IMPLEMENTASI MODEL *LONG SHORT-TERM MEMORY* (LSTM) UNTUK KLASIFIKASI *MULTI-LABEL* UJARAN KEBENCIAN PADA *TWEET* BAHASA INDONESIA

SKRIPSI

Oleh :
FIRNA
NIM. 220605110017

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

IMPLEMENTASI MODEL *LONG SHORT-TERM MEMORY* (LSTM) UNTUK KLASIFIKASI *MULTI-LABEL* UJARAN KEBENCIAN PADA *TWEET* BAHASA INDONESIA

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
FIRNA
NIM. 220605110017

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2025

HALAMAN PERSETUJUAN

IMPLEMENTASI MODEL *LONG SHORT-TERM MEMORY* (LSTM) UNTUK KLASIFIKASI *MULTI-LABEL* UJARAN KEBENCIAN PADA TWEET BAHASA INDONESIA

SKRIPSI

Oleh :
FIRNA
NIM. 220605110017

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 8 Desember 2025

Pembimbing I,	Pembimbing II,
Okta Qomaruddin Aziz, M.Kom	Dr. Irwan Budi Santoso, M.Kom
NIP. 19911019 201903 1 013	NIP. 19770103 201101 1 004

Mengetahui
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

HALAMAN PENGESAHAN

IMPLEMENTASI MODEL *LONG SHORT-TERM MEMORY* (LSTM) UNTUK KLASIFIKASI *MULTI-LABEL* UJARAN KEBENCIAN PADA TWEET BAHASA INDONESIA

SKRIPSI

Oleh :
FIRNA
NIM. 220605110017

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 17 Desember 2025

Susunan Dewan Penguji

Ketua Penguji	: Prof. Dr. Suhartono S.Si M.Kom NIP. 19680519 200312 1 001	( )
Anggota Penguji I	: Nur Fitriyah Ayu Tunjung Sari, M.Cs NIP. 19911226 202012 2 001	( )
Anggota Penguji II	: Okta Qomaruddin Aziz, M.Kom NIP. 19911019 201903 1 013	( )
Anggota Penguji III	: Dr. Irwan Budi Santoso, M.Kom NIP. 19770103 201101 1 004	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Firna
NIM	: 220605110017
Fakultas / Program Studi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Implementasi Model *Long Short-Term Memory* (LSTM) untuk Klasifikasi *Multi-Label* Ujaran Kebencian pada *Tweet* Bahasa Indonesia

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 17 Desember 2025
Yang membuat pernyataan,

Firna
NIM.220605110017

MOTTO

“teruslah hidup, barangkali esok ada cinta yang membuatmu tak ingin mati.”

HALAMAN PERSEMBAHAN

Dengan menyebut nama Allah Swt., Yang Maha Menyempurnakan setiap proses, skripsi ini akhirnya menemukan bentuknya.

Proses penyusunannya tidak lahir dari ruang kosong, melainkan melalui perjalanan yang dipenuhi doa, pengorbanan, dan keteguhan.

Skripsi ini penulis persembahkan kepada:

Mama dan Papa tersayang, Farida dan Manaf
Yang melalui keyakinan, doa yang tak pernah jeda, serta keteguhan dalam menukar lelahnya demi satu tujuan yang hari ini telah sampai.

Adik-adik tercinta, Adik Dafis, Adik Ozil, dan Adik Zafir
Yang dengan tawa dan doa senantiasa mengingatkan bahwa perjuangan ini memiliki makna yang melampaui sekadar gelar.

Seluruh Keluarga Besar
Yang selalu memberikan dukungan, kehangatan, dan doa dalam setiap proses yang dijalani.

Para Dosen
Yang menjadi penunjuk arah melalui ilmu, bimbingan, dan kesabaran di ruang yang menumbuhkan nalar dan disiplin.

Teman-teman Angkatan 22 Teknik Informatika *(Infinity)*
Yang melalui kebersamaan dan perjalanan bersama menjadikan langkah panjang terasa lebih ringan.

Skripsi ini bukan sekadar rangkaian kata dan data, melainkan penanda bahwa doa yang dijaga, pengorbanan yang ikhlas, dan kesungguhan yang dijalani dengan sabar akan menemukan maknanya.

KATA PENGANTAR

Assalamualaikum wr wb.

Alhamdulillah segala puji dan Syukur senantiasa penulis panjatkan pada Allah subhanahu wa ta'ala atas berkat Rahmat, serta hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Implementasi Model *Long Short-Term Memory* (LSTM) untu Klasifikasi *Multi-Label* Ujaran Kebencian pada *Tweet* Bahasa Indonesia". Sholawat serta salam tetap tercurahkan kepada Nabi Muhammad SAW. Dan semoga kita semua mendapat syafaatnya di hari akhir kelak, Aamiin.

Dalam penulisan skripsi ini, penulis menyadari banyak pihak yang terlibat baik dalam proses membimbing penulisan dan juga memberikan semangat dan dukungan moril atau materiil. Untuk itu, penulis ingin menyampaikan terima kasih kepada:

1. Dr. Hj. Ilfi Nur Diana, M.Si., CAHRM., CRMP., selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Agus Mulyono, M.Kes., selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Supriyono, M.Kom., selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Okta Qomaruddin Aziz, M.Kom., selaku Dosen Pembimbing I, yang sejak awal pengajuan *pra proposal* hingga skripsi ini selesai selalu membersamai proses penulis dengan penuh kesabaran. Di setiap arahan dan masukan yang

Bapak berikan, penulis selalu menemukan ketenangan, sehingga setiap kebingungan dan kepanikan perlahan bisa dihadapi. Cara Bapak membimbing yang tenang dan menenangkan membuat penulis merasa lebih percaya diri untuk terus melangkah dan menyelesaikan satu per satu proses yang ada. Terima kasih yang sebesar-besarnya atas waktu, perhatian, dan dedikasi Bapak. Kehadiran dan bimbingan Bapak menjadi bagian penting yang menguatkan penulis hingga skripsi ini dapat terselesaikan.

5. Dr. Irwan Budi Santoso, M. Kom., selaku Dosen Pembimbing II, terima kasih banyak ya Pak atas segala saran dan koreksinya. Penulis sangat terbantu dengan ketelitian Bapak dalam memeriksa bagian-bagian yang detail, sehingga skripsi ini jadi lebih rapi dan jelas arahnya.
6. Prof. Dr. Suhartono, S.Si., M.Kom., selaku Ketua Penguji, terima kasih sudah bersedia menelaah skripsi ini. Setiap masukan dan sudut pandang yang Bapak berikan sangat membantu penulis untuk belajar lebih dalam lagi dan membuat hasil penelitian ini jadi lebih lengkap.
7. Nur Fitriyah Ayu Tunjung Sari, M.Cs., selaku Anggota Penguji I, terima kasih banyak atas waktu dan masukannya. Kritik serta saran dari Ibu sangat membantu penulis dalam memperbaiki kekurangan-kekurangan yang ada, sehingga skripsi ini bisa selesai dengan lebih baik.
8. Nia Faricha, S.Si., selaku Admin Program Studi Teknik Informatika, terima kasih banyak sudah sangat sabar membantu segala urusan administrasi penulis. Terima kasih sudah selalu sigap memberikan informasi dan arahan

yang sangat memudahkan penulis sejak awal kuliah sampai akhirnya bisa menyelesaikan skripsi ini.

9. Seluruh Dosen, Admin, Laboran dan jajaran Staf Program Studi Teknik Informatika yang telah memberikan banyak bantuan selama studi ini.
10. Papa tersayang, Manaf. Sosok cinta pertama yang kehadirannya selalu penulis butuhkan. Terima kasih atas doa, kerja keras, dan keringat yang Papa curahkan tanpa keluh. Maafkan jika penulis masih jauh dari sempurna. Meski Papa sering menyimpan rasa sayang dalam diam dan gengsi, penulis selalu merasakan kasih sayang itu hadir menguatkan. Terima kasih telah membentuk penulis menjadi perempuan yang berani menghadapi dunia, percaya diri berinteraksi dengan siapa pun, dan mampu menjaga diri dengan baik. Nasihat Papa untuk *"tidak perlu takut kepada siapa pun selama kita baik dan sopan"* telah menjadi kekuatan utama bagi penulis untuk menyelesaikan skripsi ini. Semoga Allah SWT membalas segala pengorbanan Papa dengan kesehatan, umur panjang, dan pahala juga rezeki yang melimpah.
11. Mama tersayang, Farida. Mama adalah sosok yang mungkin tidak banyak mencampuri urusan pendidikan penulis karena rasa tidak tega melihat penulis terlalu lelah. Di balik itu, penulis selalu merasakan kasih sayang Mama yang menenangkan. Terima kasih telah menjadi sosok yang aktif dan ceria, sifat Mama itulah yang kini tumbuh dalam diri penulis. Jika hari ini penulis mampu bertahan dan menguatkan orang lain, itu karena Mama telah lebih dulu mencontohkannya. Terima kasih telah menjadi ruang cerita yang

paling nyaman sebagai sesama perempuan di rumah, ruang yang terasa semakin bermakna selama masa perkuliahan ini. Doa tulus Mama adalah pelindung yang mengantarkan penulis hingga ke titik ini. Semoga Mama selalu diberikan kesehatan dan umur yang panjang. Terima kasih telah menjadi Mama terbaik bagi penulis.

12. Adik-adik Tersayang, Adik Muhamad Dafis, Adik Muhammad Ozil, dan Adik Abdul Zafir. Kalian adalah laki-laki setelah Papa yang sangat penulis sayang. Penulis menyadari belum sepenuhnya mampu menjadi kakak yang terbaik. Terima kasih atas kebersamaan kalian yang telah menjadi kekuatan luar biasa bagi penulis dalam menyelesaikan skripsi ini. Kalian adalah harapan agar kelak penulis dapat mengajak kalian semua untuk tumbuh dan mewujudkan mimpi-mimpi besar bersama di masa depan.
13. Keluarga Besar Penulis. Terima kasih kepada keluarga besar yang tidak dapat penulis sebutkan satu per satu dari pihak Alm. Bapak Uni dan Almh. Mama Uni (Kakek dan Nenek), serta Bapak Raja dan Mama Raja (Kakek dan Nenek). Terima kasih atas segala bentuk dukungan dan doa tulus yang selama ini diberikan kepada penulis. Dukungan tersebut telah menjadi kekuatan besar bagi penulis untuk dapat menyelesaikan skripsi ini tepat pada waktunya.
14. Sahabat Tersayang, Fidelia Patricia Engelita Pice. Pemilik nama cantik dengan panggilan kesayangan "inden (Ikan Bandeng)”. Sahabat yang menemani penulis sejak SMA hingga berkuliah bersama di Malang. Penulis merasa sangat beruntung memiliki sahabat yang sudah seperti saudara

perempuan sendiri. Terima kasih atas kehadiran dan *effort* luar biasa yang selalu Inden berikan tanpa penulis sangka, terutama dalam menjadi pendukung utama hingga skripsi ini dapat terselesaikan. Maaf jika penulis masih memiliki banyak kekurangan sebagai seorang sahabat. Panjang umur selalu ya, Nden. Hidup lebih lama.

15. Sahabat Tersayang, Dita Ramadani Daun. Pemilik nama cantik dengan panggilan kesayangan "daun happy mom". Sahabat yang perannya sama besarnya dengan Inden. Terima kasih karena meski terpisah jarak Manado–Malang, kasih sayang dan hadiah-hadiah kecil Dita selalu sampai kepada penulis. *Effort* paling luar biasa yang penulis tidak akan lupakan adalah saat Dita nekat menyusul penulis dan Inden ke Malang demi bisa bersama kembali. Maaf jika penulis masih kurang dalam menjadi sahabat. Dari Dita dan Indenlah, untuk pertama kalinya penulis bisa merasakan bagaimana rasanya dirayakan dan merayakan, kalian adalah rumah bagi penulis. Panjang umur selalu ya, Ditt. Hidup lebih lama.
16. Sahabat Sehati, Aisyah Nur Fitriyah. Pemilik nama cantik dengan panggilan kesayangan “aisee kinann”. Sosok yang selalu menjadi bagian pertama yang tahu apa yang penulis sedang rasakan. Terima kasih banyak atas kedewasaan dan pemikiranmu yang selalu selaras dengan penulis. Terima kasih sudah selalu sedia menjadi sosok yang penulis sibukkan dan tetap setia menemani sejak awal kuliah hingga berada di tahap penulisan kata pengantar ini. Maaf jika penulis masih memiliki banyak kekurangan dalam

persabatan ini. Terima kasih telah menemani proses panjang perkuliahan penulis hingga sampai di titik ini. Panjang umur selalu, Aise.

17. Sahabat Sehati, Moh Musa Al Kadzim. Penulis tidak menyangka perkenalan yang diawali dengan kesan penulis ke Musa adalah orang yang sombong saat mahasiswa baru, tapi ternyata menjadi bagian penting dalam hidup penulis. Terima kasih atas segala hadiah dan tingkah *toddler* yang sering kali mengganggu, namun selalu berhasil membuat penulis tersenyum bahagia. Terima kasih juga telah bersedia menemani dan menjadi penguat dalam proses perkuliahan yang panjang ini. Panjang umur selalu, Musa.
18. Sahabat Sehati, Dita Suci Yofana. Pemilik nama cantik dengan panggilan kesayangan “ditutt”. Nama depan yang sama dengan Dita Ramadani Daun, bertemu dengannya seperti menemukan Dita versi baru di perkuliahan. Berawal dari matkul Basis Data, siapa sangka sosok yang sempat penulis anggap "centil" ini ternyata sangat sehati, bahkan dalam urusan *life after break up*. Terima kasih sudah menemani di masa-masa awal sempro yang sempat membuat penulis hampir setengah gila hingga skripsi ini selesai. Terima kasih juga sudah selalu dewasa menghadapi sikap penulis. Panjang umur selalu, Ditutt.
19. Sahabat Sehati, Qonita Nashifa Anabila. Pemilik nama cantik dengan panggilan kesayangan “beltutt”. Sosok paling tomboy di *circle* yang keras kepala, tapi aslinya punya hati yang lembut dan baik tak terkira, Sifatnya unik dan beda banget dari kebanyakan orang. Terima kasih sudah jadi

sahabat sehati, mulai dari menjalani PKL bersama hingga akhirnya kita bisa menyelesaikan skripsi ini bersama-sama. Panjang umur selalu, Beltutt.

20. Sahabat Sehati, Radifan Roihanul Fiqri. Sosok yang sampai saat ini tidak mau penulis panggil dengan sebutan "Abang". Berawal dari ketidaksengajaan saat perjalanan wisata kemarin, siapa sangka Radifan ternyata menjadi bagian penting dalam hidup penulis. Terima kasih sudah sering memberikan hadiah yang sangat sesuai dengan kesukaan penulis. Meski terkadang tingkahnya sangat menyebalkan hingga sering membuat penulis menangis, terima kasih sudah selalu ada menemani dan menjadi penguat dalam proses pengerjaan skripsi ini hingga selesai. Panjang umur selalu, Fann.
21. Mumtaz Salsabilla Citra Susanti dan Fitri Nur Hidayah. Sahabat trioku dengan panggilan kesayangan Mum dan Nene. Dua manusia yang penulis masih bingung, dengan sifat kalian berdua yang bisa sama persis itu, apalagi bagian menyebalkannya. Meskipun rasa gengsi kalian lebih tinggi untuk mengutarakan kasih sayang, tapi penulis tahu kepedulian kalian lebih dari apa pun. Terima kasih karena selalu memberikan rasa sayang dan cinta lewat tindakan yang "gong" dan ide-ide yang tidak terduga, membuat penulis merasa selalu “hidup”. Penulis sangat bersyukur menjadi bagian dari kehidupan kalian. Terima kasih selalu ada sejak awal sempro hingga skripsi ini selesai. Panjang Umur Selalu ya Mum dan Nene tersayang.
22. Sahabat Sehati, Lailatul Ilmi Silviana. Pemilik nama cantik dengan panggilan kesayangan “siltuttt”. Sahabat sekelas penulis dari awal semester

perkuliahan di mulai hingga skripsi ini selesai. Yang selalu menemani penulis saat-saat penting dan banyak menemani penulis mengelilingi Malang. Terima kasih juga sudah menerima dan memperkenalkan penulis dengan sangat baik di keluarga Silvi dan memberikan pengalaman berkesan keliling Jawa bersama. Meskipun perkenalan kita terasa singkat dalam perjalanan waktu, terima kasih telah menemani proses studi ini hingga akhirnya selesai.

23. Yoza Setya Febriyanti**.** Pemilik nama cantik yang orangnya pun sama cantiknya. Berawal dari kedekatan di kelas Taklim Qur'an saat ma'had, ternyata Yoza sangat selaras pemikirannya dengan penulis hingga menjadi sahabat baik sampai sekarang. Terima kasih telah menemani perjalanan ini, dari awal kuliah hingga akhirnya kita bisa berada di bawah bimbingan dosen yang sama untuk menyelesaikan skripsi ini.
24. Teman-teman seperjuangan satu dosen pembimbing. Atsila, Anissa, Gavrila, Kila dan Iim. Terima kasih sudah selalu bersama dan saling memberikan support satu sama lain tiap kali bimbingan. Terima kasih juga sudah menjadi orang-orang yang selalu meramaikan lab bersama penulis. Kehadiran kalian membuat perjalanan panjang menyelesaikan skripsi ini terasa jauh lebih ringan.
25. Seluruh Sahabat Penulis. Untuk teman-teman Ma'had Kamar 15 (Belva, Nata, Nadia, Faren, Luhan, Lilia, dan Caca), terima kasih atas satu tahun yang berkesan di awal kuliah. Untuk teman-teman kost (Qila, Kak Tere, dan Citra), terima kasih telah menjadi penghilang lelah setelah seharian

berkuliah. Untuk sahabat di Sulawesi (Fahira, Mira, dan Yuni) terima kasih selalu menjadi teman curhat penulis dan teman jalan setiap kali penulis pulang ke Sulawesi. Terima kasih juga untuk teman-teman PKL (Ela, Kirana, dan Rara) atas kebersamaannya. Tak lupa untuk teman-teman KKN (Kei, Nadin, Momy, Ira, Ernita, Aufa, Ghifar, Esa, dan Farhan), terima kasih untuk 40 hari yang sangat membekas. Kehadiran kalian semua telah memberikan warna dalam perjalanan studi penulis hingga di tahap ini.

26. Teman-Teman "*Infinity*" Teknik Informatika angkatan 2022, yang telah menemani penulis atas kebersamaan, semangat, serta kenangan indah yang akan selalu menjadi bagian berharga dari perjalanan studi ini.
27. Semua pihak yang telah memberikan kebaikan dalam berbagai bentuk selama proses penyusunan skripsi ini, meskipun tidak bisa disebutkan satu per satu, penulis mengucapkan terima kasih yang sebesar-besarnya.
28. Teruntuk seseorang yang penulis temui di tahun 2023, yang penulis tidak bisa sebutkan Namanya. Terima kasih telah memilih untuk usai tepat saat penulis sedang menyusun skripsi ini. Kepergian Anda di titik ini justru menjadi pengingat bahwa setiap pertemuan memiliki tujuannya, dan bagi penulis, tujuan itu adalah untuk tetap melangkah dan menyelesaikan apa yang telah dimulai. Terima kasih telah menjadi bagian dari proses yang mendewasakan. Benar adanya bahwa 'setiap orang ada masanya, dan setiap masa ada orangnya.'
29. Terakhir, rasa terima kasih yang sedalam-dalamnya ditujukan kepada diri penulis sendiri, Firna. Seorang perempuan sederhana dengan hati kecil

namun memiliki impian besar. Sebagai anak perempuan pertama dan harapan orang tua, terima kasih telah bertahan sejauh ini dan terus berjalan melewati segala tantangan yang dihadirkan semesta. Tetaplah berani menjadi diri sendiri dan teruslah merasa bangga atas setiap langkah kecil serta pencapaian yang mungkin tidak dirayakan oleh orang lain. Meski terkadang harapan tidak sejalan dengan apa yang diberikan semesta, semoga penerimaan dan rasa syukur selalu menyertai. Janganlah lelah untuk tetap berusaha dan tetaplah berbahagia di mana pun berada. Segala hal dalam diri patut dirayakan, dan semoga diri ini selalu menjadi sosok yang bermanfaat bagi pribadi maupun orang lain. Doa terbaik dipanjatkan semoga langkah-langkah kecil ke depan selalu diperkuat, dikelilingi oleh orang-orang baik, serta mimpi-mimpi yang ada dapat terwujud satu per satu. Aamiin.

Penulis menyadari bahwa skripsi ini masih jauh dari kata sempurna. Oleh karena itu, penulis sangat mengharapkan saran dan kritik yang membangun demi perbaikan di masa mendatang. Akhir kata, semoga skripsi ini dapat memberikan manfaat bagi pembaca dan bagi perkembangan ilmu pengetahuan.

Wassalamualaikum Warahmatullahi Wabarakatuh.

Malang, 20 Desember 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Firna. 2025. **Implementasi Model *Long Short-Term Memory* (LSTM) untuk Klasifikasi *Multi-Label* Ujaran Kebencian pada *Tweet* Bahasa Indonesia.** Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Okta Qomaruddin Aziz, M.Kom (II) Dr. Irwan Budi Santoso, M.Kom.

