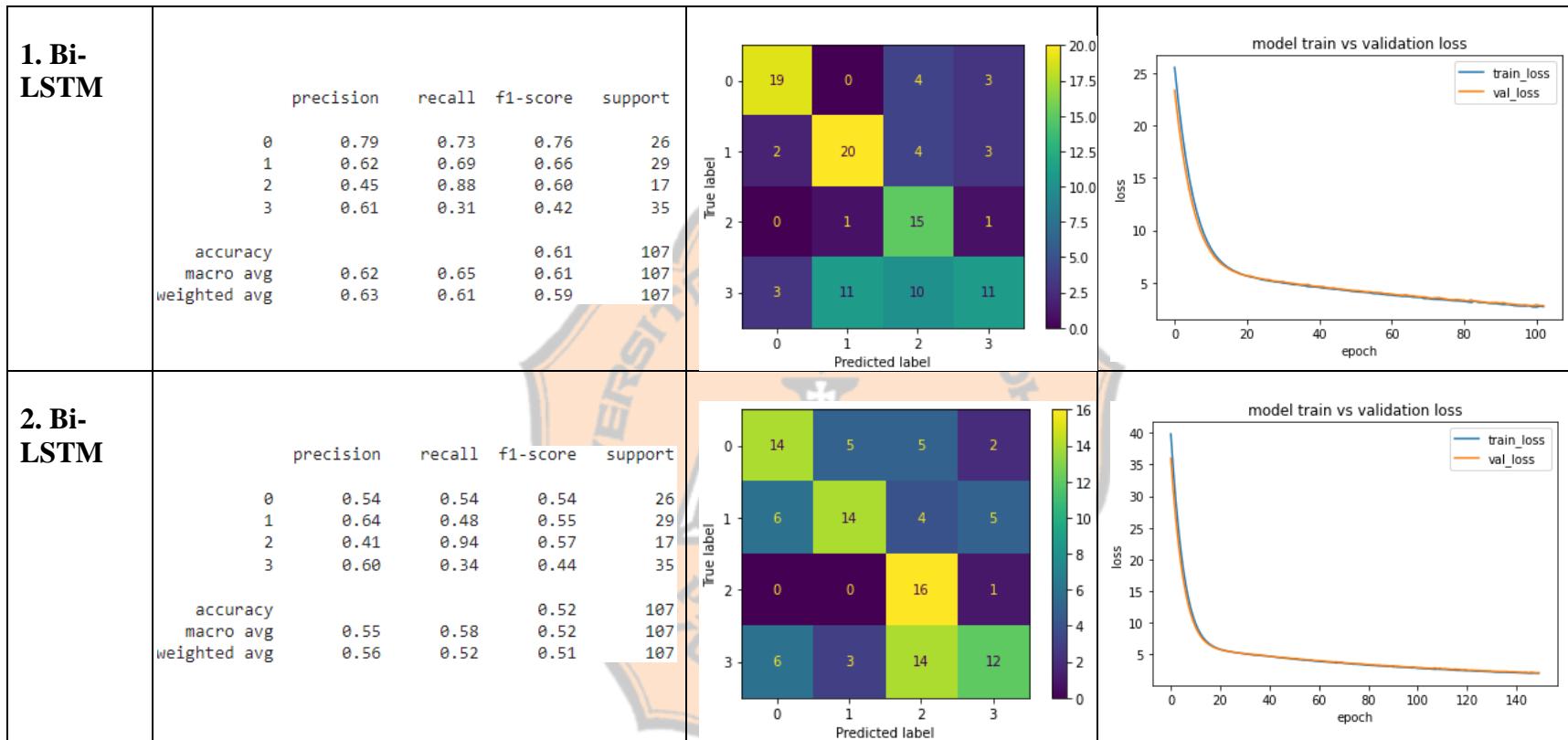












<b>3. Bi-LSTM</b>	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.92</td><td>0.46</td><td>0.62</td><td>26</td></tr> <tr><td>1</td><td>0.65</td><td>0.38</td><td>0.48</td><td>29</td></tr> <tr><td>2</td><td>0.45</td><td>0.59</td><td>0.51</td><td>17</td></tr> <tr><td>3</td><td>0.51</td><td>0.80</td><td>0.62</td><td>35</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.57</td><td>107</td></tr> <tr><td>macro avg</td><td>0.63</td><td>0.56</td><td>0.56</td><td>107</td></tr> <tr><td>weighted avg</td><td>0.64</td><td>0.57</td><td>0.56</td><td>107</td></tr> </tbody> </table>		precision	recall	f1-score	support	0	0.92	0.46	0.62	26	1	0.65	0.38	0.48	29	2	0.45	0.59	0.51	17	3	0.51	0.80	0.62	35	accuracy			0.57	107	macro avg	0.63	0.56	0.56	107	weighted avg	0.64	0.57	0.56	107	<table border="1"> <thead> <tr> <th colspan="2"></th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> </tr> <tr> <th rowspan="2">True label</th> <th>Predicted label</th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>12</td> <td>4</td> <td>4</td> <td>6</td> </tr> <tr> <td>1</td> <td>1</td> <td>11</td> <td>3</td> <td>14</td> </tr> <tr> <td>2</td> <td>0</td> <td>0</td> <td>10</td> <td>7</td> </tr> <tr> <td>3</td> <td>0</td> <td>2</td> <td>5</td> <td>28</td> </tr> </tbody> </table>			0	1	2	3	True label	Predicted label	0	1	2	3	0	12	4	4	6	1	1	11	3	14	2	0	0	10	7	3	0	2	5	28	<p>model train vs validation loss</p> <p>train_loss (blue line)</p> <p>val_loss (orange line)</p> <table border="1"> <thead> <tr> <th>epoch</th> <th>train_loss</th> <th>val_loss</th> </tr> </thead> <tbody> <tr><td>0</td><td>17.5</td><td>15.0</td></tr> <tr><td>10</td><td>5.5</td><td>4.5</td></tr> <tr><td>20</td><td>4.5</td><td>3.5</td></tr> <tr><td>30</td><td>3.5</td><td>3.0</td></tr> <tr><td>40</td><td>3.0</td><td>2.5</td></tr> <tr><td>50</td><td>2.8</td><td>2.5</td></tr> </tbody> </table>	epoch	train_loss	val_loss	0	17.5	15.0	10	5.5	4.5	20	4.5	3.5	30	3.5	3.0	40	3.0	2.5	50	2.8	2.5
	precision	recall	f1-score	support																																																																																												
0	0.92	0.46	0.62	26																																																																																												
1	0.65	0.38	0.48	29																																																																																												
2	0.45	0.59	0.51	17																																																																																												
3	0.51	0.80	0.62	35																																																																																												
accuracy			0.57	107																																																																																												
macro avg	0.63	0.56	0.56	107																																																																																												
weighted avg	0.64	0.57	0.56	107																																																																																												
		0	1	2	3																																																																																											
True label	Predicted label	0	1	2	3																																																																																											
	0	12	4	4	6																																																																																											
1	1	11	3	14																																																																																												
2	0	0	10	7																																																																																												
3	0	2	5	28																																																																																												
epoch	train_loss	val_loss																																																																																														
0	17.5	15.0																																																																																														
10	5.5	4.5																																																																																														
20	4.5	3.5																																																																																														
30	3.5	3.0																																																																																														
40	3.0	2.5																																																																																														
50	2.8	2.5																																																																																														