Kata kunci: *ujaran kebencian, multilabel classification, LSTM, Word2Vec, tweet berbahasa Indonesia.*

Ujaran kebencian di media sosial merupakan permasalahan serius yang berdampak pada kenyamanan dan keharmonisan interaksi daring, khususnya pada platform seperti Twitter. Penelitian ini bertujuan mengklasifikasikan ujaran kebencian secara *Multi-Label* pada *tweet* berbahasa Indonesia menggunakan model *Long Short-Term Memory* (*LSTM*). Dataset diproses melalui tahapan praproses teks dan direpresentasikan menggunakan *word2vec* berbasis metode *Skip-Gram*, kemudian diuji melalui 24 skenario dengan variasi dimensi *embedding*, jumlah unit *LSTM*, *batch size*, dan nilai *threshold*. Performa model diukur menggunakan *subset accuracy*, *hamming loss*, *macro precision*, *macro recall*, dan *macro F1-score*. Hasil penelitian menunjukkan bahwa *threshold* memiliki pengaruh paling signifikan terhadap keseimbangan performa, di mana nilai yang lebih tinggi meningkatkan *precision* dan *subset accuracy* serta menurunkan *hamming loss*. Konfigurasi terbaik diperoleh pada Skenario 18 (100d–64H–64B–0.3t) dengan *subset accuracy* 52,16%, *hamming loss* 17,15%, dan *macro F1-score* 61,44%, serta didukung oleh validasi *K-Fold* yang menunjukkan performa stabil. Hasil ini menegaskan bahwa model *LSTM* efektif digunakan untuk klasifikasi *Multi-Label* ujaran kebencian dengan pemilihan *hyperparameter* yang tepat.

ABSTRACT

Firna. 2025. **Implementation of Long Short-Term Memory (LSTM) Model for *Multi-Label* Hate Speech Classification on Indonesian *Tweet*s.** Thesis. Informatics Engineering Study Program, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University Malang. Supervisor: (I) Okta Qomaruddin Aziz, M.Kom (II) Dr. Irwan Budi Santoso, M.Kom.

Key words: *hate speech, multilabel classification, LSTM, Word2Vec, Indonesian tweets.*

Hate speech on social media is a serious issue that affects the quality and harmony of online interactions, particularly on platforms such as Twitter. This study aims to perform *Multi-Label* hate speech classification on Indonesian-language *tweet*s using a Long Short-Term Memory (LSTM) model. The dataset was processed through text preprocessing stages and represented using Word2Vec based on the Skip-Gram method. Model testing was conducted through a hyperparameter tuning process involving 24 experimental scenarios that combined embedding dimensions, the number of LSTM units, batch size, and threshold values. Model performance was measured using subset accuracy, hamming loss, macro precision, macro recall, and macro F1-score. The results indicate that the threshold parameter has the most significant impact on performance balance, where higher threshold values improve precision and subset accuracy while reducing hamming loss. The best configuration was achieved in Scenario 18 (100d–64H–64B–0.3t), yielding a subset accuracy of 52.16%, a hamming loss of 17.15%, and a macro F1-score of 61.44%, supported by K-Fold validation that demonstrated stable performance. These findings confirm that the LSTM model is effective for *Multi-Label* hate speech classification with appropriate hyperparameter selection.

مستخلص البحث

ف فيرنا. ٢٠٢٥. **تنفيذ نموذج الذاكرة طويلة وقصيرة المدى (Long Short-Term Memory – LSTM) لتصنيف خطاب الكراهية متعدد التسميات في التغريدات باللغة الإندونيسية**. رسالة جامعية. قسم هندسة المعلوماتية، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف الأول: أوكتا قمار الدين عزيز، ماجستير في علوم الحاسوب، الثاني: الدكتور إروان بودي سانتوسو، ماجستير في علوم الحاسوب.

الكلمات المفتاحية: خطاب الكراهية، التصنيف متعدد التسميات، LSTM، Word2Vec، التغريدات باللغة الإندونيسية.

يُعدّ خطاب الكراهية في وسائل التواصل الاجتماعي مشكلةً خطيرة تؤثر سلبًا على راحة المستخدمين وانسجام التفاعلات الرقمية، لا سيما على منصات مثل تويتر. يهدف هذا البحث إلى تصنيف خطاب الكراهية متعدد التسميات في التغريدات المكتوبة باللغة الإندونيسية باستخدام نموذج الذاكرة طويلة وقصيرة المدى .(LSTM) تمّت معالجة البيانات من خلال مراحل المعالجة المسبقة للنص، ثم تمثيلها باستخدام نموذج Word2Vec المعتمد على طريقة Skip-Gram ، وبعد ذلك تم اختبار النموذج عبر 24 سيناريوًا مع اختلاف أبعاد التضمين، وعدد وحداتLSTM ، وحجم الدفع (Batch Size)، وقيم العتبة .(Threshold) تم تقييم أداء النموذج باستخدام مقاييس الدقة الجزئية (Subset Accuracy)، وخسارة هامنج (Hamming Loss)، والدقة الكلية (Macro Precision)، والاستدعاء الكلي(Macro Recall)، ودرجة F1 الكلية .(Macro F1-Score) أظهرت نتائج البحث أن قيمة العتبة لها التأثير الأكثر أهمية على توازن الأداء، حيث تؤدي القيم الأعلى إلى زيادة الدقة الجزئية والدقة الكلية، وتقليل خسارة هامنج. وقد تم الحصول على أفضل أداء في السيناريو الثامن عشر (100d–64H–64B–0.3t) مع دقة جزئية بلغت 52.16%، وخسارة هامنج بنسبة 17.15%، ودرجة F1 كلية بنسبة 61.44%، كما دعمت نتائج التحقق باستخدام K-Foldاستقرار أداء النموذج. تؤكد هذه النتائج أن نموذج LSTM فعّال في تصنيف خطاب الكراهية متعدد التسميات عند اختيار معاملات الضبط المناسبة.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam era digital yang semakin berkembang, media sosial telah menjadi bagian penting dari kehidupan sehari-hari. Platform seperti Twitter, yang kini dikenal dengan nama X, memberikan ruang bagi individu untuk berbagi informasi, berinteraksi, dan menyuarakan pendapat mereka secara terbuka. Di Indonesia, Twitter merupakan salah satu platform media sosial yang paling populer, dengan lebih dari 24 juta pengguna aktif pada tahun 2023 (Kemp, 2023).

Meskipun memberikan kesempatan bagi orang untuk berbicara dan berdiskusi, Twitter juga menjadi tempat penyebaran ujaran kebencian yang dapat memperburuk hubungan sosial dan meningkatkan ketegangan antar kelompok di masyarakat. Ujaran kebencian ini dapat berupa diskriminasi terhadap ras, agama, individu, kelompok tertentu, serta bentuk makian atau fitnah lainnya, yang jika tidak ditangani dengan baik, berpotensi memengaruhi stabilitas sosial dan menyebabkan polarisasi yang lebih mendalam dalam masyarakat (Kartika & Nurhayati, 2023).

Penelitian ini berfokus pada pengembangan sistem klasifikasi *Multi-Label* ujaran kebencian pada *tweet* berbahasa Indonesia menggunakan model *Long Short-Term Memory* (LSTM). Upaya ini selaras dengan nilai yang diajarkan dalam Al-Qur'an, yaitu pentingnya menjaga tutur kata dan menghormati sesama manusia. Dalam surat Al-Hujurat ayat 11, Allah SWT berfirman:

يٰٓاَيُّهَا الَّذِيْنَ اٰمَنُوْا لَا يَسْخَرْ قَوْمٌ مِّنْ قَوْمٍ عَسٰٓى اَنْ يَّكُوْنُوْا خَيْرًا مِّنْهُمْ وَلَا نِسَاۤءٌ مِّنْ نِّسَاۤءٍ عَسٰٓى اَنْ يَّكُنَّ خَيْرًا مِّنْهُنَّۚ وَلَا تَلْمِزُوْٓا اَنْفُسَكُمْ وَلَا تَنَابَزُوْا بِالْاَلْقَابِۗ بِئْسَ الِاسْمُ الْفُسُوْقُ بَعْدَ الْاِيْمَانِۚ وَمَنْ لَّمْ يَتُبْ فَاُولٰۤىِٕكَ هُمُ الظّٰلِمُوْنَ ﴿١١﴾

"Hai orang-orang yang beriman, janganlah suatu kaum mengolok-olok kaum yang lain (karena) boleh jadi mereka (yang diolok-olok) lebih baik dari mereka (yang mengolok-olok), dan jangan pula wanita-wanita (mengolok-olok) wanita-wanita lain (karena) boleh jadi wanita-wanita (yang diperolok-olokkan) lebih baik dari wanita (yang mengolok-olok), dan janganlah kamu mencela dirimu sendiri dan janganlah kamu panggil memanggil dengan gelar-gelar yang buruk. Seburuk-buruk panggilan ialah (panggilan) yang buruk sesudah iman, dan barang siapa yang tidak bertobat, maka mereka itulah orang-orang yang dzalim." (QS. Al-Hujurat [49]: 11).

Ayat ini mengandung pesan moral agar setiap individu menjaga lisan dan perilaku dalam berinteraksi, baik di dunia nyata maupun di ruang digital. Dalam *Tafsir al-Azhar*, Buya Hamka menjelaskan bahwa larangan mengolok-olok sesama manusia bertujuan untuk mencegah timbulnya permusuhan dan menjaga kehormatan antar sesama. Sifat sombong dan merendahkan orang lain, menurut beliau, merupakan bentuk kesalahan moral yang berakar dari hati yang tidak bersih dan dapat merusak tatanan sosial.

Dengan demikian, penelitian mengenai deteksi ujaran kebencian melalui pendekatan *machine learning* dapat dipandang sebagai bentuk penerapan nilai keislaman dalam konteks teknologi modern. Prinsip untuk mencegah penghinaan dan menjaga keharmonisan sosial sebagaimana terkandung dalam QS. Al-Hujurat ayat 11, menjadi relevan dengan tujuan penelitian ini, yakni membangun sistem yang mampu mengenali ujaran kebencian secara otomatis agar lingkungan digital menjadi lebih santun dan beretika.

Penyebaran ujaran kebencian di media sosial semakin mengkhawatirkan dengan bertambahnya jumlah pengguna aktif dan semakin tingginya volume data yang dihasilkan setiap harinya. Platform seperti Twitter memproduksi ribuan hingga jutaan *tweet* setiap hari, yang menyebar secara cepat ke seluruh dunia. Dengan volume data yang begitu besar, deteksi manual terhadap ujaran kebencian menjadi hampir tidak mungkin dilakukan. Oleh karena itu, sistem otomatis yang dapat mengidentifikasi dan mengklasifikasikan ujaran kebencian secara cepat dan efisien sangat dibutuhkan. Dalam hal ini, *machine learning* dan *deep learning* menawarkan solusi yang sangat potensial karena mampu memproses dan menganalisis data dalam jumlah besar secara otomatis dan menghasilkan hasil klasifikasi yang akurat dalam waktu yang relatif singkat (Ayo *et al*., 2020).

Namun, penerapan *machine learning* pada teks di media sosial seperti Twitter menghadapi beberapa tantangan besar. Salah satu tantangan utama adalah keragaman bahasa yang digunakan oleh pengguna. Pengguna Twitter sering kali menggunakan bahasa tidak baku, slang, singkatan, dan bahkan bahasa gaul yang sangat sulit dipahami oleh model machine learning yang dilatih dengan data teks formal. Penggunaan kata-kata seperti "km" untuk "kamu", "gak" untuk "tidak", atau "baper" untuk "bawa perasaan". Variasi ini menunjukkan kompleksitas bahasa di media sosial yang menjadi tantangan tersendiri dalam klasifikasi teks (Badry Ali Mustofa & Wawan Laksito Yuly Saptomo, 2025).

Recurrent Neural Networks (RNN) sering digunakan dalam pemrosesan teks, karena RNN dirancang untuk menangani data urutan, seperti teks, di mana urutan kata-kata dalam kalimat sangat penting untuk memahami makna

keseluruhan (Nistor *et al*., 2021). Namun, RNN tradisional menghadapi masalah *vanishing gradient* yang menyulitkan dalam mengingat informasi jangka panjang, terutama pada teks yang panjang atau kompleks (Noh, 2021).

Untuk mengatasi masalah ini, dikembangkanlah *Long Short-Term Memory* (LSTM), varian dari RNN yang dapat mengingat informasi dalam jangka panjang dan mengatasi masalah *vanishing gradient* (Noh, 2021). LSTM memiliki kemampuan untuk menangkap hubungan antarkata dalam urutan teks yang lebih panjang dan lebih kompleks, yang sangat penting dalam analisis teks di Twitter, di mana *tweet* sering kali mengandung lebih dari satu kalimat dan memerlukan pemahaman konteks yang lebih dalam (Naveen P, 2025).

LSTM menjadi pilihan yang sangat tepat untuk tugas *Multi-Label classification*, yaitu tugas di mana satu *tweet* dapat mengandung lebih dari satu kategori ujaran kebencian. Sebagai contoh, satu *tweet* dapat mengandung ujaran kebencian berbasis ras dan serangan terhadap individu atau kelompok secara bersamaan (Walsh & Greaney, 2025) (Handayani & Hilmansyah Gani, 2023). Dengan menggunakan LSTM, model dapat mengklasifikasikan *tweet* tersebut ke dalam beberapa label atau kategori sekaligus, tanpa kehilangan konteks antar kategori yang relevan. Pendekatan ini sangat penting karena ujaran kebencian sering kali melibatkan berbagai bentuk diskriminasi dalam satu kalimat yang harus diklasifikasikan dengan tepat (Bisht *et al*., 2020).

Untuk memastikan bahwa model LSTM yang dikembangkan dapat mengklasifikasikan berbagai kategori ujaran kebencian dengan akurat, evaluasi menjadi langkah kunci dalam penelitian ini. Evaluasi model memungkinkan

pengukuran kinerja model dalam *Multi-Label classification*, yang menilai apakah model dapat menangani konteks yang kompleks, seperti variasi bahasa tidak baku dan slang yang digunakan dalam *tweet* (Badry Ali Mustofa & Wawan Laksito Yuly Saptomo, 2025). Dalam hal ini, metrik evaluasi seperti *precision*, *recall*, *F1-score*, *Hamming Loss*, dan *Subset Accuracy* sangat relevan untuk menilai kemampuan model dalam mendeteksi beberapa kategori ujaran kebencian dalam satu *tweet* (Al-Smadi, 2024). Evaluasi juga membantu mengidentifikasi potensi *overfitting*, di mana model bisa tampil baik pada data pelatihan namun tidak dapat menggeneralisasi pada data yang tidak terlihat sebelumnya (Rawal, 2025).

Penelitian ini juga berfokus pada penyetelan *hyperparameter*, yang merupakan bagian penting dalam mengoptimalkan kinerja model. Penyetelan *hyperparameter* ini mencakup pemilihan nilai terbaik untuk parameter-parameter penting seperti dimensi *embedding*, jumlah unit LSTM, tingkat *dropout*, dan *learning rate*. Penyesuaian parameter-parameter ini sangat memengaruhi kemampuan model dalam menangani data yang besar dan kompleks, seperti *tweet* yang mengandung banyak kategori ujaran kebencian sekaligus (Handayani & Hilmansyah Gani, 2023).

Penelitian terdahulu, seperti yang dilakukan oleh (Angger Saputra & Sibaroni, 2025), menunjukkan bahwa LSTM sangat efektif dalam mendeteksi ujaran kebencian pada *Multi-Label classification*, terutama pada dataset Twitter yang kompleks. Keunggulan LSTM terletak pada kemampuannya mengingat informasi jangka panjang dan menangkap konteks antar kata dalam *tweet*, yang penting dalam klasifikasi *Multi-Label* di mana satu *tweet* bisa mengandung lebih

dari satu kategori ujaran kebencian. Selain itu, penyetelan hyperparameter yang hati-hati, seperti yang dijelaskan oleh (Handayani & Hilmansyah Gani, 2023), dapat meningkatkan kinerja model dalam mengklasifikasikan berbagai kategori dalam satu *tweet*. Penelitian ini akan mengikuti pendekatan tersebut untuk mengoptimalkan model LSTM dalam *Multi-Label hate speech classification* pada data *tweet* berbahasa Indonesia.

Penelitian ini bertujuan menerapkan model *Long Short-Term Memory* (LSTM) pada tugas klasifikasi *Multi-Label* ujaran kebencian pada *tweet* berbahasa Indonesia. Mengingat keragaman bahasa di Twitter, termasuk bahasa tidak baku, *slang*, dan singkatan, model ini dilatih untuk menangani variasi tersebut. Fokus penelitian juga pada penyetelan *hyperparameter*, seperti dimensi embedding, jumlah unit LSTM, batch size, dan nilai ambang batas atau *threshold* dapat memengaruhi hasil klasifikasi. Evaluasi dilakukan menggunakan metrik seperti *precision*, *recall*, *F1-score*, *Hamming Loss*, dan *Subset Accuracy* untuk memastikan efektivitas model dalam klasifikasi. Penelitian ini mengacu pada penelitian terdahulu yang menunjukkan efektivitas LSTM dalam *Multi-Label classification* pada data Twitter yang kompleks dan menekankan pentingnya penyetelan *hyperparameter*. Dengan demikian, diharapkan model yang dikembangkan dapat memberikan solusi yang lebih akurat dan efisien dalam mendeteksi ujaran kebencian di media sosial.

1.2 Pernyataan Masalah

Bagaimana performa *Long Short-Term Memory* (LSTM) dalam mendeteksi berbagai kategori ujaran kebencian pada *tweet* berbahasa Indonesia yang bersifat

Multi-Label, serta sejauh mana penyetelan *hyperparameter* dapat memengaruhi hasil klasifikasinya?

1.3 Tujuan Penelitian

Mengukur performa model LSTM dalam melakukan klasifikasi *Multi-Label* ujaran kebencian pada *tweet* berbahasa Indonesia. Dan mengukur pengaruh penyetelan *hyperparameter* (dimensi *embedding*, jumlah unit LSTM, *batch size*, dan nilai *threshold*) terhadap hasil klasifikasi model.

1.4 Manfaat Penelitian

Dengan dilakukannya penelitian ini, diharapkan dapat memberi beberapa manfaat yaitu:

1. Hasil penelitian ini diharapkan dapat menambah pengetahuan tentang penerapan model *Long Short-Term Memory* (LSTM) pada kasus klasifikasi *Multi-Label*, khususnya dalam mendeteksi ujaran kebencian pada teks berbahasa Indonesia. Penelitian ini juga dapat menjadi bahan rujukan bagi penelitian selanjutnya yang membahas penerapan metode *deep learning* pada bidang *Natural Language Processing* (NLP).
2. Menambah referensi penelitian terkait implementasi model LSTM dalam menangani teks berbahasa Indonesia yang kaya variasi (slang, tidak baku, singkatan).
3. Mendorong literasi digital masyarakat, penelitian ini secara tidak langsung menunjukkan bahwa ujaran kebencian bisa dipetakan dan dideteksi, sehingga

diharapkan memengaruhi kesadaran pengguna untuk lebih berhati-hati dalam berkomunikasi.

1.5 Batasan Masalah

Karena ruang lingkup permasalahan yang diteliti cukup luas, penulis menetapkan batasan penelitian sebagai berikut:

1. Data yang digunakan dalam penelitian ini berupa *tweet* berbahasa Indonesia yang mengandung ujaran kebencian. Dataset diperoleh dari platform Kaggle, yaitu *Indonesian Abusive and Hate Speech Twitter Text* (**https://www.kaggle.com/datasets/ilhamfp31/indonesian-abusive-and-hate-speech-twitter-text**) dan *Indonesian Stoplist* (**https://www.kaggle.com/datasets/oswinrh/indonesian-stoplist**), yang telah melalui proses pelabelan sebelumnya sehingga dapat diklasifikasikan ke dalam beberapa kategori.
2. Penelitian ini berfokus pada klasifikasi *Multi-Label*, di mana setiap *tweet* dapat diklasifikasikan ke dalam lebih dari satu kategori ujaran kebencian dengan 6 label kelas yang digunakan.

BAB II

STUDI PUSTAKA

2.1 Ujaran Kebencian

Ujaran kebencian adalah ungkapan yang mengandung rasa benci, penghinaan, atau permusuhan terhadap orang atau kelompok tertentu berdasarkan identitas seperti agama, ras, etnis, jenis kelamin, ideologi, atau pandangan politik. Dalam komunikasi digital, ujaran kebencian muncul sebagai perilaku verbal yang agresif dan berkembang pesat seiring meningkatnya penggunaan media sosial (Ash-Shidiq & Pratama, 2021). Hal ini membuktikan bahwa kemudahan berekspresi yang diberikan teknologi tidak selalu menghasilkan komunikasi yang baik, justru sering kali dimanfaatkan untuk meluapkan kebencian secara terang-terangan (Susanti, 2022). Dengan demikian, ujaran kebencian tidak hanya menggambarkan pandangan pribadi pengguna, tetapi juga menjadi bukti dari polarisasi sosial yang ada di masyarakat (Andriani *et al*., 2024).

Ujaran kebencian di media sosial bisa dikenali dari ciri-ciri bahasanya. Tamam (2021) menjelaskan bahwa ujaran kebencian biasanya menggunakan bahasa yang menyerang, merendahkan, atau menghasut orang lain untuk membenci kelompok tertentu. Dalam konteks politik, ujaran kebencian sering digunakan untuk menciptakan pandangan negatif terhadap pihak lawan, sebuah pola yang terlihat jelas dalam studi Jamilah & Wahyuni (2020) selama masa pemilu. Penyebaran ujaran kebencian juga dipercepat oleh sistem algoritma media sosial yang mengarahkan pengguna pada konten serupa, menciptakan *echo chamber* dan

filter bubble (Andriani *et al*., 2024). Akibatnya, sikap fanatik dan tidak toleran menjadi lebih kuat karena pengguna hanya mendapat informasi yang sejalan dengan keyakinan mereka.

Dampak sosial ujaran kebencian sangat luas, mencakup aspek psikologis, sosial, dan budaya. Menurut Kartika & Nurhayati (2023), penyebarannya di media sosial bisa meningkatkan perpecahan sosial dan mengurangi toleransi antarindividu. Menurut Siregar (2023), ujaran kebencian juga bisa melemahkan kebersamaan dan ikatan sosial dalam masyarakat digital. Dari sisi psikologis, penelitian oleh Afif *et al*. (2021) menunjukkan bahwa para korban ujaran kebencian sering mengalami stres, cemas, dan merasa tidak aman di dunia maya. Selain itu, penelitian Andriani *et al*. (2024) menunjukkan hubungan antara literasi digital dan kecenderungan menyebarkan ujaran kebencian, di mana semakin tinggi pemahaman digital seseorang, semakin kecil kecenderungannya untuk melakukan hal tersebut.

Ujaran kebencian berdampak luas, tidak hanya merusak hubungan sosial, tetapi juga dapat menimbulkan tekanan psikologis pada korban serta memperparah konflik antarindividu dan antarkelompok. Oleh karena itu, upaya pencegahan perlu dilakukan secara terstruktur melalui peningkatan literasi digital, penegakan hukum yang konsisten, serta penerapan teknologi pendeteksi otomatis oleh platform media sosial. Dengan pendekatan yang menyeluruh, ekosistem digital yang aman, sehat, dan saling menghormati dapat tercipta.

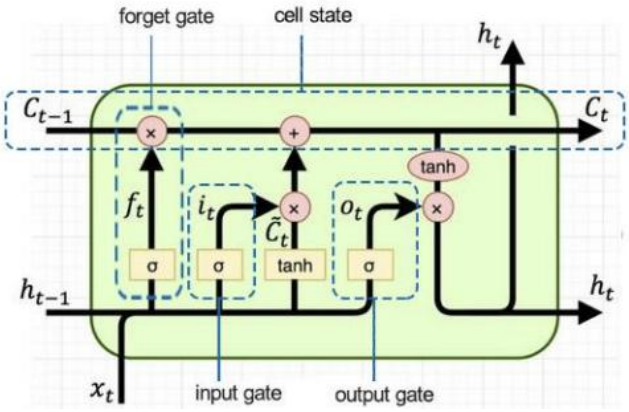
2.2 *Long Short-Term Memory* (LSTM)

Long Short-Term Memory (LSTM) merupakan salah satu bentuk jaringan saraf tiruan yang dibuat guna mengatasi kekurangan pada *Recurrent Neural Network* (RNN), khususnya masalah hilangnya gradien (*vanishing gradient*) dan keterbatasan dalam memproses ketergantungan jangka panjang (*long-term dependency*). LSTM ini pertama kali diperkenalkan oleh Hochreiter dan Schmidhuber pada tahun 1997, hingga kini model ini digunakan secara luas dalam pengolahan data sekuensial, termasuk teks, suara, dan sinyal waktu (Ambari *et al*., 2025). Dengan cara kerja yang ada di dalamnya, LSTM mampu menjaga informasi yang relevan untuk waktu yang panjang sekaligus menyingkirkan hal-hal yang tidak penting. Keunggulan ini menjadikan LSTM lebih efektif daripada RNN tradisional yang sering gagal mengolah urutan data panjang secara baik (Utama, 2023).

LSTM memiliki unit memori internal yang disebut *memory cell*. Setiap *memory cell* berfungsi untuk menyimpan, memperbarui, dan menghapus informasi sesuai kebutuhan pembelajaran. Di dalamnya terdapat dua komponen utama, yaitu *cell state* (C_t) dan *hidden state* (h_t). *Cell state* menjadi jalur utama penyimpanan informasi jangka panjang, sedangkan *hidden state* menyimpan informasi jangka pendek dan berperan sebagai keluaran jaringan pada waktu ke-t (Natzir & Jatiprasetya, 2025). Kemampuan tersebut membuat LSTM lebih efektif dan stabil dalam mempelajari pola data yang berbentuk urutan panjang dibandingkan dengan RNN konvensional (Cahyani *et al*., 2023).

Konsep utama dari LSTM terletak pada cell state dan empat gerbang utama (gates) yang berfungsi mengatur aliran informasi di dalam jaringan. Cell state berperan sebagai jalur utama yang membawa informasi penting dari satu cell ke cell berikutnya, sehingga konteks data tetap terjaga sepanjang proses pembelajaran (Marietta *et al*., 2025).

Diagram pada Gambar 2.1 berikut menggambarkan mekanisme internal dari satu unit LSTM. Diagram tersebut memperlihatkan bagaimana *cell state* (C_t)berinteraksi dengan empat gerbang utama, yaitu forget gate (f_t), input gate (i_t), candidate cell state $(\tilde{C}_t)$, dan output gate (o_t) (Susilo *et al*., 2025). Keempat komponen ini bekerja secara terkoordinasi untuk mengontrol aliran informasi pada setiap langkah waktu, sehingga *cell state* (C_t)tetap stabil dan model mampu mengenali pola ketergantungan jangka panjang secara lebih akurat.



Gambar 2.1 Unit LSTM
Sumber: Susilo et al (2025)

Pada Gambar 2.1 diatas, menunjukkan bahwa setiap garis menggambarkan aliran data dari satu komponen ke komponen lainnya. Simbol kotak kecil bertanda σ merepresentasikan fungsi aktivasi sigmoid, yang menghasilkan nilai antara 0 dan 1 untuk mengatur seberapa banyak informasi diteruskan ke tahap berikutnya. Kotak

bertanda *tanh* menghasilkan kandidat nilai baru yang dibatasi antara -1 dan 1. Lingkaran bertanda "×" menunjukkan operasi perkalian elemen per elemen, sedangkan lingkaran bertanda "+" menunjukkan operasi penjumlahan untuk memperbarui *cell state*. Selain itu, garis bercabang menunjukkan bahwa sebagian informasi disalin dan dikirim ke beberapa jalur paralel secara bersamaan. Dengan rancangan seperti ini, LSTM dapat melakukan seleksi informasi secara terkontrol pada setiap langkah waktu. Mekanisme tersebut membantu mengurangi risiko *vanishing gradient* dan memastikan konteks urutan data tetap terjaga dengan baik.

Keempat gerbang tersebut berperan mengatur aliran informasi di dalam jaringan, baik dalam proses penyimpanan, pembaruan, maupun pengeluaran informasi. Penjelasan berikut memaparkan fungsi dan formulasi matematis dari masing-masing gerbang.

1. *Forget Gate*

Forget Gate berfungsi untuk menentukan bagian informasi dari *cell state* sebelumnya (C_{t-1}) yang masih relevan dan perlu dipertahankan, serta bagian mana yang tidak lagi diperlukan dan harus dihapus. Tahap ini menjadi langkah awal yang penting, karena tidak semua informasi masa lalu memiliki kontribusi terhadap konteks data saat ini. Oleh karena itu, LSTM melakukan penyaringan memori menggunakan fungsi aktivasi sigmoid.

Gerbang ini menerima dua masukan utama, yaitu *hidden state* dari waktu sebelumnya (h_{t-1}) dan *input* saat ini (x_t). Kedua masukan ini digabung (*concatenate*) kemudian dikalikan dengan bobot W_f, dijumlahkan dengan bias b_f, dan selanjutnya dilewatkan ke fungsi sigmoid. Hasil keluaran fungsi sigmoid

berada pada rentang 0 hingga 1, nilai yang mendekati 0 menunjukkan bahwa informasi tersebut akan dilupakan, sedangkan nilai yang mendekati 1 menunjukkan bahwa informasi tersebut tetap dipertahankan dalam memori.

Persamaan 2.1 menunjukkan proses perhitungan yang digunakan untuk menentukan seberapa besar bagian informasi dari memori sebelumnya dipertahankan:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (2.1)$$

Keterangan:
f_t : *forget gate.*
σ : fungsi aktivasi *sigmoid.*
W_f : bobot *forget gate.*
h_{t-1} : nilai *hidden state timestep* sebelumnya.
x_t : nilai masukan.
b_f : bias *forget gate.*

Mekanisme ini memastikan agar jaringan tidak membawa informasi yang tidak relevan ke tahap selanjutnya. Nilai f_t yang dihasilkan kemudian digunakan bersama keluaran dari *input gate* dalam proses pembaruan *cell state*.

2. *Input Gate* (i_t)

Input Gate berfungsi menambahkan informasi baru ke dalam *cell state*. Gerbang ini mengatur seberapa besar kontribusi data baru yang akan disimpan ke dalam memori jangka panjang, sehingga jaringan dapat menyesuaikan diri terhadap pola baru tanpa kehilangan konteks yang sudah ada.

Prosesnya melibatkan dua fungsi utama. Pertama, fungsi sigmoid menghasilkan nilai i_t yang menentukan seberapa besar informasi baru akan dimasukkan. Kedua, fungsi *tanh* menghasilkan *candidate cell state* $(\tilde{C}_t)$, yaitu

vektor berisi kandidat informasi baru dengan rentang nilai $[-1,1]$. Kedua hasil ini akan dikombinasikan untuk memperbarui memori jaringan.

Proses tersebut dirumuskan secara matematis pada Persamaan 2.2 dan 2.3:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2.2}$$

$$\tilde{C}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right) \tag{2.3}$$

Keterangan:
i_t : *input gate.*
σ : fungsi aktivasi *sigmoid.*
W_i : bobot *input gate.*
h_{t-1} : nilai *hidden state timestep* sebelumnya.
x_t : nilai masukan.
b_i : bias *input gate.*
$\tilde{C}_t$: kandidat *cell state* baru.
tanh : fungsi *tanh.*
W_c : bobot.
b_c : bias.

Nilai i_t berperan sebagai penimbang (*weight*) terhadap informasi baru, sementara $\tilde{C}_t$ memuat kandidat informasi yang akan disimpan ke dalam memori. Kedua hasil ini nantinya digabungkan untuk memperbarui *cell state*, di mana *input gate* bertindak sebagai penghubung antara informasi lama dan informasi baru yang relevan.

3. *Cell State*

Cell State diperbarui dengan menggabungkan informasi lama yang masih penting dan informasi baru yang relevan. Proses ini menjadi inti dari kemampuan LSTM dalam menjaga konteks jangka panjang tanpa kehilangan informasi dari waktu sebelumnya. Proses ini dilakukan dengan memanfaatkan keluaran dari dua gerbang sebelumnya, *forget gate* dan *input gate.* Pertama, nilai *forget gate* (f_t)

dikalikan dengan *cell state* sebelumnya (C_{t-1}) untuk mempertahankan bagian informasi lama yang penting. Kemudian, hasil dari *input gate* (i_t) dikalikan dengan *candidate cell state* ($\tilde{C}_t$) untuk menambahkan informasi baru ke dalam memori. Kedua hasil tersebut dijumlahkan untuk membentuk *cell state* baru (C_t). Proses pembaruan ini ditunjukkan pada Persamaan 2.4 berikut:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (2.4)$$

Keterangan:
C_t : *cell state.*
f_t : *forget gate.*
C_{t-1} : nilai *cell state timestep* sebelumnya.
i_t : *input gate.*
$\tilde{C}_t$: kandidat *cell state* baru.

Persamaan ini menggambarkan bahwa *cell state* membawa kombinasi informasi lama yang dianggap penting dan informasi baru yang relevan dengan konteks data saat ini. Dengan mekanisme ini, LSTM mampu menjaga kestabilan memori jangka panjang tanpa kehilangan konteks dari data sebelumnya.

4. *Output Gate*

Output Gate menentukan nilai keluaran (*hidden state*) (h_t) dari unit LSTM pada waktu ke-t. Gerbang ini mengatur bagian dari memori internal yang akan ditampilkan sebagai keluaran aktual dan diteruskan ke langkah waktu berikutnya.

Proses ini memastikan bahwa hasil keluaran yang dihasilkan mempertimbangkan baik konteks lama maupun informasi terbaru yang telah diperbarui. Pertama, fungsi sigmoid menghasilkan nilai o_t yang menentukan seberapa besar bagian dari *cell state* yang akan dikeluarkan. Kemudian, *cell state* (C_t) diproses menggunakan fungsi *tanh* untuk menormalkan nilainya agar tetap berada dalam rentang $[-1,1]$. Hasil dari kedua fungsi ini dikalikan untuk

membentuk *hidden state* baru (h_t). Rumus perhitungannya ditunjukkan pada Persamaan 2.5 dan 2.6:

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{2.5}$$

$$h_t = o_t \cdot tanh\left(C_t\right) \tag{2.6}$$

Keterangan:
o_t : *output gate.*
σ : fungsi aktivasi *sigmoid.*
W_o : bobot *output gate.*
h_{t-1} : nilai *hidden state timestep* sebelumnya.
x_t : nilai masukan.
b_o : bias *output gate.*
h_t : *hidden state* baru.
tanh : fungsi *tanh.*
C_t : *cell state.*

Nilai h_t inilah yang menjadi keluaran akhir dari unit LSTM pada waktu ke-t, sekaligus menjadi masukan bagi unit LSTM pada waktu ke-$(t+1)$. Dengan cara ini, LSTM dapat mempertahankan konteks informasi dalam urutan data dengan stabil dan berkelanjutan.

Keempat gerbang tersebut bekerja secara berurutan dan saling mendukung. *Forget gate* berfungsi menghapus informasi yang tidak relevan, *input gate* menambahkan pengetahuan baru, *cell state* memperbarui memori utama, dan *output gate* menghasilkan keluaran yang diteruskan ke waktu berikutnya.

Berdasarkan beberapa penelitian yang telah dibahas pada bagian sebelumnya, penulis merangkum hasil temuan tersebut dalam bentuk perbandingan yang disajikan pada Tabel 2.1.

Tabel 2.1 Penelitian-Penelitian terkait Model LSTM

No.	Judul Penelitian	Peneliti	Metode yang digunakan	Hasil
1.	*Prediction of Budget Planning Using the Long Short-Term Memory*	(Ambari *et al*., 2025)	*Long Short-Term Memory (LSTM)*	Model LSTM mampu memprediksi perencanaan anggaran dengan akurasi tinggi dan stabilitas prediksi baik pada data deret waktu nonlinier.
2.	Pendekatan Deep Learning Menggunakan Metode LSTM untuk Prediksi Harga Bitcoin	(Utama, 2023)	*Long Short-Term Memory (LSTM)*	Model LSTM menghasilkan nilai RMSE 77.74 dan MAE 278.33, menunjukkan akurasi prediksi tinggi terhadap fluktuasi harga Bitcoin.
3.	Prediksi Harga Cryptocurrency XLM Menggunakan Metode Deep Learning LSTM dan GRU	(Natzir & Jatiprasetya, 2025)	*LSTM dan GRU*	GRU unggul dengan MAPE 3,61%; LSTM 4,56%, keduanya efektif mengenali pola tren jangka panjang harga XLM.
4.	Implementasi Metode *Long Short-Term Memory* (LSTM) untuk Memprediksi Harga Bahan Pokok Nasional	(Cahyani *et al*., 2023)	*LSTM dengan Optimasi ADAM, AdaGrad, dan RMSProp*	Optimasi ADAM menghasilkan RMSE 0.0492 dan R^2 0.8852, menunjukkan performa prediksi harga bahan pokok sangat baik.
5.	Prediksi Penjualan untuk Optimasi Stok Produk Menggunakan Algoritma *Long Short-Term Memory*	(Susilo *et al*., 2025)	*Long Short-Term Memory (LSTM)*	Model LSTM mencapai MAPE 3,60%, tergolong sangat akurat dalam prediksi penjualan dan membantu optimasi stok produk.

Pada Tabel 2.1 ditampilkan beberapa referensi penelitian terdahulu yang berkaitan dengan penerapan metode LSTM dalam berbagai bidang prediksi dan klasifikasi. Hasil dari beberapa penelitian tersebut menunjukkan bahwa metode LSTM mampu memberikan performa prediksi yang unggul karena kemampuannya dalam mengenali pola temporal dan mempertahankan informasi jangka panjang.

Penelitian yang dilakukan oleh Ambari *et al*. (2025)membahas penggunaan model LSTM dalam memprediksi perencanaan anggaran berdasarkan data historis

keuangan. Penelitian ini bertujuan menguji kemampuan LSTM dalam mengenali pola temporal kompleks untuk mendukung pengambilan keputusan dalam perencanaan keuangan. Hasil penelitian menunjukkan bahwa LSTM mampu menghasilkan prediksi yang stabil dengan tingkat kesalahan rendah, sehingga terbukti efektif dalam menangani data deret waktu nonlinier.

Selanjutnya, Utama (2023) menerapkan pendekatan *Deep Learning* berbasis LSTM untuk melakukan prediksi terhadap pergerakan harga Bitcoin menggunakan data historis harga harian. Penelitian ini berfokus pada pengujian kemampuan LSTM dalam menangani volatilitas data aset digital yang tinggi. Berdasarkan hasil pengujian, model LSTM berhasil mencapai nilai RMSE sebesar 77.74 dan MAE sebesar 278.33, yang menunjukkan akurasi tinggi dalam memprediksi fluktuasi harga dan kestabilan dalam proses pembelajaran.

Kemudian, penelitian oleh Natzir & Jatiprasetya (2025) membandingkan performa algoritma LSTM dan GRU dalam melakukan prediksi harga mata uang kripto Stellar Lumens (XLM). Tujuan penelitian ini adalah untuk menilai efektivitas kedua model dalam menangkap pola jangka panjang pada data harga kripto. Hasilnya menunjukkan bahwa GRU memperoleh nilai MAPE sebesar 3,61%, sedangkan LSTM memiliki nilai MAPE sebesar 4,56%. Meskipun GRU sedikit lebih unggul, model LSTM tetap menunjukkan kemampuan yang baik dalam memahami hubungan temporal antar data serta menghasilkan prediksi yang konsisten.

Penelitian berikutnya dilakukan oleh Cahyani *et al*. (2023) yang menerapkan model LSTM dengan beberapa algoritma optimasi, seperti ADAM,

AdaGrad, dan RMSProp, untuk meningkatkan akurasi dalam memprediksi data ekonomi. Hasil penelitian menunjukkan bahwa optimasi menggunakan ADAM memberikan performa terbaik dengan nilai RMSE sebesar 0.0492 dan R^2 sebesar 0.8852. Hal ini membuktikan bahwa LSTM yang dikombinasikan dengan algoritma optimasi tertentu dapat meningkatkan efektivitas prediksi data deret waktu secara signifikan.

Terakhir, penelitian oleh Susilo *et al*. (2025) membahas penerapan algoritma LSTM untuk melakukan prediksi penjualan dalam upaya optimasi stok produk di sektor distribusi. Penelitian ini bertujuan membantu perusahaan memperkirakan permintaan dan mengelola persediaan dengan lebih efisien. Hasil penelitian menunjukkan bahwa model LSTM mencapai nilai MAPE sebesar 3,60%, yang dikategorikan sangat akurat dan efektif dalam memprediksi volume penjualan serta mendukung proses pengambilan keputusan pada manajemen inventori.

2.3 Klasifikasi *Multi-Label*

Klasifikasi adalah salah satu teknik dasar dalam pembelajaran mesin *(machine learning)* yang bertujuan untuk mengategorikan data ke dalam kelas-kelas tertentu. Kelas-kelas ini merujuk pada kategori yang sudah didefinisikan sebelumnya, yang dalam konteks klasifikasi, dapat mencakup berbagai jenis atau label yang relevan sesuai dengan karakteristik data yang akan dianalisis. Misalnya, dalam klasifikasi email, kelas yang mungkin ada adalah "spam" dan "bukan spam." Teknik ini memiliki peran penting karena memungkinkan sistem untuk mengidentifikasi pola dalam data, yang kemudian digunakan untuk mengelompokkan data ke dalam kategori yang sesuai.

Istilah "klasifikasi" sendiri berasal dari "*classificatie*" dalam bahasa Belanda dan "*classification*" dalam bahasa Prancis, yang digunakan untuk menggambarkan proses pengelompokan data menurut kategori tertentu Dalam konteks machine learning, klasifikasi sangat penting karena membantu sistem untuk memahami dan memproses data dalam berbagai aplikasi, seperti deteksi spam, analisis sentimen, atau pengenalan wajah. Misalnya, dalam analisis sentimen, teks dapat dikategorikan ke dalam dua kelas, yaitu "positif" atau "negatif" (Dharmawan *et al*., 2023; Syahid *et al*., 2025).

Pada dasarnya, klasifikasi teks adalah penerapan teknik klasifikasi untuk mengkategorikan teks atau dokumen berdasarkan isi atau konten teks tersebut. Dalam klasifikasi teks, model pembelajaran mesin dilatih menggunakan data yang telah diberi label, untuk mengenali pola atau karakteristik tertentu dalam teks dan kemudian mengklasifikasikan teks tersebut ke dalam kategori yang sesuai (Ezzat *et al*., 2023). Proses klasifikasi teks ini banyak diterapkan dalam berbagai bidang, mulai dari analisis sentimen untuk mengidentifikasi opini positif atau negatif dalam teks, spam filtering untuk memisahkan email yang relevan dari spam, hingga deteksi ujaran kebencian dalam komunikasi digital (Huan *et al*., 2022).

Klasifikasi juga dapat dibedakan menjadi dua jenis berdasarkan jumlah label yang dihasilkan, yaitu klasifikasi tunggal *(single-label classification)* dan klasifikasi jamak *(Multi-Label classification)*. Pada klasifikasi tunggal, setiap data hanya dapat memiliki satu label kelas. Misalnya, sebuah *tweet* hanya dapat dikategorikan sebagai ujaran kebencian *atau* bukan ujaran kebencian. Sebaliknya, pada klasifikasi jamak *(Multi-Label classification)*, sebuah data dapat memiliki

lebih dari satu label karena kontennya mungkin memuat lebih dari satu makna atau konteks sekaligus (Syifa *et al*., 2025).