<b>4. Bi-LSTM</b>	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.87</td><td>0.50</td><td>0.63</td><td>26</td></tr> <tr><td>1</td><td>0.57</td><td>0.69</td><td>0.62</td><td>29</td></tr> <tr><td>2</td><td>0.45</td><td>0.76</td><td>0.57</td><td>17</td></tr> <tr><td>3</td><td>0.64</td><td>0.51</td><td>0.57</td><td>35</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.60</td><td>107</td></tr> <tr><td>macro avg</td><td>0.63</td><td>0.62</td><td>0.60</td><td>107</td></tr> <tr><td>weighted avg</td><td>0.65</td><td>0.60</td><td>0.60</td><td>107</td></tr> </tbody> </table>		precision	recall	f1-score	support	0	0.87	0.50	0.63	26	1	0.57	0.69	0.62	29	2	0.45	0.76	0.57	17	3	0.64	0.51	0.57	35	accuracy			0.60	107	macro avg	0.63	0.62	0.60	107	weighted avg	0.65	0.60	0.60	107	<table border="1"> <thead> <tr> <th colspan="2"></th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> </tr> <tr> <th rowspan="2">True label</th> <th>Predicted label</th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>13</td> <td>6</td> <td>5</td> <td>2</td> </tr> <tr> <td>1</td> <td>2</td> <td>20</td> <td>3</td> <td>4</td> </tr> <tr> <td>2</td> <td>0</td> <td>0</td> <td>13</td> <td>4</td> </tr> <tr> <td>3</td> <td>0</td> <td>9</td> <td>8</td> <td>18</td> </tr> </tbody> </table>			0	1	2	3	True label	Predicted label	0	1	2	3	0	13	6	5	2	1	2	20	3	4	2	0	0	13	4	3	0	9	8	18	<p>model train vs validation loss</p> <p>train_loss (blue line)</p> <p>val_loss (orange line)</p> <table border="1"> <thead> <tr> <th>epoch</th> <th>train_loss</th> <th>val_loss</th> </tr> </thead> <tbody> <tr><td>0</td><td>25.5</td><td>24.0</td></tr> <tr><td>10</td><td>8.5</td><td>7.5</td></tr> <tr><td>20</td><td>4.5</td><td>3.5</td></tr> <tr><td>30</td><td>3.5</td><td>3.0</td></tr> <tr><td>40</td><td>3.0</td><td>2.5</td></tr> </tbody> </table>	epoch	train_loss	val_loss	0	25.5	24.0	10	8.5	7.5	20	4.5	3.5	30	3.5	3.0	40	3.0	2.5			
	precision	recall	f1-score	support																																																																																												
0	0.87	0.50	0.63	26																																																																																												
1	0.57	0.69	0.62	29																																																																																												
2	0.45	0.76	0.57	17																																																																																												
3	0.64	0.51	0.57	35																																																																																												
accuracy			0.60	107																																																																																												
macro avg	0.63	0.62	0.60	107																																																																																												
weighted avg	0.65	0.60	0.60	107																																																																																												
		0	1	2	3																																																																																											
True label	Predicted label	0	1	2	3																																																																																											
	0	13	6	5	2																																																																																											
1	2	20	3	4																																																																																												
2	0	0	13	4																																																																																												
3	0	9	8	18																																																																																												
epoch	train_loss	val_loss																																																																																														
0	25.5	24.0																																																																																														
10	8.5	7.5																																																																																														
20	4.5	3.5																																																																																														
30	3.5	3.0																																																																																														
40	3.0	2.5																																																																																														
<b>4.2.2.3 Pengujian dengan Menggunakan Sastrawi sebagai Stemmer</b>																																																																																																