Namun, klasifikasi *Multi-Label* memiliki tantangan lebih besar dibandingkan klasifikasi tunggal. Salah satu tantangan utamanya adalah adanya ketergantungan antar label *(label correlation)*, yaitu ketika satu label sering muncul bersamaan dengan label lainnya. Misalnya, dalam deteksi ujaran kebencian, teks yang mengandung kebencian terhadap satu kelompok seringkali juga mengandung kebencian terhadap individu atau kelompok lain. Oleh karena itu, model pembelajaran mesin harus bisa menangkap hubungan antar label ini agar dapat meningkatkan akurasi dalam klasifikasi (Li *et al*., 2023). \

Selain itu, tantangan lain adalah skala label yang besar atau *extreme Multi-Label classification*, di mana jumlah label dapat mencapai ribuan, seperti pada sistem rekomendasi berita atau penelitian ilmiah (Zhang *et al*., 2024). Untuk mengatasi tantangan ini, berbagai pendekatan telah dikembangkan, antara lain metode *problem transformation*, *algorithm adaptation*, dan *label embedding*.

Pada pendekatan *problem transformation*, masalah *Multi-Label* diubah menjadi beberapa masalah klasifikasi biner menggunakan teknik seperti *Binary Relevance (BR)* yang melatih satu model untuk setiap label secara independen, atau *Label Powerset (LP)* yang memperlakukan kombinasi label sebagai satu kelas baru (Alfaro *et al*., 2023). Sementara itu, *algorithm adaptation* memodifikasi algoritma pembelajaran agar mampu menangani beberapa label sekaligus secara langsung, seperti pada model *Backpropagation Multi-Label Learning (BP-MLL)*.

Dalam menilai performa model *Multi-Label*, diperlukan ukuran evaluasi yang dapat menggambarkan sejauh mana model mampu memprediksi beberapa label dengan benar secara bersamaan. Evaluasi model *Multi-Label* berbeda dari klasifikasi tunggal karena satu instance bisa memiliki lebih dari satu label benar. Beberapa ukuran performa yang umum digunakan antara lain *Hamming Loss*, mengukur proporsi kesalahan label yang diprediksi model, di mana semakin kecil nilainya, semakin baik performa model. *Subset Accuracy*, atau *exact match ratio*) menghitung persentase instance yang seluruh label prediksinya tepat sama dengan label sebenarnya, menjadikannya ukuran yang paling ketat dalam evaluasi.

Sementara itu, *Precision, Recall,* dan *F1-score* merupakan metrik berbasis label yang dihitung untuk setiap label kemudian dirata-ratakan menggunakan dua pendekatan, yaitu macro average dan micro average. *Macro average* memberikan bobot yang sama untuk setiap label dengan menghitung rata-rata dari semua nilai F1, sedangkan *micro average* menghitung F1-score berdasarkan total *true positive* (TP), *false positive* (FP), dan *false negative* (FN) dari seluruh label. (Alfaro *et al*., 2023; Rainio *et al*., 2024).

Evaluasi performa pada klasifikasi *Multi-Label* membutuhkan metrik khusus yang berbeda dari klasifikasi konvensional. Salah satu metrik yang banyak digunakan adalah *Hamming Loss*, yang mengukur rata-rata kesalahan prediksi label pada seluruh instance. Selain itu, *Precision, Recall,* dan *F1-Score* digunakan baik dalam skala macro maupun micro untuk memberikan gambaran menyeluruh terkait kualitas prediksi. *Subset Accuracy* juga digunakan untuk menilai proporsi prediksi yang tepat secara keseluruhan, meskipun metrik ini cenderung sangat ketat karena

mengharuskan semua label benar untuk dianggap akurat (Rizkyani *et al*., 2022). Penggunaan metrik ini secara kombinatif penting agar evaluasi lebih komprehensif dan tidak bias terhadap label mayoritas.

1. *Hamming Loss*

Hamming loss mengukur proporsi kesalahan label yang diprediksi oleh model. Metrik ini mengukur berapa banyak label yang salah diprediksi pada seluruh instance dataset. Semakin kecil nilai *Hamming Loss*, semakin baik performa model. Perhitungan *Hamming loss* ditunjukkan pada Persamaan 2.7:

$$\mathrm{HL} = \frac{1}{\mathrm{N} \times \mathrm{L}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{L}} 1\left[\mathrm{y_{ij}} \neq \hat{\mathrm{y}}_{\mathrm{ij}}\right] \tag{2.7}$$

Keterangan:
HL : nilai *Hamming Loss.*
N : Jumlah Data.
L : Jumlah Label.
y_{ij} : label sebenarnya untuk instance ke- i dan label ke- j.
$\hat{y}_{ij}$: label hasil prediksi untuk instance ke- i dan label ke- j.
$1[.]$: fungsi indikator (1 jika benar, 0 jika salah).

2. *Precision*

Precision mengukur seberapa banyak label yang diprediksi sebagai positif oleh model, ternyata benar-benar positif. Dalam klasifikasi *Multi-Label*, *Precision* dihitung untuk setiap label secara terpisah dan dirata-ratakan menggunakan *macro average*. Perhitungan *precision* per label ditunjukkan pada Persamaan 2.8 dan rata-rata makro pada Persamaan 2.9 berikut:

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2.8}$$

$$Precision_{macro} = \frac{1}{L}\sum_{l=1}^{L} Precision_l \quad (2.9)$$

Keterangan:
TP : *True Positive* (label positif yang terdeteksi benar).
FP : *False Positive* (label negatif yang salah diprediksi positif).

3. *Recall*

Recall mengukur seberapa banyak label positif yang sebenarnya ada yang berhasil ditemukan oleh model. *Recall* dihitung untuk setiap label secara terpisah dan kemudian dirata-ratakan menggunakan *macro average*. Perhitungan *recall* per label ditunjukkan pada Persamaan 2.10 dan rata-rata makro pada Persamaan 2.11:

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad (2.10)$$

$$Recall_{macro} = \frac{1}{L}\sum_{l=1}^{L} Recall_l \quad (2.11)$$

Keterangan:

TP : *True Positive* (label positif yang terdeteksi benar).

FN : *False Negative* (label positif yang gagal terdeteksi).

4. *F1-Score*

F1-Score rata-rata harmonis antara *Precision* dan Recall yang memberikan gambaran yang lebih seimbang mengenai performa model. F1-Score dihitung untuk setiap label, dan hasilnya dirata-ratakan menggunakan *macro average*. Perhitungan F1 per label ditunjukkan pada Persamaan 2.12 dan rata-rata makro pada Persamaan 2.13:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.12}$$

$$F1_{macro} = \frac{1}{L}\sum_{l=1}^{L} F1_l \tag{2.13}$$

Keterangan:
TP : *True Positive* (label positif yang terdeteksi benar).
FN : *False Negative* (label positif yang gagal terdeteksi).

5. *Accuracy*

Accuracy mengukur tingkat keseluruhan prediksi yang benar dari seluruh prediksi yang dibuat oleh model. Perhitungan akurasi ditunjukkan pada Persamaan 2.14:

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{2.14}$$

Keterangan:
TP *(True Positive)* **:** Jumlah data positif yang diprediksi benar.
TN *(True Negative)* **:** Jumlah data negatif yang diprediksi benar.
FP *(False Positive)* **:** Jumlah data negatif yang diprediksi salah sebagai positif.
FN *(False Negative)* **:** Jumlah data positif yang diprediksi salah sebagai negatif.

6. *Subset Accuracy*

Subset Accuracy mengukur proporsi *instance* yang memiliki semua label yang diprediksi dengan tepat, yaitu seluruh label harus benar untuk dianggap akurat. Perhitungannya ditunjukkan pada Persamaan (2.15):

$$Subset\ Accuracy = \frac{1}{N}\sum_{i=1}^{N} 1\left[Y_i = \hat{Y}_i\right] \tag{2.15}$$

Keterangan:
N : Jumlah data (instance).
Y_i : himpunan label sebenarnya untuk instance ke- i.
$\hat{Y}_i$: himpunan label hasil prediksi untuk instance ke- i.

1[.] : fungsi indikator (1 jika benar, 0 jika salah).

Penggunaan kombinasi metrik ini sangat penting agar evaluasi lebih komprehensif. Misalnya, hanya menggunakan akurasi bisa menyesatkan jika dataset tidak seimbang antarlabel, sementara *Hamming loss* dan *F1-Score* dapat memberikan gambaran lebih objektif terhadap performa model.

Berdasarkan beberapa penelitian yang telah dibahas pada bagian sebelumnya, penulis merangkum hasil temuan tersebut dalam bentuk perbandingan yang disajikan pada Tabel 2.2.

Tabel 2.2 Penelitian-Penelitian terkait Klasifikasi *Multi-Label*

No.	Judul Penelitian	Peneliti	Metode yang digunakan	Hasil
1.	*Bidirectional Long Short Term Memory Method and Word2vec Extraction Approach for Hate Speech Detection*	(Isnain *et al*., 2020)	*Bidirectional Long Short-Term Memory (Bi-LSTM), Word2vec*	Model Bi-LSTM dengan fitur *Word2vec* menghasilkan akurasi 90,2%, presisi 88,6%, *recall* 91,4%, dan *f1-score* 89,9%, menunjukkan kemampuan tinggi dalam mendeteksi ujaran kebencian pada teks bahasa Indonesia.
2.	Deteksi Ujaran Kebencian dengan Metode Klasifikasi Naïve Bayes dan Metode N-Gram pada Dataset *Multi-Label*	(Yazid *et al*., 2024)	*Naïve Bayes, N-Gram)*	Pendekatan Naïve Bayes dengan fitur *N-Gram* berhasil mencapai akurasi 86,3%, presisi 84,1%, *recall* 85,7%, dan *f1-score* 85,1%; hasil tersebut menunjukkan efektivitas model dalam mengenali lebih dari satu label ujaran kebencian.
3.	*Multi-Label* Klasifikasi Genre Film Berdasarkan Sinopsis Menggunakan Metode *Long Short-Term Memory* (LSTM)	(J. Akbar *et al*., 2025)	*Long Short-Term Memory (LSTM) dengan Word2vec, GloVe, FastText*	Model LSTM dengan embedding GloVe menunjukkan hasil terbaik dengan f1-score 0,603 dan akurasi 0,171; menunjukkan bahwa *GloVe* lebih efektif dibanding *Word2vec* dan FastText dalam klasifikasi genre *Multi-Label* film.

No.	Judul Penelitian	Peneliti	Metode yang digunakan	Hasil
4.	*Multi-Label* Classification of Indonesian Qur'an Translation Using *Long Short-Term Memory* Model	(I. Akbar *et al*., 2024)	*Long Short-Term Memory (LSTM), Bidirectional LSTM (Bi-LSTM), Word2vec, FastText*	Kombinasi *Bi-LSTM* dengan *FastText* dan penyetelan hiperparameter memberikan akurasi 71,63%, presisi 64,06%, *recall* 63,60%, dan *hamming loss* 36,17%; pendekatan ini meningkatkan akurasi klasifikasi *Multi-Label* terjemahan Al-Qur'an.
5.	Klasifikasi *Multi-Label* Jenis dan Warna Buah Menggunakan Convolutional Neural Network (CNN) dengan Encoder Fitur	(Nida Ulhasanah *et al*., 2025)	*Convolutional Neural Network (CNN) berbasis ResNet-50 dengan cross-validation*	Model CNN dengan *encoder ResNet-50* dan augmentasi data mencapai akurasi 97%, presisi 98,20%, recall 97,61%, dan *f1-score* 0,84; menunjukkan kemampuan tinggi dalam klasifikasi *Multi-Label* jenis dan warna buah.

Pada Tabel 2.2 mengenai beberapa referensi penelitian yang menganalisis penerapan berbagai metode *machine learning* dan *deep learning* dalam klasifikasi *Multi-Label* yang menunjukkan hasil dan pendekatan yang berbeda-beda.

Penelitian yang dilakukan oleh Isnain *et al*. (2020) membahas penerapan algoritma *Bidirectional Long Short-Term Memory (Bi-LSTM)* yang dikombinasikan dengan metode ekstraksi fitur *Word2vec* untuk mengidentifikasi ujaran kebencian pada teks bahasa Indonesia yang memiliki lebih dari satu label. Tujuan dari penelitian ini adalah meningkatkan kemampuan model dalam memahami konteks kalimat dari dua arah agar lebih akurat dalam mendeteksi kategori ujaran kebencian yang kompleks. Hasil pengujian menunjukkan nilai akurasi sebesar 90,2%, presisi 88,6%, recall 91,4%, dan *f1-score* sebesar 89,9%, sehingga dapat disimpulkan bahwa arsitektur Bi-LSTM mampu menangkap

konteks semantik secara mendalam dan efektif dalam kasus klasifikasi *Multi-Label* berbasis teks.

Penelitian yang dilakukan oleh Yazid *et al*. (2024)menggunakan algoritma *Naive Bayes* dengan pendekatan *N-Gram* untuk mendeteksi ujaran kebencian berlabel ganda dalam bahasa Indonesia. Tujuan penelitian ini yaitu mengevaluasi efektivitas model probabilistik sederhana dalam menangani data teks *Multi-Label* dengan fitur yang dihasilkan dari pola urutan kata. Pengujian dilakukan menggunakan dataset dengan pembagian data latih dan uji sebesar 80:20, menghasilkan nilai akurasi 86,3%, presisi 84,1%, recall 85,7%, dan *f1-score* 85,1%. Hasil ini menunjukkan bahwa meskipun tergolong metode klasik, kombinasi antara *Naive Bayes* dan *N-Gram* masih mampu memberikan performa yang baik dalam mengklasifikasikan teks dengan lebih dari satu kategori label.

Selanjutnya, J. Akbar *et al*. (2025) melakukan penelitian dengan menerapkan algoritma *Long Short-Term Memory* (LSTM) untuk klasifikasi *Multi-Label* berbasis teks. Penelitian ini membandingkan tiga jenis *word embedding* yaitu *Word2vec*, *GloVe*, dan *FastText* guna menganalisis pengaruh representasi vektor kata terhadap performa model. Model yang dikembangkan terdiri atas beberapa lapisan seperti *Embedding layer*, *SpatialDropout1D*, *LSTM layer*, dan *Dense layer*. Dari hasil pengujian, *word embedding GloVe* menunjukkan kinerja terbaik dengan nilai *f1-score* sebesar 0,603, sedangkan *Word2vec* memperoleh 0,591 dan *FastText* 0,580. Temuan ini memperlihatkan bahwa *GloVe* lebih efektif dalam menangkap hubungan semantik antar kata sehingga menghasilkan klasifikasi yang lebih akurat pada data berlabel ganda.

Penelitian lain yang dilakukan oleh I. Akbar *et al*. (2024) mengembangkan pendekatan klasifikasi *Multi-Label* berbasis teks dengan menggunakan kombinasi *Long Short-Term Memory* (LSTM) dan *Bidirectional* LSTM (Bi-LSTM) serta metode *word embedding Word2vec* dan *FastText*. Penelitian ini bertujuan untuk meningkatkan akurasi klasifikasi melalui pemilihan model dan penyetelan parameter yang optimal. Dari hasil pengujian, kombinasi Bi-LSTM dengan FastText memberikan performa terbaik dengan nilai akurasi sebesar 71,63%, presisi 64,06%, recall 63,60%, dan *hamming loss* 36,17%. Hasil ini menunjukkan bahwa Bi-LSTM memiliki kemampuan lebih unggul dalam menangani konteks teks yang kompleks dan multi-topik dibandingkan dengan LSTM biasa.

Selanjutnya, penelitian yang dilakukan oleh Nida Ulhasanah *et al*. (2025) menerapkan algoritma *Convolutional Neural Network* (CNN) dengan arsitektur ResNet-50 untuk klasifikasi *Multi-Label* berbasis citra, yang dalam hal ini digunakan untuk mengenali dua atribut utama dalam satu gambar, yaitu jenis dan warna buah. Penelitian ini melibatkan dataset Fruit-360 sebanyak 94.110 gambar dan menggunakan teknik *data augmentation* serta *cross-validation* untuk meningkatkan generalisasi model. Hasil eksperimen menunjukkan akurasi validasi sebesar 97%, presisi 98,20%, recall 97,61%, dan *f1-score* sebesar 0,84. Temuan ini membuktikan bahwa integrasi CNN dengan *encoder* ResNet-50 sangat efektif dalam mendeteksi atribut ganda pada data visual secara simultan dan akurat.

BAB III

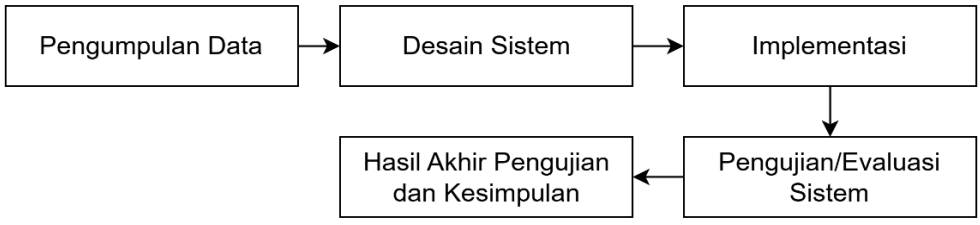
METODE PENELITIAN

Bab ini menjelaskan metode yang digunakan dalam pelaksanaan penelitian. Perancangan metode penelitian dilakukan agar proses penyusunan skripsi berjalan dengan lebih terarah, efisien, serta memastikan setiap tahapan penelitian dilaksanakan secara sistematis dan terstruktur.

3.1 Prosedur Penelitian

Prosedur penelitian dapat diartikan sebagai serangkaian langkah yang dilakukan oleh peneliti untuk menyusun dan melakukan penelitian. Pada penelitian ini, prosedur penelitian mencakup beberapa tahapan yang saling berkaitan, dimulai dari pengumpulan data hingga evaluasi hasil akhir.

Gambaran mengenai alur prosedur penelitian ditunjukkan pada Gambar 3.1 untuk memastikan bahwa seluruh tahapan penelitian berjalan secara terstruktur dan terarah.



Gambar 3.1 Alur Prosedur Penelitian

Diagram pada Gambar 3.1 menunjukkan bahwa penelitian ini terdiri atas lima tahapan utama, yaitu pengumpulan data, perancangan sistem, implementasi, pengujian atau evaluasi sistem, serta penarikan kesimpulan.

Penelitian dimulai dari pengumpulan data, yaitu proses memperoleh data yang akan digunakan dalam membangun dan melatih model. Data yang dikumpulkan merupakan data sekunder berupa *tweet* berbahasa Indonesia yang telah memiliki label kategori ujaran kebencian. Data ini dipilih karena relevan dengan tujuan penelitian yang berfokus pada klasifikasi *Multi-Label* ujaran kebencian. Setelah dikumpulkan, data melalui tahapan pra-pemrosesan untuk memastikan kualitas dan konsistensi sebelum digunakan pada tahap perancangan sistem.

Setelah data diperoleh, penelitian berlanjut pada desain sistem, yaitu perancangan alur kerja dan arsitektur model yang akan digunakan dalam proses klasifikasi. Desain sistem berfungsi untuk menggambarkan hubungan antara proses *preprocessing* data, pembentukan representasi kata menggunakan *Word2vec*, serta rancangan arsitektur jaringan LSTM sebagai inti model pembelajaran. Pada tahap ini ditentukan parameter utama model, seperti dimensi *embedding*, jumlah unit LSTM, fungsi aktivasi, dan *loss function* yang akan digunakan. Rancangan sistem yang dihasilkan menjadi dasar dalam tahap implementasi berikutnya.

Tahapan berikutnya adalah implementasi, yaitu proses penerapan rancangan sistem ke dalam bentuk program yang dapat dijalankan. Implementasi dilakukan menggunakan bahasa pemrograman Python dengan pustaka *deep learning* seperti *TensorFlow* dan Keras. Model LSTM yang telah dirancang dilatih menggunakan data latih hasil *preprocessing* dan representasi *Word2vec*. Selain itu, dilakukan penyesuaian *hyperparameter* seperti jumlah unit LSTM, *batch size*, dan *threshold,*

dan Dimensi *Embedding* untuk mendapatkan performa terbaik. Proses ini menghasilkan model klasifikasi yang siap diuji menggunakan data uji.

Setelah implementasi selesai, dilakukan pengujian dan evaluasi sistem untuk menilai kinerja model yang telah dibangun. Tahapan ini bertujuan mengetahui sejauh mana model mampu mengenali dan mengklasifikasikan ujaran kebencian dengan lebih dari satu label. Pengujian dilakukan menggunakan data uji yang belum pernah digunakan sebelumnya agar hasilnya objektif. Evaluasi dilakukan dengan beberapa metrik, yaitu *precision*, *recall*, *F1-score*, *Hamming Loss*, dan *Subset Accuracy*. Metrik-metrik ini digunakan untuk mengukur tingkat akurasi, sensitivitas, serta kemampuan model dalam menggeneralisasi data baru.

Bagian terakhir dari proses penelitian adalah hasil akhir pengujian dan kesimpulan. Tahapan ini mencakup analisis terhadap hasil evaluasi model untuk menjawab rumusan masalah dan tujuan penelitian. Hasil pengujian dianalisis secara kuantitatif dan kualitatif guna melihat kelebihan serta keterbatasan model yang digunakan. Berdasarkan hasil analisis tersebut kemudian disusun kesimpulan yang menggambarkan efektivitas model dalam mendeteksi ujaran kebencian berbahasa Indonesia. Selain itu, bagian ini juga memuat rekomendasi untuk penelitian selanjutnya agar sistem yang dikembangkan dapat ditingkatkan baik dari segi metode, data, maupun cakupan penerapannya.

Secara keseluruhan, prosedur penelitian ini menggambarkan proses yang terstruktur dan berkesinambungan mulai dari pengumpulan data hingga penarikan kesimpulan. Setiap komponen dalam diagram saling melengkapi dan berperan penting dalam menghasilkan model klasifikasi *Multi-Label* yang mampu

mendeteksi ujaran kebencian secara akurat, efisien, dan dapat diterapkan dalam konteks media sosial berbahasa Indonesia.

3.2 Pengumpulan Data

Pengumpulan data merupakan tahapan yang dilakukan untuk memperoleh informasi yang relevan serta diperlukan dalam suatu penelitian. Data pada penelitian ini terbagi menjadi dua jenis, yaitu data primer dan data sekunder.

Data primer adalah data yang didapat langsung dari sumber asli melalui metode seperti wawancara, eksperimen, atau observasi. Akan tetapi, dalam penelitian ini digunakan data sekunder yang diunduh dari platform Kaggle. Data sekunder tersebut berupa dataset yang telah disiapkan oleh pihak ketiga yang sesuai dengan topik penelitian.

Penelitian ini menggunakan dua dataset, yaitu *Indonesian Abusive and Hate Speech Twitter Text dan Indonesian Stoplist*. *Dataset Indonesian Abusive and Hate Speech Twitter Text* berisi kumpulan *tweet* berbahasa Indonesia yang telah diberi label untuk menunjukkan apakah suatu teks termasuk ujaran kebencian (*hate speech*) atau tidak. Jumlah data yang tersedia sebanyak 13.169 *tweet*, dengan distribusi 5.561 data dikategorikan sebagai *hate speech* (42,2%) dan 7.608 data dikategorikan sebagai *non-hate speech* (57,8%). Dataset ini digunakan sebagai sumber utama untuk proses klasifikasi *Multi-Label* ujaran kebencian yang menjadi fokus penelitian.

Sementara itu, dataset *Indonesian Stoplist* berisi daftar kata umum dalam bahasa Indonesia (*stopwords*) yang tidak memiliki makna penting dalam proses analisis teks. *Dataset* ini digunakan pada tahap *preprocessing* untuk menghapus

kata-kata umum seperti “dan”, “yang”, atau “di”, sehingga hasil analisis menjadi lebih akurat. Dengan menggunakan daftar *stopwords* ini, model dapat lebih fokus mempelajari kata-kata yang memiliki makna signifikan terhadap klasifikasi ujaran kebencian.

Detail atribut yang disertakan dalam dataset ini (*Indonesian Abusive and Hate Speech Twitter Text*) ditampilkan pada Tabel 3.1.

Tabel 3.1 Atribut Dataset

Kolom	Deskripsi	Tipe Data
Tweet	Teks *tweet* asli yang ditulis oleh pengguna.	*String*
HS	Ujaran kebencian (1: *hate speech*, 0: non-*hate speech*).	*Integer*
HS_Individual	Ujaran kebencian yang ditujukan kepada individu atau seseorang.	*Integer*
HS_Group	Ujaran kebencian yang ditujukan kepada suatu kelompok.	*Integer*
HS_Religion	Ujaran kebencian terkait agama atau kepercayaan tertentu.	*Integer*
HS_Race	Ujaran kebencian terkait ras atau suku atau etnis.	*Integer*
HS_Other	Ujaran kebencian terkait dengan makian atau fitnah lainnya.	*Integer*

Setelah memahami atribut-atribut pada Tabel 3.1, diperlukan contoh penyajian data untuk memperlihatkan bagaimana *tweet* dan label *Multi-Label* direpresentasikan dalam dataset. Oleh karena itu, Gambar 3.2 menampilkan contoh potongan data yang digunakan sebagai dasar dalam proses pemodelan.

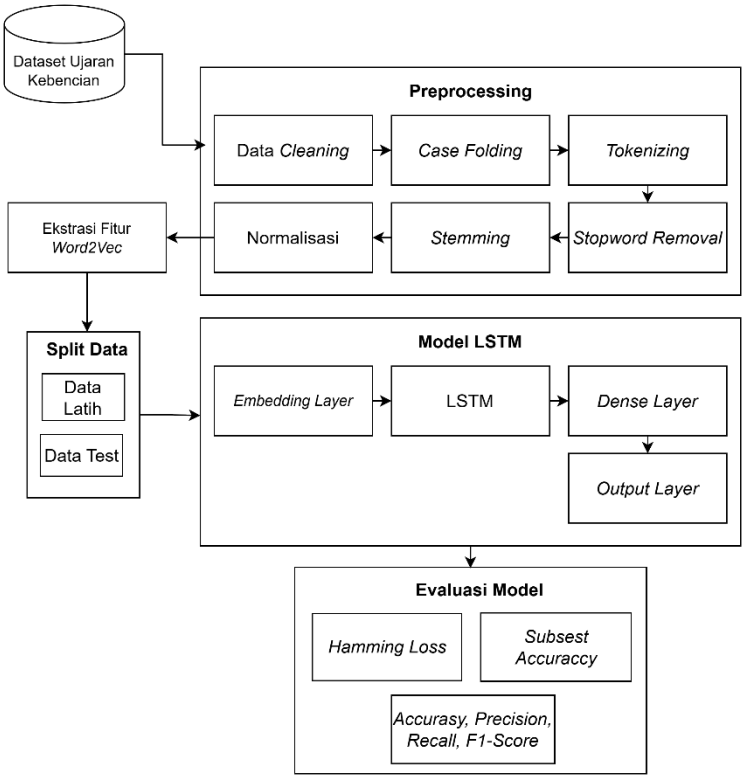
	Tweet	HS	HS_Individual	HS_Group	HS_Religion	HS_Race	HS_Other
0	- disaat semua cowok berusaha melacak perhatian gue. loe lantas remehkan perhatian yg gue kasih khusus ke elo. basic elo cowok bego ! ! !'	1	1	0	0	0	1
1	RT USER: USER siapa yang telat ngasih tau elu?edan sarap gue bergaul dengan cigax jifla calis sama siapa noh licew juga'	0	0	0	0	0	0
2	41. Kadang aku berfikir, kenapa aku tetap percaya pada Tuhan padahal aku selalu jatuh berkali-kali. Kadang aku merasa Tuhan itu ninggalkan aku sendirian. Ketika orangtuaku berencana berpisah, ketika kakakku lebih memilih jadi Kristen. Ketika aku anak ter	0	0	0	0	0	0
3	USER USER AKU ITU AKU\n\nKU TAU MATAMU SIPIT TAPI DILIAT DARI MANA ITU AKU'	0	0	0	0	0	0
4	USER USER Kaum cebong kapir udah keliatan dongoknya dari awal tambah dongok lagi hahahah'	1	0	1	1	0	0

Gambar 3.2 Lima teratas *tweet* dataset dan Label *Multi-Label*

Pada Gambar 3.2 dapat dilihat bahwa dataset penelitian ini terdiri atas satu kolom *tweet* sebagai teks utama yang menjadi objek analisis, serta enam kolom label biner yang menunjukkan kategori ujaran kebencian. Setiap kolom label bernilai 1 apabila kategori tersebut muncul pada *tweet* yang bersangkutan dan bernilai 0 apabila kategori tersebut tidak muncul. Struktur ini menunjukkan bahwa dataset bersifat *Multi-Label*, yang berarti satu *tweet* dapat mengandung lebih dari satu kategori ujaran kebencian secara bersamaan. Gambar ini memperlihatkan bentuk data mentah yang menjadi dasar proses *preprocessing* dan pemodelan LSTM pada tahap selanjutnya.

3.3 Desain Sistem

Desain sistem pada penelitian ini menggambarkan alur proses kerja model deteksi ujaran kebencian berbahasa Indonesia secara *Multi-Label* menggunakan algoritma LSTM. Setiap tahapan dalam sistem saling terhubung, mulai dari pengumpulan dan pengolahan data hingga proses pelatihan dan evaluasi model. Gambaran keseluruhan desain sistem ditunjukkan pada Gambar 3.3.



Gambar 3.3 Desain Sistem

Pada penelitian ini, proses sistem dimulai dari penggunaan dataset ujaran kebencian berbahasa Indonesia yang berisi kumpulan *tweet* yang telah diberi label ke dalam beberapa kategori ujaran kebencian yang mencakup ujaran kebencian umum, individu, kelompok, agama, ras, serta makian atau fitnah lainnya. Dataset ini menjadi sumber utama dalam proses pembangunan sistem klasifikasi *Multi-Label*. Sebelum digunakan, data melalui tahapan *preprocessing* untuk memastikan teks berada dalam kondisi bersih dan siap diolah oleh model.

Proses *preprocessing* meliputi pembersihan karakter khusus (*data cleaning*), pengubahan huruf menjadi huruf kecil (*case folding*), pemisahan kata (*tokenization*), penghapusan kata umum yang tidak memiliki makna penting (*stopword removal*), serta pengembalian kata ke bentuk dasarnya (*stemming*).

Tahapan-tahapan tersebut dilakukan agar model dapat memahami isi teks secara lebih efektif dan konsisten.

Setelah teks selesai diproses, dilakukan pembentukan representasi kata menggunakan *Word2Vec*. Tahapan ini bertujuan untuk mengubah kata-kata dalam teks menjadi vektor numerik yang merepresentasikan hubungan semantik antar kata. Dengan pendekatan ini, kata-kata yang sering muncul dalam konteks serupa akan memiliki representasi vektor yang berdekatan, sehingga membantu model memahami makna ujaran kebencian secara lebih kontekstual. Hasil dari proses ini selanjutnya digunakan sebagai masukan bagi model pembelajaran.

Data yang telah direpresentasikan dalam bentuk vektor selanjutnya digunakan untuk melatih model LSTM yang dirancang untuk menghasilkan prediksi *Multi-Label* pada enam kategori ujaran kebencian. Model ini dipilih karena kemampuannya dalam memahami urutan kata dan menangkap hubungan jangka panjang antar kata pada kalimat. Pada proses pelatihan, data dibagi menjadi dua bagian, yaitu data latih (*training data*) sebesar 80% dan data uji (*testing data*) sebesar 20%. Pembagian ini bertujuan agar model dapat belajar dari sebagian besar data dan diuji pada data baru untuk menilai kemampuan generalisasinya. Selain itu, dilakukan penyesuaian terhadap beberapa parameter seperti seperti jumlah unit LSTM, *batch size*, *threshold*, dan dimensi *embedding* untuk mendapatkan performa terbaik.

Tahap berikutnya adalah evaluasi model, yaitu proses pengujian hasil prediksi model terhadap data uji untuk menilai performa sistem dalam melakukan klasifikasi *Multi-Label*. Evaluasi dilakukan dengan menghitung beberapa metrik

seperti *precision*, *recall*, *F1-score*, *Hamming Loss*, dan *Subset Accuracy*. Nilai dari metrik-metrik tersebut digunakan untuk mengetahui tingkat akurasi model dalam mengenali lebih dari satu kategori ujaran kebencian dari enam label yang digunakan dalam penelitian ini.

Tahap evaluasi menjadi proses terakhir dalam sistem yang dirancang. Hasil dari tahap ini dijadikan dasar untuk melakukan analisis lebih lanjut mengenai efektivitas model dalam mendeteksi ujaran kebencian berbahasa Indonesia. Analisis hasil evaluasi tersebut juga digunakan sebagai acuan dalam menarik kesimpulan dan memberikan rekomendasi pengembangan penelitian di masa mendatang.

3.4 *Preprocessing*

Preprocessing adalah langkah awal yang penting dalam analisis teks untuk menyiapkan data agar dapat digunakan oleh algoritma LSTM. Proses ini mencakup berbagai tahapan yang bertujuan untuk membersihkan dan memformat data teks agar lebih mudah dianalisis.

3.4.1 Data *Cleaning*

Tahap *data cleaning* bertujuan untuk membersihkan teks dari komponen yang tidak memiliki informasi penting agar tidak menimbulkan *noise* pada proses ekstraksi fitur. Proses pembersihan dilakukan dengan menggunakan *regular expressions* (*regex*) untuk menghapus tautan (URL), *mention* (@), *hashtag* (#) yang tidak relevan, angka, tanda baca berlebihan, serta untuk merapikan spasi ganda dan

karakter berulang. Sebagai contoh, kata “gemeeeees” disederhanakan menjadi “gemes”.

Secara khusus, untuk konteks deteksi ujaran kebencian, bentuk penyamaran kata hinaan (seperti “t0lol”, “kntl”) tidak dihapus pada tahap ini. Karakter penyamaran tetap dipertahankan agar makna ofensifnya dapat dipulihkan secara terarah pada tahap normalisasi. Contoh hasil penerapan tahap *data cleaning* pada data teks ditunjukkan pada Tabel 3.2 berikut.

Tabel 3.2 Proses Data *Cleaning*

Sebelum Data *Cleaning*	**Setelah Data *Cleaning***
Asw ya tapi gua jarang ngambek, tacut wkkwkwkwkw gua kan bucin.	Asw ya tapi gua jarang ngambek tacut wkwk gua kan bucin

3.4.2 *Case Folding*

Tahap *case folding* bertujuan mengubah seluruh huruf kapital menjadi huruf kecil agar semua kata berada dalam format yang seragam. Tahapan ini memastikan bahwa kata yang sama, meskipun memiliki perbedaan kapitalisasi, seperti “Benci” dan “benci”, dianggap sebagai satu kesatuan. Dengan demikian, variasi penulisan yang tidak perlu dapat dihindari tanpa mengubah makna kata. Contoh hasil penerapan tahap *case folding* pada data teks ditampilkan pada Tabel 3.3 berikut.

Tabel 3.3 Proses *Case Folding*

Sebelum *Case Folding*	**Sesudah *Case Folding***
Asw ya tapi gua jarang ngambek, tacut wkkwkwkwkw gua kan bucin.	asw ya tapi gua jarang ngambek tacut wkwk gua kan bucin.

3.4.3 *Tokenizing*

Setelah tahap pembersihan teks, dilakukan proses tokenisasi, yaitu pemecahan teks menjadi token atau potongan kata berdasarkan spasi dan tanda

baca. Tahapan ini memungkinkan teks diproses sebagai deret elemen terpisah sehingga setiap kata dapat dianalisis dan diberi bobot secara individual. Contoh hasil penerapan tahap tokenisasi pada data teks ditunjukkan Tabel 3.4 berikut.

Tabel 3.4 Proses *Tokenizing*

Sebelum *Tokenizing*	**Sesudah *Tokenizing***
Asw ya tapi gua jarang ngambek, tacut wkkwkwkwkw gua kan bucin.	“asw”, “ya”, “tapi”, “gua”, “jarang”, “ngambek”, “tacut”, “wkwk”, “gua”, “kan”, “bucin”.

3.4.4 *Stopword Removal*

Tahap *stopword removal* bertujuan untuk menghilangkan kata-kata fungsi yang umumnya tidak memiliki bobot makna topikal, seperti “yang”, “dan”, atau “di”. Proses ini mengacu pada *Indonesian Stoplist* yang diperoleh dari Kaggle. Namun, karena ujaran kebencian sering berkaitan dengan bentuk negasi, beberapa kata seperti “tidak/tdk/tak”, “bukan”, “jangan”, “nggak/ga/gak”, “tanpa”, dan “belum” tetap dipertahankan melalui mekanisme *whitelisting*.

Pendekatan serupa juga diterapkan pada nama kelompok, entitas tertentu, serta token yang merepresentasikan makna ofensif. Strategi ini bertujuan menyederhanakan dimensi fitur tanpa menghilangkan informasi penting yang relevan dengan proses klasifikasi. Contoh hasil penerapan tahap *stopword removal* pada data teks ditunjukkan pada Tabel 3.5 berikut.

Tabel 3.5 Proses *Stopword Removal*

Sebelum *Stopword Removal*	**Sesudah *Stopword Removal***
Asw ya tapi gua jarang ngambek, tacut wkkwkwkwkw gua kan bucin.	“asw”, “gua”, “jarang”, “ngambek”, “tacut”, “wkwk”, “gua”, “bucin”

3.4.5 *Stemming*

Tahap *stemming* bertujuan mengembalikan kata ke bentuk dasarnya untuk menyatukan variasi kata yang timbul akibat penggunaan imbuhan, baik awalan, akhiran, maupun konfiks. Pada penelitian ini, proses *stemming* dilakukan menggunakan algoritma Sastrawi. Dengan pendekatan ini, kata-kata seperti "menghina", "penghinaan", dan "dihina" dikonversi ke bentuk dasar yang sama.

Pendekatan *stemming* yang digunakan bersifat konservatif, dengan menjaga agar nama diri, istilah serapan, atau akronim tertentu, seperti "Jawa Barat", tidak mengalami perubahan yang dapat merusak makna aslinya. Contoh hasil penerapan tahap *stemming* pada data teks ditunjukkan pada Tabel 3.6 berikut.

Tabel 3.6 Proses *Stemming*

Sebelum *Stemming*	**Sesudah *Stemming***
Asw ya tapi gua jarang ngambek, tacut wkkwkwkwkw gua kan bucin.	"asw", "gua", "jarang", "ambek", "takut", "wkkwkwkwkw", "gua", "bucin"

3.4.6 Normalisasi

Tahap normalisasi bertujuan mengubah kata tidak baku, *slang*, maupun bentuk kamuflase menjadi bentuk standar agar proses perhitungan fitur, seperti TF–IDF, berlangsung secara konsisten. Sebagai contoh, "sdh" dinormalisasi menjadi "sudah", sedangkan "jabar" diubah menjadi "jawa_barat", dengan penggunaan *underscore* agar dianggap sebagai satu token.

Selain itu, *emoji* dan *emotikon* dipetakan ke dalam token kata yang merepresentasikan makna pragmatisnya, seperti ":)" menjadi "senyum", "😡" menjadi "marah", dan "😂" menjadi "tertawa". Proses pemetaan *slang* dan *emoji* dikelola melalui berkas kamus khusus, seperti *slang_map.csv* dan *emoji_map.csv*,

agar proses normalisasi dapat direplikasi secara konsisten. Contoh hasil penerapan tahap normalisasi pada data teks ditunjukkan pada Tabel 3.7.

Tabel 3.7 Proses Normalisasi

Sebelum Normalisasi	Sesudah Normalisasi
Asw ya tapi gua jarang ngambek, tacut wkkwkwkwkw gua kan bucin.	"anjing", "saya", "jarang", "ambek", "takut", "haha", "saya", "budak_cinta"

3.5 *Word2vec*

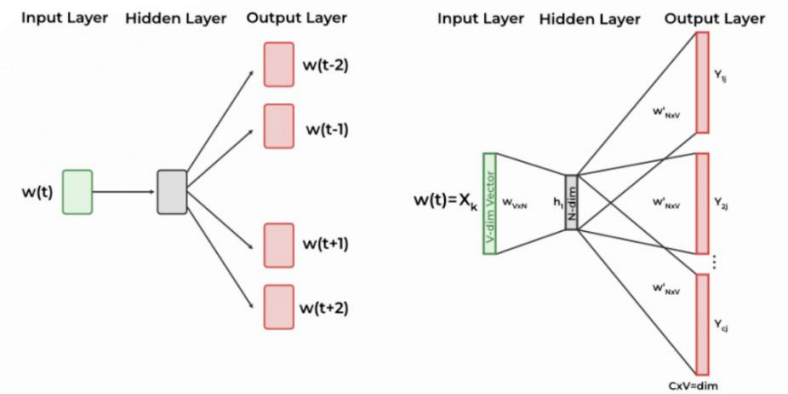
Pada tahap ini dilakukan ekstraksi fitur teks menggunakan metode *Word2vec*, yaitu salah satu teknik representasi kata yang banyak digunakan dalam bidang NLP Metode ini mengubah kata-kata dalam korpus menjadi vektor numerik sehingga hubungan semantik antar kata dapat dianalisis secara matematis (Mikolov *et al*., 2013).

Melalui representasi vektor ini, setiap kata tidak hanya direpresentasikan secara tunggal, tetapi juga berdasarkan konteks kemunculannya dalam kalimat. Akibatnya, kata-kata yang sering muncul dalam konteks serupa akan berada pada posisi vektor yang berdekatan, menandakan kemiripan makna di ruang representasi.

Word2vec memiliki dua pendekatan utama, yaitu *Continuous Bag of Words* (CBOW) dan *Skip-Gram*. Keduanya dirancang untuk mempelajari hubungan antar kata melalui pola kemunculannya dalam teks. Dalam penelitian ini digunakan pendekatan *Skip-Gram* karena diketahui lebih efektif dalam mempelajari kata-kata yang jarang muncul *(rare words)* dan mampu memprediksi konteks di sekitar kata target dengan lebih akurat (Rong, 2016).

Secara sederhana, model *Skip-Gram* bekerja dengan memilih sebuah kata sebagai pusat *(target word)*, kemudian berusaha memprediksi kata-kata di sekitarnya (kata konteks) dalam suatu jendela konteks *(window size)*. Melalui

proses pembelajaran tersebut, model dapat menangkap keterkaitan makna antar kata, sehingga kata yang sering muncul bersama akan memiliki vektor representasi yang saling berdekatan di ruang vektor.



Gambar 3.4 Arsitektur *Word2vec Skip-Gram*

Secara umum, arsitektur *Word2vec Skip-Gram* terdiri atas tiga lapisan berikut:

1. Lapisan Input *(Input layer)*

   Lapisan ini menerima satu kata target dalam bentuk vektor *one-hot* berdimensi $|V|$, di mana $|V|$ adalah ukuran kosakata (*vocabulary size*).

2. Lapisan Tersembunyi (*Hidden layer*)

   Hidden layer terdiri atas dua matriks bobot utama, yaitu:

   a. Matriks bobot $W \in \mathbb{R}^{|V| \times N}$, untuk memetakan vektor *one-hot* menjadi vektor embedding berdimensi N.

   b. Matriks bobot $U \in \mathbb{R}^{N \times |V|}$, untuk memproyeksikan kembali hasil embedding ke ruang kosakata.

   Tidak terdapat fungsi aktivasi pada ini sehingga hasilnya berupa representasi linear dari kata target.

Keterangan:

W	: matriks bobot input berukuran $\mid V \mid \times N$.
U	: matriks bobot keluaran berukuran $N \times \mid V \mid$.
$\mid V \mid$	: ukuran kosakata (*vocabulary size*).
N	: Dimensi embedding (*embedding dimension*).
$x \in \mathbb{R}^{\mid V \mid}$	: vektor one-hot yang mewakili kata target.
$h \in \mathbb{R}^{N}$	: vektor representasi tersembunyi setelah dikalikan dengan W.
$z \in \mathbb{R}^{\mid V \mid}$	: vektor skor keluaran (output scores) sebelum normalisasi probabilitas.

3. Lapisan Keluaran (*Output layer*)

   Lapisan keluaran menghitung distribusi probabilitas untuk semua kata dalam kosakata menggunakan fungsi aktivasi *softmax*. Nilai keluaran menunjukkan peluang setiap kata menjadi kata konteks dari kata target yang sedang dipelajari.

Tahap pelatihan *Word2vec Skip-Gram* dilakukan melalui beberapa langkah sebagai berikut:

1. Pembentukan Pasangan Kata Target dan Konteks

   Setiap kata dalam korpus dipasangkan dengan kata-kata di sekitarnya berdasarkan ukuran jendela (window size). Contoh, untuk kalimat “kamu cowok bodoh”, dengan *window* = 2, pasangan target-konteks yang terbentuk adalah:

   - (cowok → kamu),
   - (cowok → bodoh),
   - (kamu → cowok),
   - (bodoh → cowok).

   Pasangan-pasangan ini digunakan untuk melatih model agar mengenali hubungan antar kata yang sering muncul bersama dalam konteks yang sama.

2. Representasi *One-Hot*

Setiap kata diubah menjadi vektor *one-hot* sepanjang ukuran kosakata $| V |$. Sebagai contoh, jika kosakata terdiri dari tiga kata {kamu, cowok, bodoh}, maka representasinya dapat digambarkan sebagai berikut:

- kamu = [1, 0, 0]
- cowok = [0, 1, 0]
- bodoh = [0, 0, 1]

Vektor ini menjadi masukan bagi model pada lapisan pertama.

Sebagai ilustrasi, kata "cowok" digunakan sebagai kata target dengan vektor input sebagai berikut:

$$x = [0, 1, 0]^T$$

3. Proses *Feedforward*

Misalkan bobot awal diasumsikan sebagai berikut (untuk contoh sederhana):

$$W = \begin{bmatrix} 0.10 & 0.20 \\ 0.00 & 0.15 \\ 0.05 & 0.10 \end{bmatrix}, U = \begin{bmatrix} 0.10 & 0.25 \\ 0.20 & 0.05 \\ 0.15 & 0.10 \end{bmatrix}$$

Di mana, baris-baris W berukuran $| V | \times N = 3 \times 2$ dan mewakili vektor embedding awal setiap kata dalam kosakata. Matriks U digunakan pada lapisan keluaran untuk menghitung skor dari hasil embedding. Selanjutnya, vektor input dikalikan dengan matriks bobot W untuk menghasilkan representasi tersembunyi. Proyeksi ke ruang *embedding* ditunjukkan pada Persamaan 3.1:

$$h = W^T \cdot x \tag{3.1}$$

Persamaan 3.1 digunakan untuk memproyeksikan kata target dari ruang *one-hot* ke ruang embedding berdimensi N. Nilai h merupakan representasi semantik kata target.

Karena $x = [0, 1, 0]^T$, maka:

$$h = \begin{bmatrix} 0.10 & 0.00 & 0.05 \\ 0.20 & 0.15 & 0.10 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.00 \\ 0.15 \end{bmatrix}$$

Jadi, representasi tersembunyi untuk kata “cowok” adalah:

$$h = [0.00, \quad 0.15]^T$$

Hasil vektor tersembunyi (h) kemudian dikalikan dengan matriks bobot U untuk menghasilkan skor untuk seluruh kata dalam kosakata. Skor keluaran untuk seluruh kata dihitung sesuai Persamaan 3.2:

$$\mathrm{z} = U^T \cdot h \tag{3. 2}$$

Keterangan:

h	: vektor representasi tersembunyi (hidden representation) dari kata target setelah dikalikan dengan bobot input.
W	: matriks bobot input berukuran $\mid V \mid \times N$.
x	: vektor *one-hot encoding* yang mewakili kata target, berdimensi $\mid V \mid$.
z	: vektor skor keluaran (*output scores*) sebelum dilakukan normalisasi probabilitas.
U	: matriks bobot keluaran berukuran $N \times \mid V \mid$.

Persamaan 3.2 ini digunakan untuk menghitung nilai skor (logit) awal untuk setiap kata dalam kosakata. Nilai z menggambarkan skor mentah (*unnormalized scores*) yang menunjukkan tingkat kemungkinan setiap kata menjadi konteks dari kata target sebelum dilakukan normalisasi probabilitas.

$$z = \begin{bmatrix} 0.10 & 0.25 \\ 0.20 & 0.05 \\ 0.15 & 0.10 \end{bmatrix} \cdot \begin{bmatrix} 0.00 \\ 0.15 \end{bmatrix}$$

$$z = \begin{bmatrix} (0.10) & (0.00) + (0.25) & (0.15) \\ (0.20) & (0.00) + (0.05) & (0.15) \\ (0.15) & (0.00) + (0.10) & (0.15) \end{bmatrix} = \begin{bmatrix} 0.0375 \\ 0.0075 \\ 0.015 \end{bmatrix}$$

Dengan demikian diperoleh skor awal untuk setiap kata dalam kosakata yang disajikan dalam Tabel 3.8.

Tabel 3.8 Skor awal tiap kata

Kata	Nilai z
kamu	0.0375
cowok	0.0075
bodoh	0.015

Skor ini menunjukkan seberapa besar kemungkinan setiap kata menjadi kata konteks dari kata target yang sedang diproses.

4. Perhitungan Probabilitas Konteks

Nilai z yang diperoleh dari Persamaan 3.2 kemudian diubah menjadi distribusi probabilitas dengan menggunakan fungsi aktivasi *softmax*. Fungsi ini bertujuan untuk menormalkan skor linear z_i agar menjadi nilai probabilitas dengan total keseluruhan sama dengan 1. Normalisasi skor menggunakan *softmax* ditunjukkan pada Persamaan (3.3).

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{|V|} e^{z_j}} \tag{3. 3}$$

Keterangan:
y_i : probabilitas kata ke-i sebagai konteks.
z_i : skor linear (*logit*) untuk kata ke-i sebelum normalisasi.
$|V|$: jumlah kata dalam kosakata (*vocabulary size*).
e : konstanta bilangan natural (≈ 2.71828).

Persamaan 3.3 digunakan untuk menghitung peluang setiap kata dalam kosakata menjadi kata konteks dari kata target. Semakin besar nilai y_i, semakin tinggi kemungkinan kata tersebut muncul di sekitar kata target pada korpus pelatihan.

Untuk memudahkan penulisan, proses *softmax* seringkali dinyatakan kembali dalam bentuk fungsi vektor seperti pada Persamaan 3.4 berikut:

$$\mathrm{y} = softmax(z) \tag{3. 4}$$

Keterangan:
y : vektor probabilitas atas seluruh kata dalam kosakata.
softmax(·) : fungsi aktivasi yang menormalkan skor linear menjadi distribusi probabilitas.
z : vektor skor linear hasil keluaran dari lapisan sebelumnya.

Persamaan 3.4 merupakan bentuk ringkas dari Persamaan 3.3. Dengan notasi fungsi ini, *softmax* dapat dipahami sebagai operasi yang menerima vektor skor z dan menghasilkan distribusi probabilitas y dengan total nilai 1.

Nilai z diubah menjadi probabilitas kemunculan kata konteks dengan menggunakan fungsi *softmax* sebagaimana dijelaskan pada Persamaan 3.4:

$$e^{z_{kamu}} = e^{0.0375} = 1.0382,$$

$$e^{z_{cowok}} = e^{0.0075} = 1.0075,$$

$$e^{z_{bodoh}} = e^{0.0150} = 1.0151$$

$$\sum e^{z_j} = 1.0382 + 1.0075 + 1.0151 = 3.0608$$

Maka,

$$y_{kamu} = \frac{1.0382}{3.0608} = 0.3392,$$

$$y_{cowok} = \frac{1.0075}{3.0608} = 0.3291,$$

$$y_{bodoh} = \frac{1.0151}{3.0608} = 0.3317$$

Sehingga dihasilkan distribusi probabilitas konteks yang disajikan dalam Tabel 3.9.

Tabel 3.9 Distribusi probabilitas konteks

Kata	Probabilitas (y)
kamu	0.3392
cowok	0.3291
bodoh	0.3317

Dari hasil distribusi probabilitas pada tabel 3.9 di atas menunjukkan bahwa kata "kamu" memiliki peluang tertinggi sebagai konteks dari kata target "cowok". Nilai-nilai ini akan terus diperbarui selama proses pelatihan hingga model mencapai konvergensi.

5. Perhitungan Kesalahan dan Pembaruan Bobot

Setelah probabilitas konteks diperoleh, langkah berikutnya adalah menghitung error antara prediksi y dan kata konteks sebenarnya t. Fungsi kerugian yang digunakan adalah *negative log-likelihood loss* (sering juga disebut *cross-entropy loss*), dan bobot diperbarui menggunakan algoritma *Stochastic Gradient Descent* (SGD). Aturan pembaruan bobot ditunjukkan pada Persamaan 3.5 dan Persamaan 3.6.

$$W_{baru} = W_{lama} - \eta \frac{\partial L}{\partial W} \quad (3.5)$$

$$U_{baru} = U_{lama} - \eta \frac{\partial L}{\partial U} \quad (3.6)$$

Keterangan:
W_{baru}, U_{baru} : nilai pembaruan matriks bobot setelah satu iterasi pelatihan.
W_{lama}, U_{lama} : nilai bobot sebelum pembaruan.
$\frac{\partial L}{\partial W}$ dan $\frac{\partial L}{\partial U}$: gradien fungsi kerugian terhadap bobot masing-masing, digunakan untuk menyesuaikan bobot agar nilai *L* berkurang.

Persamaan 3.5 dan 3.6 ini digunakan untuk memperbarui parameter model agar prediksi semakin mendekati data sebenarnya. Dengan η merupakan *learning rate* dan *L* adalah fungsi *loss* model.

Turunan gradien pada kedua bobot diperoleh untuk menghitung arah perubahan bobot yang menurunkan error secara bertahap (belajar melalui propagasi balik). Gradien dihitung sesuai Persamaan 3.7 dan Persamaan 3.8.

$$\frac{\partial L}{\partial U} = (y - t) \cdot h^T \quad (3.7)$$

$$\frac{\partial L}{\partial W} = x \cdot U^T (y - t) \quad (3.8)$$

Keterangan:
t : vektor *one-hot* yang menunjukkan kata konteks sebenarnya,
y : vektor probabilitas hasil *softmax,*
$h = W^T x$: vektor representasi tersembunyi dari kata target.
x : vektor *one-hot* kata target.
U : matriks bobot keluaran.

Langkah Pembaruan:

1. Model menghitung probabilitas setiap kata konteks melalui *softmax*
2. Dibandingkan dengan konteks sebenarnya (label t), diperoleh error $(y - t)$.
3. Nilai error ini dikalikan dengan representasi tersembunyi h untuk memperbarui bobot keluaran U.

4. Kemudian, error dipropagasikan ke lapisan input untuk memperbarui bobot W.
5. Pembaruan dilakukan berulang kali untuk semua pasangan target–konteks hingga nilai loss stabil (konvergen).

Sebagai ilustrasi sederhana, diasumsikan parameter berikut:

$$\eta = 0.05,\ t = [1, 0, 0]^T,\ y = [0.3392, 0.3291, 0.3317]^T$$

Maka,

$$(y - t) = [-0.6608,\quad 0.3291,\quad 0.3317]^T$$

Jika $h = [0.00,\quad 0.15]^T$, maka gradien untuk matriks U dihitung sebagai:

$$\frac{\partial L}{\partial U} = (y - t) \cdot h^T = \begin{bmatrix} -0.6608 \\ 0.3291 \\ 0.3317 \end{bmatrix} [0.00,\quad 0.15] = \begin{bmatrix} 0 & -0.0991 \\ 0 & 0.0494 \\ 0 & 0.0498 \end{bmatrix}$$

Kemudian, bobot U diperbarui dengan Persamaan 3.6 menghasilkan bobot baru pada kolom kedua:

$$U_{baru} = U_{lama} - \eta \frac{\partial L}{\partial U}$$

$$U_{baru} = \begin{bmatrix} 0.10 & 0.25 \\ 0.20 & 0.05 \\ 0.15 & 0.10 \end{bmatrix} - 0.05 \times \begin{bmatrix} 0 & -0.0991 \\ 0 & 0.0494 \\ 0 & 0.0498 \end{bmatrix} = \begin{bmatrix} 0.10 & 0.2549 \\ 0.20 & 0.0475 \\ 0.15 & 0.0975 \end{bmatrix}$$

Keterangan:
t : vektor *one-hot* yang menunjukkan kata konteks sebenarnya.
y : vektor probabilitas hasil *softmax*.
$h = W^T x$: vektor representasi tersembunyi dari kata target.

Nilai-nilai pada matriks U inilah yang menjadi bobot baru setelah satu kali pembaruan (*one iteration*). Proses serupa juga dilakukan untuk matriks W menggunakan Persamaan 3.5, dan terus berulang hingga model mencapai konvergensi.

6. Iterasi Pelatihan

Langkah-langkah di atas diulang untuk seluruh pasangan target - konteks selama sejumlah *epoch* hingga model mencapai konvergensi (nilai loss stabil). Setelah pelatihan selesai, model menghasilkan matriks embedding berukuran $| V | \times N$, di mana $| V |$ adalah jumlah kata dalam kosakata, dan N adalah dimensi embedding yang ditetapkan.

Pada penelitian ini, misalnya, bila $| V | = 3$ dan $N = 4$ untuk contoh sederhana dengan kata "kamu", "cowok", "bodoh", maka hasil representasi embedding dapat disajikan seperti pada Tabel 3.10:

Tabel 3.10 Hasil representasi *embedding Word2vec*

Token	Vektor Embedding
kamu	$x_1 = [0.10, -0.20, 0.05, 0.30]$
cowok	$x_2 = [0.00, 0.10, -0.10, 0.20]$
bodoh	$x_3 = [0.20, -0.10, 0.00, 0.05]$

3.6 Pembagian Data

Tahap pembagian data (*split data*) merupakan langkah penting dalam proses pembelajaran mesin yang bertujuan untuk memisahkan dataset menjadi dua bagian utama, yaitu *training set* (data latih) dan *testing set* (data uji). Pemisahan ini dilakukan agar model dapat dilatih dan diuji secara objektif, sehingga performa yang dihasilkan benar-benar mencerminkan kemampuan model dalam melakukan generalisasi terhadap data baru yang belum pernah dilihat sebelumnya.

Pada penelitian ini, pembagian data dilakukan setelah tahap pembentukan representasi fitur menggunakan *Word2Vec*. Pendekatan ini bertujuan agar setiap kata dalam keseluruhan korpus *tweet* telah memperoleh representasi vektor yang

lengkap sebelum dataset dibagi. Dengan demikian, proses pembagian data tidak mengganggu pembentukan vektor kata yang digunakan sebagai masukan model.

Langkah tersebut juga dilakukan untuk menghindari permasalahan *out-of-vocabulary* (OOV), yaitu kondisi ketika kata yang muncul pada data uji tidak memiliki representasi vektor karena tidak ikut dilatih pada tahap pembentukan *Word2Vec*. Dengan membangun *embedding* menggunakan seluruh korpus terlebih dahulu, seluruh data uji tetap dapat diproses oleh model LSTM menggunakan *embedding matrix* yang sama dengan data latih.

Selain itu, karena penelitian ini menggunakan pendekatan klasifikasi *Multi-Label* dengan enam kategori label ujaran kebencian, proses pembagian data dilakukan menggunakan metode iterative stratification dengan rasio 80:20. Metode ini dipilih karena mampu mempertahankan distribusi label pada data latih dan data uji secara lebih seimbang, sehingga setiap label memiliki peluang yang proporsional untuk muncul pada kedua subset data.

3.7 Implementasi LSTM

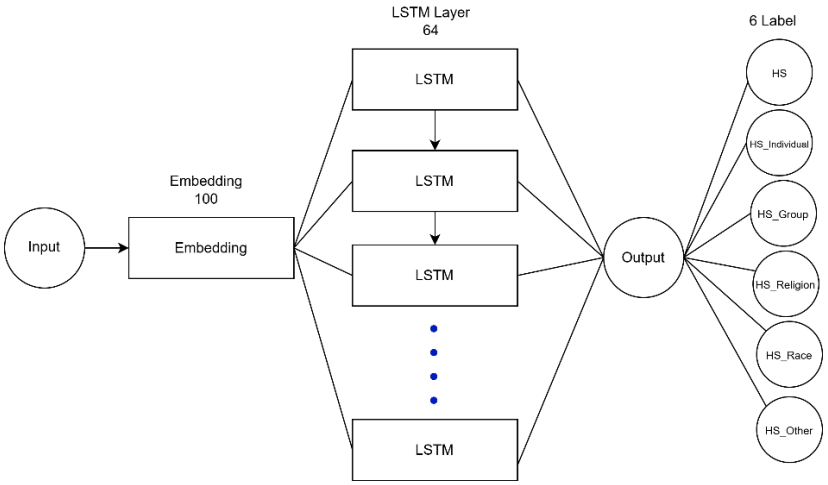
Implementasi LSTM pada penelitian ini dijelaskan secara rinci melalui tahapan perhitungan dan alur kerja model yang digunakan. LSTM merupakan salah satu varian *Recurrent Neural Network* (RNN) yang dirancang untuk memproses data berurutan dan mengatasi permasalahan *vanishing gradient* yang sering muncul pada RNN konvensional. Keunggulan utama LSTM terletak pada mekanisme *gating* meliputi *forget gate*, *input gate*, dan *output gate* yang memungkinkan jaringan untuk menyimpan, memperbarui, dan membuang informasi secara selektif pada setiap langkah waktu (*time step*). Dengan kemampuan ini, LSTM mampu

mempertahankan konteks jangka panjang pada data teks, sehingga performanya lebih stabil pada tugas pemrosesan bahasa alami.

Setelah memahami mekanisme internal pada satu unit LSTM, penelitian ini menjelaskan arsitektur lengkap model LSTM yang digunakan. Arsitektur ini terdiri atas tiga komponen utama, yaitu *Embedding Layer*, rangkaian lapisan LSTM bertingkat, dan *Output Layer* untuk klasifikasi *Multi-Label* dengan enam unit keluaran. Data yang telah melalui tahap prapemrosesan dikonversi menjadi indeks kata dan direpresentasikan sebagai vektor melalui *Embedding Layer*. Representasi ini kemudian diproses oleh beberapa lapisan LSTM untuk menangkap pola urutan kata dan konteks semantik pada berbagai tingkat kedalaman.

Hasil keluaran dari lapisan LSTM selanjutnya diproyeksikan ke *Output Layer*, yang menggunakan fungsi aktivasi *sigmoid* untuk menghasilkan probabilitas independen pada masing-masing enam kategori ujaran kebencian yang digunakan dalam penelitian ini. Dengan demikian, arsitektur ini menggambarkan bagaimana unit-unit LSTM bekerja secara terintegrasi dalam menyelesaikan tugas klasifikasi multilabel.

Gambaran contoh arsitektur model LSTM yang digunakan dalam penelitian ini ditunjukkan pada Gambar 3.5.



Gambar 3.5 Contoh arsitektur LSTM (64 layer)

Berdasarkan Gambar 3.5, dapat dilihat bahwa alur pemrosesan dimulai dari *Input* teks, dilanjutkan dengan pembentukan representasi kata pada *Embedding Layer*, pemodelan konteks melalui lapisan LSTM bertingkat, dan diakhiri dengan pemetaan ke enam neuron keluaran pada *Output Layer*. Seluruh komponen tersebut bekerja secara terintegrasi untuk menghasilkan prediksi kategori ujaran kebencian secara *Multi-Label*.

Setelah pemaparan arsitektur model secara konseptual, bagian selanjutnya akan membahas proses perhitungan yang terjadi pada setiap lapisan, mulai dari *Embedding Layer*, mekanisme kerja LSTM, hingga penerapan nilai *threshold* sebagai dasar penentuan keputusan klasifikasi akhir.

3.7.1 *Embedding Layer*

Lapisan *embedding* merupakan tahap awal dalam arsitektur model LSTM yang berfungsi mengubah token hasil pra-pemrosesan menjadi representasi vektor berdimensi tetap (*dense vector*).

Berbeda dengan *one-hot encoding*, yang menghasilkan vektor biner berdimensi besar dan jarang terisi (*sparse vector*), *embedding layer* menghasilkan representasi padat yang mampu menangkap hubungan semantik antar kata. Artinya, kata-kata dengan makna atau konteks serupa akan memiliki posisi vektor yang berdekatan, sedangkan kata-kata yang maknanya berlawanan akan berada lebih jauh satu sama lain di ruang vektor (Almeida & Xexéo, 2023).

Dalam penelitian ini, *embedding layer* tidak dibangun dari nol (*from scratch*), tetapi menggunakan bobot awal *(initial weights)* yang berasal dari hasil pelatihan model *Word2vec Skip-Gram* pada tahap sebelumnya. Model *Word2vec* menghasilkan *embedding matrix* berukuran $|V| \times N$, di mana $|V|$ adalah jumlah kata unik dalam korpus (*vocabulary size*) dan N adalah dimensi embedding yang digunakan (misalnya 200).

Matriks inilah yang dimasukkan sebagai bobot awal ke dalam *embedding layer*. Dengan cara ini, setiap kata dalam teks (misalnya dalam *tweet*) sudah memiliki representasi semantik yang sesuai konteks penggunaannya di korpus pelatihan.

Secara umum, setiap kata dalam teks terlebih dahulu diubah menjadi indeks numerik oleh proses *tokenization*. Misalnya, kata "kamu" mungkin diberi indeks 1, "cowok" indeks 2, dan "bodoh" indeks 3. Selanjutnya, indeks token ke-i (t_i) dipetakan menjadi vektor embedding melalui matriks bobot E. $E \in \mathbb{R}^{|V| \times N}$, sebagaimana ditunjukkan pada Persamaan 3.9:

$$e_i = E\,[t_i] \tag{3.9}$$

Persamaan 3.9 menunjukkan bahwa setiap token dengan indeks t_i akan mengambil baris ke-t_i dari matriks embedding E. Hasilnya adalah vektor embedding e_i berdimensi N, yang menjadi representasi numerik dari kata tersebut. Dengan kata lain, *embedding layer* berfungsi seperti kamus numerik, setiap kata memiliki alamat tertentu di dalam matriks E, dan ketika dipanggil melalui indeksnya, model mengambil baris yang berisi representasi vektor kata tersebut.

Hasil dari proses *embedding lookup* pada setiap kata kemudian disusun menjadi satu matriks yang mewakili keseluruhan kalimat. Jika panjang kalimat setelah proses *padding* adalah L token, maka bentuk umum matriks representasi embedding dinyatakan pada Persamaan (3.10):

$$X \in \mathbb{R}^{L \times N} \tag{3.10}$$

Persamaan 3.10 menunjukkan bahwa keluaran dari *embedding layer* berupa matriks X berdimensi $L \times N$, di mana L menyatakan jumlah token dalam satu kalimat (*sequence length*), dan N menunjukkan jumlah dimensi embedding untuk setiap token. Rumus ini digunakan untuk menggambarkan struktur data hasil transformasi teks ke dalam bentuk numerik sebelum diproses oleh LSTM. Notasi ini merupakan konvensi umum dalam teori *deep learning* dan bukan hasil eksperimen tertentu.

Sebagai ilustrasi, kalimat *"kamu cowok bodoh"* digunakan untuk menggambarkan proses pemetaan kata menjadi vektor embedding. Tabel 3.11 menampilkan hasil embedding kata yang diperoleh dari pelatihan *Word2vec Skip-*

Gram. Nilai-nilai pada tabel ini digunakan sebagai bobot awal (*initial weights*) dalam *embedding layer* pada model LSTM.

Tabel 3.11 Hasil Embedding Kata dari *Word2vec Skip-Gram*

Token	Vektor Embedding
kamu	$[0.10, -0.20, 0.05, 0.30]$
cowok	$[0.00, 0.10, -0.10, 0.20]$
bodoh	$[0.20, -0.10, 0.00, 0.05]$

Tabel 3.11 menunjukkan representasi numerik (*embedding vector*) dari tiga kata hasil pelatihan *Word2vec*. Setiap baris pada tabel mewakili satu token, sedangkan setiap kolom menggambarkan fitur semantik yang dipelajari oleh model berdasarkan konteks kemunculan kata di korpus.

Selanjutnya, seluruh vektor embedding dari kata-kata dalam satu kalimat disusun menjadi sebuah matriks yang merepresentasikan kalimat secara utuh. Bentuk matematis matriks kalimat tersebut ditunjukkan pada Persamaan 3.11:

$$X = \begin{bmatrix} 0.10 & -0.20 & 0.05 & 0.30 \\ 0.00 & 0.10 & -0.10 & 0.20 \\ 0.20 & -0.10 & 0.00 & 0.05 \end{bmatrix} \in \mathbb{R}^{3\times4} \tag{3.11}$$

Persamaan 3.11 menggambarkan matriks representasi embedding untuk kalimat *"kamu cowok bodoh"*, yang terdiri atas tiga token dengan dimensi embedding empat. Setiap baris pada matriks mewakili satu token, sedangkan setiap kolom mewakili dimensi vektor embedding yang dipelajari oleh model.

Dengan demikian, matriks X merupakan bentuk representasi numerik dari kalimat yang akan digunakan sebagai masukan (*input*) pada lapisan LSTM untuk mempelajari konteks urutan kata secara temporal.

3.7.2 *LSTM Layer*

Setelah melewati tahap *Embedding Layer*, setiap token pada teks telah diubah menjadi vektor berdimensi tetap yang merepresentasikan makna semantik dari kata tersebut. Sebagai ilustrasi, token pertama pada kalimat *"kamu cowok bodoh"*, yaitu kata *"kamu"*, direpresentasikan oleh vektor:

$$x_1 = [0.10, -0.20, 0.05, 0.30]$$

Vektor ini menjadi masukan (*input vector*) bagi model LSTM untuk menghasilkan representasi kontekstual yang memperhatikan hubungan antar kata dalam kalimat.

Pada awal proses, model belum memiliki konteks dari kata sebelumnya, sehingga *hidden state* (h_0) dan *cell state* (c_0) diinisialisasi dengan nilai nol. Kondisi ini menunjukkan bahwa jaringan belum menyimpan informasi apa pun dari langkah sebelumnya.

Dalam LSTM, semua bobot jaringan baik bobot masukan (W), bobot rekuren (U), maupun bias (b) digunakan secara berulang di setiap langkah pemrosesan kata dalam satu urutan kalimat. Mekanisme ini disebut *parameter sharing*, di mana bobot yang sama digunakan agar model dapat mempelajari pola urutan secara konsisten.

Pembaruan bobot tidak dilakukan setiap kali satu kata diproses, melainkan setelah seluruh urutan kata selesai melalui jaringan. Proses pembaruan dilakukan menggunakan metode *Backpropagation Through Time* (BPTT), yaitu mekanisme yang menelusuri kembali urutan data untuk memperbaiki bobot berdasarkan selisih

antara hasil prediksi dan nilai sebenarnya. Dengan demikian, setiap *mini-batch* data hanya akan menghasilkan satu kali pembaruan bobot.

Pada perhitungan berikut, langkah-langkah yang dijelaskan menggambarkan bagaimana model memproses token pertama ($t = 1$), di mana nilai bias dianggap nol dan komponen rekuren (Uh_{t-1}) belum berpengaruh karena $h_0 = 0$.

$$h_0 = [0, 0],\ c_0 = [0,\ 0]$$

1. Proses perhitungan dimulai dengan *forget gate*, yang berfungsi menentukan informasi lama mana yang masih relevan untuk dipertahankan dalam memori. Gerbang ini bekerja dengan menghitung keluaran dari fungsi aktivasi sigmoid sebagaimana dinyatakan pada Persamaan 2.1. Karena $h_0 = 0$, perhitungan disederhanakan menjadi:

$$f_1 = \sigma\left(x_1 W_{xf}\right)$$

Dengan matriks bobot:

$$W_{xf} = \begin{bmatrix} 0.10 & -0.20 \\ -0.20 & 0.10 \\ 0.05 & 0.20 \\ 0.30 & 0.05 \end{bmatrix}$$

Hasil perkalian menghasilkan:

$$x_1 W_{xf} = [0.1425, -0.015]$$

Setelah melewati fungsi aktivasi *sigmoid*, diperoleh:

$$f_1 = \sigma\left([0.1425, -0.015]\right) \approx [0.535, 0.496].$$

Nilai ini menunjukkan bahwa sekitar 53,5% informasi lama pada dimensi pertama dipertahankan, sedangkan sisanya dihapus.

2. Selanjutnya, *input gate* mengatur seberapa besar informasi baru dari token saat ini akan dimasukkan ke dalam memori. Perhitungannya mengacu pada Persamaan 2.2. Maka, karena $h_0 = 0$, perhitungan disederhanakan menjadi:

$$i_1 = \sigma\,(x_1 W_{xi})$$

Dengan matriks bobot:

$$W_{xi} = \begin{bmatrix} 0.05 & 0.10 \\ -0.10 & 0.20 \\ 0.10 & 0.00 \\ 0.05 & -0.05 \end{bmatrix}$$

Perhitungannya menjadi:

$$x_1 W_{xi} = [0.045, -0.045]$$

Dengan aktivasi sigmoid:

$$i_1 = \sigma\,([0.045, -0.045]) \approx [0.511, 0.489].$$

Hasil ini mengindikasikan bahwa sekitar 51.1% informasi baru akan diizinkan masuk, dengan kandidat informasi baru yang direpresentasikan oleh vektor.

3. Setelah itu, jaringan menghitung *candidate cell state* ($\tilde{C}_t$) yang mewakili kandidat informasi baru yang akan disimpan. Persamaannya mengacu pada Persamaan 2.3.

$$\tilde{C}_t = \tanh\,(x_1 W_{xc})$$

Dengan bobot:

$$W_{xc} = \begin{bmatrix} 0.20 & -0.10 \\ 0.10 & 0.00 \\ -0.20 & 0.30 \\ 0.05 & 0.25 \end{bmatrix}$$

Hasil perkalian menghasilkan:

$$x_1 W_{xc} = [0.005, 0.080]$$

Dengan aktivasi $tanh$:

$$\tilde{C}_1 = tanh([0.005, 0.080]) \approx [0.005, 0.080].$$

Ini merepresentasikan memori jangka panjang model setelah memproses token pertama, "kamu."

4. Berikutnya, memori jangka panjang diperbarui dengan menggabungkan informasi lama yang telah difilter oleh *forget gate* dan informasi baru dari *input gate* serta *candidate cell state*. Persamaannya mengacu pada Persamaan 2.4:

   Karena $c_0 = [0,0]$, maka:

$$c_1 = i_1 \cdot \tilde{C}_1 = [0.511 \times 0.005, 0.489 \times 0.080]$$

$$c_1 \approx [0.0026, 0.0391]$$

   Hasil ini menunjukkan memori jangka panjang yang terbentuk setelah memproses token pertama.

5. Proses berlanjut dengan *output gate*, yang menentukan seberapa banyak informasi dari *cell state* akan diteruskan sebagai *hidden state*. Mengacu pada Persamaan 2.5.

   Maka, karena $h_0 = 0$:

$$o_1 = \sigma(x_1 W_{xo})$$

   Dengan bobot:

$$W_{xo} = \begin{bmatrix} -0.10 & 0.00 \\ 0.20 & -0.10 \\ 0.10 & 0.10 \\ 0.00 & 0.20 \end{bmatrix}$$

   Perhitungannya menjadi:

$$x_1 W_{xo} = [-0.045, 0.090]$$

Dengan aktivasi sigmoid:

$$o_1 = \sigma([-0.045, 0.090]) \approx [0.489, 0.523].$$

Nilai ini mengontrol seberapa besar proporsi informasi dari memori yang akan diteruskan ke keluaran jaringan.

6. Langkah terakhir menghasilkan *hidden state* (h_t) sebagai keluaran jaringan pada waktu ke-t, sebagaimana dijelaskan pada Persamaan 2.6:

   Dengan:

$$tanh(c_1) \approx [0.0026, 0.0391]$$

   Maka:

$$h_1 = [0.489 \times 0.0026, 0.523 \times 0.0391]$$

$$h_1 \approx [0.00127, 0.02045]$$

Nilai h_1 inilah yang menjadi representasi kontekstual kata *"kamu"*, yang tidak hanya membawa makna leksikal, tetapi juga konteks dari urutan kalimat. Nilai ini kemudian digunakan untuk memproses token selanjutnya hingga seluruh kalimat selesai dianalisis oleh jaringan.

Hasil keluaran dari setiap *hidden state* pada lapisan LSTM merepresentasikan pemahaman model terhadap konteks kata dalam urutan kalimat. Nilai-nilai ini memuat informasi semantik dan hubungan temporal yang telah dipelajari dari data sebelumnya. Setelah seluruh token dalam kalimat diproses, *hidden state* terakhir menyimpan representasi yang paling lengkap dari keseluruhan konteks teks. Representasi inilah yang kemudian digunakan sebagai masukan bagi lapisan selanjutnya, yaitu *Dense Layer*, untuk melakukan proses klasifikasi emosi

secara lebih spesifik berdasarkan pola yang telah terbentuk pada tahap pembelajaran sebelumnya.

3.7.3 *Dense Layer*

Setelah *hidden state* terakhir dari LSTM diperoleh (misalnya dari token pertama kita dapat $h_1 \approx [0.00127, 0.02045]$, vektor representasi ini kemudian diproyeksikan ke lapisan *Dense Hidden Layer* atau *fully connected layer*. Lapisan ini berfungsi untuk melakukan transformasi lanjutan terhadap representasi keluaran LSTM agar lebih kaya secara fitur dan siap digunakan untuk klasifikasi pada lapisan keluaran (*output layer*).

Secara konsep, *Dense Layer* bekerja seperti jaringan saraf tiruan (*Artificial Neural Network*) klasik, di mana setiap neuron pada lapisan ini terhubung dengan seluruh neuron pada lapisan sebelumnya. Tujuan utamanya adalah mengombinasikan informasi yang diperoleh dari *hidden state* LSTM secara non-linear untuk membentuk representasi yang lebih diskriminatif terhadap pola data (Javid *et al*., 2021).

Secara matematis, operasi pada *Dense Layer* ditunjukkan pada Persamaan 3.12.

$$z_{hidden} = h \cdot W_{hidden} + b_{hidden} \tag{3.12}$$

Persamaan 3.12 merupakan fungsi linier neuron yang digunakan untuk menghitung nilai sebelum diberi fungsi aktivasi. Rumus ini menjelaskan bahwa vektor *hidden state* hasil keluaran LSTM (h) dikalikan dengan matriks bobot

(W_{hidden}) dan ditambahkan dengan bias (b_{hidden}) untuk menghasilkan keluaran linier z_{hidden}.

Keterangan:
h : hidden state keluaran LSTM.
W_{hidden} : matriks bobot yang menghubungkan hidden state dengan neuron-neuron pada lapisan dense.
b_{hidden} : bias.

Rumus ini digunakan untuk mengubah ruang representasi yang diperoleh dari LSTM menjadi ruang fitur baru yang lebih padat. Karena operasi pada Persamaan ini masih bersifat linier, maka diperlukan fungsi aktivasi agar model mampu mempelajari hubungan non-linier antar fitur.

Fungsi aktivasi yang digunakan pada penelitian ini adalah *Rectified Linear Unit (ReLU)*, dengan formulasi yang ditunjukkan pada Persamaan 3.13:

$$ReLU(x) = max(0, x) \tag{3.13}$$

Persamaan 3.13 menunjukkan bahwa fungsi *ReLU* hanya mempertahankan nilai positif dan mengubah semua nilai negatif menjadi nol. Fungsi ini digunakan untuk memberikan non-linearitas pada jaringan sehingga model dapat belajar dari pola data yang kompleks, sekaligus mencegah terjadinya *vanishing gradient* saat proses pelatihan (Dubey *et al*., 2022).

Melanjutkan ilustrasi proses perhitungan, dengan hidden state terakhir dari LSTM yang diperoleh:

$$h_1 = [0.00127, 0.02045]$$

Kemudian diteruskan ke *Dense Layer* yang diasumsikan memiliki tiga neuron keluaran dengan bobot berukuran 2×3 (artinya menghasilkan 3 neuron):

$$W_{hidden} = \begin{bmatrix} 0.5 & -0.2 & 0.1 \\ 0.3 & 0.4 & -0.1 \end{bmatrix},\ b_{hidden} = [0, \quad 0, \quad 0]$$

Perhitungan linier dilakukan menggunakan Persamaan 3.12:

- Neuron 1: $0.00127(0.5) + 0.02045(0.3) = 0.00677$
- Neuron 2: $0.00127(-0.5) + 0.02045(0.4) = 0.00793$
- Neuron 3: $0.00127(0.1) + 0.02045(-0.1) = -0.00192$

Sehingga diperoleh:

$$z_{hidden} \approx [0.00677, \quad 0.00793, \quad -0.00192]$$

Nilai z_{hidden} tersebut merupakan keluaran linier dari tiga neuron sebelum fungsi aktivasi diterapkan. Selanjutnya, setiap nilai pada z_{hidden} diproses menggunakan fungsi *ReLU* berdasarkan Persamaan 3.13:

$$h' = ReLU(z_{hidden}) \approx [0, .00677, \quad 0.00793, \quad 0]$$

Output dari Dense Hidden Layer ini berupa vektor h' berdimensi 3 yang akan menjadi representasi baru dari input *tweet* sebelum diproyeksikan ke lapisan keluaran (output layer). Hasil tersebut menunjukkan bahwa ReLU mengubah nilai negatif menjadi nol, sedangkan nilai positif tetap dipertahankan. Dengan demikian, lapisan dense ini berfungsi memberikan non-linearitas pada model, memperkaya representasi fitur, serta mengurangi potensi masalah *vanishing gradient* sebelum diproyeksikan lebih lanjut ke lapisan keluaran.

3.7.4 *Output Layer*

Setelah melewati *Dense Hidden Layer* dengan fungsi aktivasi *ReLU*, diperoleh vektor representasi laten $\hat{h}'$ yang berfungsi sebagai masukan ke *Dense Output Layer*. Lapisan ini merupakan tahap akhir dari arsitektur model, yang

berperan penting dalam mengubah hasil representasi fitur menjadi nilai probabilitas klasifikasi.

Dalam konteks penelitian ini, model digunakan untuk mendeteksi ujaran kebencian dengan enam kategori label. Oleh karena itu, *Output Layer* terdiri atas enam neuron, di mana masing-masing neuron bertugas memprediksi peluang kemunculan satu label ujaran kebencian *(HS, HS_Individual, HS_Group, HS_Religion, HS_Race , dan HS_Other)*.

Secara matematis, proses pada lapisan keluaran dimulai dengan operasi linier yang perhitungannya ditujukan pada Persamaan 3.14.

$$z = h'W_{out} + b_{out} \tag{3.14}$$

Keterangan:
h' : hasil dari dense hidden layer.
W_{out} : matriks bobot.
b_{out} : vektor bias.
z : vektor skor linear (*logit*) yang mewakili kekuatan prediksi setiap neuron sebelum fungsi aktivasi diterapkan.

Persamaan 3.14 disebut fungsi linier neuron keluaran (*output neuron linear transformation*). Rumus ini digunakan untuk menghitung *nilai logit* (skor mentah sebelum aktivasi) dari masing-masing neuron pada lapisan keluaran. Nilai ini diperoleh melalui perkalian antara vektor hasil lapisan sebelumnya h' dengan matriks bobot keluaran (W_{out}), kemudian ditambahkan dengan bias (b_{out}).

Dalam jaringan saraf tiruan, Persamaan linier ini berfungsi sebagai tahap transformasi terakhir yang menghubungkan fitur hasil pembelajaran ke domain keluaran. Namun, karena hasilnya masih berupa nilai mentah (belum dalam bentuk probabilitas), tahap selanjutnya memerlukan fungsi aktivasi untuk menormalkan nilainya.

Nilai z_k yang dihasilkan pada setiap neuron keluaran kemudian diproses menggunakan fungsi aktivasi *sigmoid*, yang dinyatakan dalam Persamaan 3.15:

$$\hat{y}_k = \sigma(z_k) = \frac{1}{1+e^{-z_k}},\ k = 1, \ldots, 6. \tag{3.15}$$

Persamaan 3.15 disebut fungsi aktivasi sigmoid, yaitu fungsi non-linear yang digunakan untuk mengubah skor linier (logit) menjadi nilai probabilitas dengan rentang antara 0 hingga 1. Fungsi ini memastikan bahwa setiap neuron keluaran dapat menghasilkan probabilitas independen untuk masing-masing label. Fungsi sigmoid memiliki bentuk kurva S yang halus, ketika z_k bernilai besar (positif), hasilnya mendekati 1; dan jika z_k bernilai negatif besar, hasilnya mendekati 0. Sifat ini membuat fungsi sigmoid sangat cocok untuk kasus klasifikasi *Multi-Label*, karena setiap neuron pada *Output Layer* dapat memberikan prediksi secara independent (Y. Zhang *et al*., 2023).

Melanjutkan ilustrasi proses perhitungan, Dari hasil lapisan *dense* sebelumnya diperoleh.

$$h' = [0.00677,\quad 0.00793,\quad 0]$$

Karena model memiliki enam label, maka *Output Layer* juga memiliki enam neuron. Berdasarkan Persamaan 3.14, skor linier dihitung sebagai berikut:

$$z = h'W_{out} + b_{out}, z \in \mathbb{R}^6$$

Dalam ilustrasi manual ini, bias diambil bernilai nol untuk menyederhanakan proses. Inisialisasi $b_{out} = 0$. Dan matriks bobot yang digunakan adalah:

$$W_{out} = \begin{bmatrix} 0.50 & 0.20 & -0.10 & 0.00 & 0.30 & -0.20 \\ 0.10 & -0.30 & 0.20 & 0.40 & -0.10 & 0.00 \\ 0.00 & 0.10 & -0.20 & 0.10 & 0.00 & 0.30 \end{bmatrix}$$

Selanjutnya, dilakukan perhitungan nilai $z_k$untuk setiap neuron (6 label):

1. Label 1

$$z_1 = (0.00677)(0.50) + (0.00793)(0.10) + (0)(0.00)$$
$$= 0.003385 + 0.000793$$
$$\approx 0.004178 \rightarrow 0.00418$$

2. Label 2

$$z_2 = (0.00677)(0.20) + (0.00793)(-0.30) + (0)(0.10)$$
$$= 0.001354 + 0.002378$$
$$\approx -0.001024 \rightarrow -0.00102$$

Hasil dari perhitungan linear memberikan delapan skor keluaran,

$$z \approx [0.00418, -0.00102, 0.00091, 0.00317, 0.00124, -0.00135]$$

Setiap skor z_k kemudian dilewatkan ke fungsi aktivasi sigmoid untuk diubah menjadi probabilitas, sesuai Persamaan 3.15. Hasil akhir berupa vektor probabilitas *Multi-Label* dituliskan:

$$\hat{y} \approx [0.5010, 0.4997, 0.5002, 0.5008, 0.5003, 0.4997]$$

Setiap nilai dalam vektor $\hat{y}$ merepresentasikan probabilitas kemunculan satu label ujaran kebencian dari enam label yang digunakan. Karena pendekatan ini bersifat *Multi-Label*, maka seluruh nilai probabilitas tersebut tidak dijumlahkan, melainkan diperlakukan secara independen.

3.7.5 *Binary Cross-Entropy (BCE) Loss Function*

Setelah model menghasilkan nilai probabilitas melalui *output layer* menggunakan fungsi aktivasi *sigmoid*, langkah berikutnya adalah menghitung seberapa jauh hasil prediksi tersebut berbeda dari label sebenarnya. Tahap ini

penting karena dari sinilah model dapat "belajar" memperbaiki kesalahannya. Fungsi yang digunakan untuk mengukur tingkat kesalahan itu disebut *loss function* atau fungsi rugi (M.-L. Zhang & Zhou, 2014).

Dalam penelitian ini digunakan fungsi *Binary Cross-Entropy* (BCE) karena permasalahan yang diselesaikan bersifat *Multi-Label*, artinya setiap data dapat memiliki lebih dari satu label aktif secara bersamaan. Contohnya, satu *tweet* dapat mengandung ujaran kebencian terhadap individu *(HS_Individual)* sekaligus terhadap kelompok tertentu *(HS_Group)*. Maka, setiap label harus dievaluasi secara terpisah.

Fungsi BCE digunakan untuk mengukur jarak antara probabilitas hasil prediksi model dengan label sebenarnya. Nilai ini menunjukkan tingkat kesalahan model, semakin kecil nilainya, semakin akurat model mengenali data. Secara konsep, BCE akan memberi hukuman (penalti) lebih besar jika model sangat yakin terhadap prediksi yang ternyata salah, dan memberi nilai kecil jika prediksinya benar. Dengan cara ini, model "belajar" untuk tidak mengulang kesalahan yang sama pada iterasi berikutnya.

Untuk satu label ke-*k*, fungsi *Binary Cross-Entropy* dirumuskan dengan Persamaan 3.16:

$$BCE_k = -[y_k \cdot \log(\hat{y}_k) + (1 - y_k) \cdot \log(1 - \hat{y}_k)] \quad (3.16)$$

Keterangan:
$y_k \in \{0, 1\}$: label sebenarnya dari data ke-*k* (1 untuk positif, 0 untuk negatif).
$\hat{y}_k \in [0, 1]$: probabilitas hasil prediksi model pada label ke-*k* setelah fungsi *sigmoid*.
BCE_k : nilai kesalahan (loss) untuk satu label.

Persamaan 3.16 di atas menunjukkan bahwa ketika model memprediksi nilai $\hat{y}_k$ yang mendekati label sebenarnya y_k, maka nilai BCE akan semakin kecil. Sebaliknya, jika prediksi jauh dari label yang benar, nilai BCE akan membesar. Fungsi logaritma dalam rumus ini memberikan penalti lebih besar ketika model yakin terhadap prediksi yang salah, sehingga model terdorong untuk belajar lebih hati-hati (Cunha, 2022).

Pada penelitian ini karena terdapat enam label ujaran kebencian, maka total *loss* untuk satu data dihitung dengan menjumlahkan semua nilai BCE dari setiap label, kemudian dirata-ratakan menggunakan Persamaan 3.17:

$$L = \frac{1}{6} \sum_{k=1}^{6} BCE_k \tag{3.17}$$

Keterangan:
L : total *loss* untuk satu data (satu *tweet*).
K : 6 : jumlah label dalam klasifikasi *Multi-Label*.

Persamaan 3.17 ini digunakan agar setiap label memiliki pengaruh yang seimbang selama proses pelatihan model. Dengan kata lain, model tidak hanya fokus pada satu kategori, tetapi belajar secara seimbang terhadap semua jenis ujaran kebencian.

Melanjutkan ilustrasi proses perhitungan, dari hasil pada sebuah *tweet* diperoleh hasil prediksi probabilitas dari lapisan *Dense Output (Sigmoid)* sebelumnya diperoleh:

$$\hat{y} \approx [0.5010, 0.4997, 0.5002, 0.5008, 0.5003, 0.4997]$$

Sedangkan, label sebenarnya *(ground truth)* untuk data tersebut adalah:

$$y = [1, \quad 1, \quad 0, \quad 0, \quad 0, \quad 0]$$

Langkah selanjutnya adalah menghitung *Binary Cross-Entropy* untuk setiap label:

- $BCE_1 = -[1 \cdot \log(0.5010) + 0 \cdot \log(1 - 0.5010)] = -\log(0.5010) = 0.6911$
- $BCE_2 = -[1 \cdot \log(0.4997) + 0 \cdot \log(1 - 0.4997)] = -\log(0.4997) = 0.6932$
- $BCE_3 = -[0 \cdot \log(0.5002) + 1 \cdot \log(1 - 0.5002)] = -\log(0.4998) = 0.6934$
- $BCE_4 = -[0 \cdot \log(0.5008) + 1 \cdot \log(1 - 0.5008)] = -\log(0.4992) = 0.6943$
- $BCE_5 = -[0 \cdot \log(0.5003) + 1 \cdot \log(1 - 0.5003)] = -\log(0.4997) = 0.6933$
- $BCE_6 = -[0 \cdot \log(0.4997) + 1 \cdot \log(1 - 0.4997)] = -\log(0.5003) = 0.6926$

Kemudian total *loss* untuk satu data dihitung dengan merata-ratakan seluruh nilai BCE:

$$L \approx \frac{1}{6}(0.6911 + 0.6932 + 0.6934 + 0.6943 + 0.6933 + 0.6926)$$

$$L \approx 0.6930$$

Nilai $L = 0.6930$ ini menunjukkan tingkat kesalahan model untuk satu data masukan. Nilai ini belum bisa dikatakan baik atau buruk tanpa dibandingkan dengan nilai *loss* dari iterasi pelatihan lain, tetapi semakin kecil nilainya, semakin baik model dalam mengenali pola pada data.

3.7.6 *Thresholding*

Setelah model menghasilkan nilai probabilitas melalui fungsi aktivasi *sigmoid* pada lapisan keluaran, tahap berikutnya adalah mengubah probabilitas tersebut menjadi keputusan klasifikasi biner. Proses ini dikenal dengan istilah *threshold*ing, yang merupakan mekanisme untuk menentukan apakah suatu label diklasifikasikan sebagai positif atau negatif berdasarkan ambang batas tertentu (τ).

Pada klasifikasi *Multi-Label*, nilai keluaran $\hat{y}_k$ untuk setiap label k akan berada dalam rentang 0 hingga 1. Nilai ini merepresentasikan probabilitas dari kemungkinan label tersebut aktif. Namun, probabilitas tersebut belum dapat digunakan sebagai keputusan akhir klasifikasi. Untuk itu, dilakukan *threshold*ing

untuk mengubah nilai probabilitas tersebut menjadi label biner, yaitu 1 (aktif) atau 0 (tidak aktif), yang dapat lebih mudah dianalisis.

Secara matematis, proses *threshold*ing ini dituliskan pada Persamaan 3.18:

$$\tilde{y}_k = \begin{cases} 1, \hat{y}_k \geq \tau \\ 0, \hat{y}_k < \tau \end{cases} \quad (3.18)$$

Keterangan:
$\tilde{y}_k$: hasil klasifikasi biner untuk label ke-k.
$\hat{y}_k$: probabilitas hasil prediksi model untuk label ke-k.
τ : nilai ambang batas (*threshold*), yang umumnya ditetapkan pada 0,5.

Persamaan 3.18 menunjukkan bagaimana probabilitas $\hat{y}_k$ diubah menjadi keputusan biner berdasarkan ambang batas. Jika probabilitas $\hat{y}_k$ lebih besar atau sama dengan τ, maka label dianggap aktif (1). Sebaliknya, jika probabilitasnya lebih kecil dari τ, label dianggap tidak aktif (0).

Melanjutkan ilustrasi proses perhitungan, dari hasil pada sebuah *tweet* diperoleh hasil prediksi probabilitas dari lapisan Dense Output *(Sigmoid)* sebelumnya diperoleh

$$\hat{y} \approx [0.5010, 0.4997, 0.5002, 0.5008, 0.5003, 0.4997]$$

Jika diterapkan *threshold* τ = 0.5, maka hasilnya menjadi:

$$\tilde{y} = [1, \quad 0, \quad 1, \quad 1, \quad 1, \quad 0]$$

Interpretasi dari hasil di atas adalah sebagai berikut:

- Label ke-1, ke-3, ke-4, dan ke-5 bernilai 1 (aktif), yang menunjukkan bahwa *tweet* terdeteksi mengandung ujaran kebencian pada kategori *(HS)*, kelompok *(HS_Group),* agama *(HS_Religion)*, dan ras atau etnis *(HS_Race)*.

- Sementara itu, label ke-2 dan ke-6 bernilai 0 (tidak aktif), sehingga ujaran kebencian terhadap individu *(HS_Individual)* dan kategori lainnya *(HS_Other)* tidak terdeteksi pada kasus ini.

Dengan demikian, hasil *thresholding* memberikan representasi akhir dalam bentuk vektor biner yang dapat digunakan untuk tahap evaluasi kinerja model, seperti perhitungan *precision*, *recall*, dan *F1-score*.

3.8 Evaluasi Model

Setelah model selesai dilatih, tahap berikutnya adalah melakukan evaluasi terhadap performa model pada data uji. Evaluasi ini bertujuan untuk mengetahui seberapa baik model dalam memprediksi enam kategori ujaran kebencian secara bersamaan.

3.8.1 Metrik Evaluasi *Multi-Label*

Setelah keluaran probabilitas $\hat{y}$ dikonversi menjadi label biner $\tilde{y}$ melalui proses *threshold*ing, tahap berikutnya adalah mengevaluasi performa model. Karena kasus yang dihadapi adalah klasifikasi *Multi-Label*, maka evaluasi tidak cukup hanya mengukur akurasi sederhana, melainkan perlu menggunakan metrik yang mampu menilai kinerja model pada tiap label maupun secara keseluruhan. Beberapa metrik utama yang digunakan adalah sebagai berikut:

1. *Subset Accuracy (Exact Match)*

Subset accuracy menyatakan proporsi prediksi yang benar hanya jika seluruh label sesuai dengan label sebenarnya (y). Dengan kata lain, prediksi satu

tweet dianggap benar apabila semua label tepat. Rumusnya dituliskan pada Persamaan 2.15.

Berdasarkan hasil perhitungan manual, diperoleh hasil prediksi yang dilakukan model,

$$\tilde{y} = [1, \quad 0, \quad 1, \quad 1, \quad 1, \quad 0]$$

Sedangkan label sebenarnya adalah

$$y = [1, \quad 1, \quad 0, \quad 0, \quad 0, \quad 0]$$

Karena kedua vektor tidak identik, maka *subset accuracy* bernilai 0. Hal ini menunjukkan bahwa meskipun sebagian besar label sesuai, prediksi tersebut tetap dinilai salah secara keseluruhan.

2. *Hamming Loss*

Hamming loss mengukur rata-rata kesalahan per label, yaitu proporsi label yang diprediksi salah dibandingkan dengan total label, Rumusnya dituliskan pada Persamaan 2.7.

Dari kasus di atas, terdapat empat kesalahan prediksi pada label ke-2, ke-3, ke-4, dan ke-5 (4 kesalahan dari total 6 label), maka:

$$HammingLoss = \frac{4}{6} = 0.667$$

Model melakukan kesalahan prediksi pada 66,7% label untuk satu *tweet* ini. Nilai yang tinggi menandakan model terlalu banyak memberikan label yang tidak relevan.

3. *Accuracy, Precision, Recall,* dan *F1-score*

Untuk tiap label dihitung True Positive (TP), False Positive (FP), dan False Negative (FN). Berdasarkan hasil perhitungan manual, diperoleh hasil prediksi yang dilakukan model,

$$\tilde{y} = [1, \quad 0, \quad 1, \quad 1, \quad 1, \quad 0]$$

Sedangkan label sebenarnya adalah

$$y = [1, \quad 1, \quad 0, \quad 0, \quad 0, \quad 0]$$

Diperoleh nilai-nilai berikut:

a. *True Positive* (TP) = 1 (label ke-1 sesuai dengan label sebenarnya),
b. *False Positive* (FP) = 3 (label ke-3, ke-4, dan ke-5 (diprediksi positif namun ternyata salah),
c. *False Negative* (FN) = 1 (label ke-2 seharusnya positif tetapi tidak berhasil dikenali oleh model).
d. Selain itu, terdapat *True Negative* (TN) = 1 (label ke-6 sesuai dengan kondisi negatif sebenarnya).

Dengan demikian, sesuai dengan rumus pada Persamaan 2.8, 2.10, 2.12, dan 2.14.

- $Accuracy = \frac{1+1}{1+3+1+1} = \frac{2}{6} = 0.333$
- $Precision = \frac{1}{1+3} = 0.25$
- $Recall = \frac{1}{1+1} = 0.50$
- $F1 = \frac{2 \cdot 0.25 \cdot 0.50}{0.25 + 0.50} = 0.333$

Nilai *accuracy* sebesar 33,3% menunjukkan bahwa model hanya mampu memprediksi dengan benar sebagian kecil label. *Precision* yang relatif rendah mengindikasikan masih banyak prediksi positif yang keliru, sementara *recall* sebesar 50% menunjukkan bahwa setengah dari label positif berhasil dikenali oleh model. Nilai *F1-score* mencerminkan bahwa keseimbangan antara ketepatan dan kelengkapan prediksi masih perlu ditingkatkan.

4. *Macro Average Precision, Recall,* dan *F1-Score*

Dalam evaluasi *Multi-Label*, *macro average* digunakan untuk menghitung metrik evaluasi secara rata-rata di seluruh label, dengan memberikan bobot yang sama pada setiap label.

Sebagai ilustrasi, jika terdapat hasil evaluasi untuk 3 label *tweet*, maka, perhitungan *Recall, Precision,* dan *F1-Score* sesuai dengan rumus pada Persamaan 2.9, 2.11, dan 2.13 untuk tiap label *tweet* adalah sebagai berikut:

- Label *tweet* 1: *Precision* = 0.20, Recall = 0.50, F1-Score = 0.286
- Label *tweet* 2: *Precision* = 0.30, Recall = 0.60, F1-Score = 0.400
- Label *tweet* 3: *Precision* = 0.40, Recall = 0.80, F1-Score = 0.533

Untuk menghitung *macro average* dari *Precision*, Recall, dan F1-Score, cukup merata-ratakan nilai-nilai untuk masing-masing metrik.

a. *Macro Average Precision*:

$$Precision_{macro} = \frac{0.20 + 0.30 + 0.40}{3} = \frac{0.90}{3} = 0.30$$

b. *Macro Average Recall*:

$$Recall_{macro} = \frac{0.50 + 0.60 + 0.80}{3} = \frac{1.90}{3} = 0.63$$

c. *Macro Average F1-Score*:

$$F1_{macro} = \frac{0.286 + 0.400 + 0.533}{3} = \frac{1.219}{3} = 0.406$$

Dengan cara ini, setiap label dihitung secara terpisah, dan hasilnya dirata-ratakan untuk mendapatkan *macro average* untuk masing-masing metrik. *Precision* sebesar 0.30 menunjukkan bahwa secara keseluruhan, model sering kali salah dalam memprediksi label positif. *Recall* yang lebih tinggi (0.63) menunjukkan bahwa model mampu menemukan lebih banyak label positif yang benar. Namun, ada potensi untuk memperbaiki recall lebih lanjut. *F1-Score* sebesar 0.406 menunjukkan bahwa meskipun model dapat menangkap lebih banyak label positif, keseimbangan antara *Precision* dan *Recall* masih perlu ditingkatkan.

3.9 Skenario Uji

Dalam penelitian ini, skenario pengujian difungsikan sekaligus sebagai proses *hyperparameter tuning*. Setiap skenario dirancang dengan konfigurasi parameter yang berbeda, kemudian model dilatih dan diuji dengan prosedur yang sama untuk mengetahui pengaruh perubahan parameter tersebut terhadap performa klasifikasi. Dengan demikian, skenario pengujian tidak hanya berperan sebagai rancangan eksperimen, tetapi juga sebagai mekanisme untuk menemukan

konfigurasi model yang paling optimal dalam mendeteksi ujaran kebencian secara *Multi-Label*.

Parameter utama yang divariasikan dalam skenario ini mencakup dimensi embedding (d), jumlah unit pada lapisan LSTM (H), ukuran batch (*batch size*), serta nilai ambang batas (*threshold*, τ) yang digunakan pada tahap klasifikasi akhir. Variasi terhadap keempat parameter ini bertujuan untuk mengamati bagaimana kapasitas representasi kata, kompleksitas jaringan, ukuran batch, dan tingkat sensitivitas ambang keputusan memengaruhi kemampuan generalisasi model.

Parameter lain seperti *dropout*, *learning rate*, dan jumlah *epoch* dijaga konstan pada seluruh skenario, masing-masing sebesar 0.1, 0.001, dan 20 epoch. Penetapan nilai tetap pada parameter tersebut dilakukan agar pengujian terfokus pada efek perubahan empat parameter utama yang tercantum dalam tabel berikut.

Tabel 3.12 Konfigurasi Skenario Pengujian

Skenario	Embedding(d)	Unit LSTM	Batch Size	Thershold
1	50	64	32	0.1
2	50	64	32	0.2
3	50	64	32	0.3
4	50	64	64	0.1
5	50	64	64	0.2
6	50	64	64	0.3
7	50	128	32	0.1
8	50	128	32	0.2
9	50	128	32	0.3
10	50	128	64	0.1
11	50	128	64	0.2
12	50	128	64	0.3
13	100	64	32	0.1
14	100	64	32	0.2
15	100	64	32	0.3
16	100	64	64	0.1
17	100	64	64	0.2
18	100	64	64	0.3
19	100	128	32	0.1
20	100	128	32	0.2
21	100	128	32	0.3
22	100	128	64	0.1
23	100	128	64	0.2
24	100	128	64	0.3

Tabel 3.12 ini menunjukkan 24 kombinasi skenario yang dibangun dari dua nilai dimensi *embedding* (50 dan 100), dua kapasitas LSTM (64 dan 128 unit), dua ukuran *batch* (32 dan 64), serta tiga variasi nilai *threshold* (0.1, 0.2, dan 0.3).

Variasi nilai *threshold* dilakukan untuk mengamati pengaruh tingkat keputusan terhadap hasil klasifikasi *Multi-Label* berbasis fungsi aktivasi sigmoid. Secara umum, semakin tinggi nilai *threshold*, semakin ketat kriteria yang digunakan untuk menentukan label positif, sehingga cenderung meningkatkan *precision* namun menurunkan *recall*. Sebaliknya, *threshold* yang lebih rendah membuat model lebih sensitif dalam mendeteksi label positif, tetapi berpotensi meningkatkan kesalahan klasifikasi.

Dimensi *embedding* yang lebih tinggi (100) diharapkan mampu memberikan representasi semantik yang lebih kaya terhadap konteks kata, sedangkan jumlah unit LSTM yang lebih besar (128) memungkinkan model menangkap hubungan kontekstual antar kata secara lebih kompleks. Sementara itu, variasi *batch size* digunakan untuk menguji keseimbangan antara stabilitas pembaruan bobot dan kecepatan proses pelatihan.

Setiap skenario dilatih menggunakan 80% data latih dan dievaluasi pada 20% data uji. Seluruh hasil pengujian kemudian dinilai menggunakan metrik *Subset Accuracy, Hamming Loss, Precision, Recall,* dan *F1-score*. Dengan membandingkan hasil antar skenario, dapat ditentukan konfigurasi *hyperparameter* yang menghasilkan performa paling optimal dan stabil dalam mendeteksi ujaran kebencian berbahasa Indonesia secara *Multi-Label*.

BAB IV

HASIL DAN PEMBAHASAN

Pada penelitian ini, dilakukan serangkaian uji coba untuk mengevaluasi kinerja model klasifikasi *Multi-Label* dalam mendeteksi ujaran kebencian berbahasa Indonesia menggunakan model *Long Short-Term Memory* (LSTM) yang dipadukan dengan ekstraksi fitur melalui *Word2Vec*.

Uji coba dilakukan dengan berbagai variasi skenario yang melibatkan perubahan beberapa parameter utama dalam model, seperti dimensi embedding, jumlah unit LSTM, ukuran batch, serta nilai ambang batas (*threshold*). Setiap skenario bertujuan untuk mengamati pengaruh perubahan parameter terhadap performa model dalam mengklasifikasikan data uji. Model dilatih dengan 80% data latih dan diuji menggunakan 20% data uji, untuk kemudian dievaluasi dengan metrik evaluasi yang mencakup *Subset Accuracy*, *Hamming Loss*, *Precision*, *Recall*, dan *F1-score*.

4.1 Hasil Uji Coba

Uji coba dilakukan untuk mengukur performa model LSTM dalam mengklasifikasikan 6 label ujaran kebencian. Pengujian membandingkan 24 skenario *hyperparameter* yang berbeda, yang berfokus pada variasi dimensi *embedding* (d), unit LSTM (H), *batch size* (B), dan *threshold* (t), sebagaimana telah dirinci pada Tabel 3.12. Seluruh skenario menggunakan pembagian data latih 80% dan data uji 20%.

4.1.1 Hasil Skenario Uji 1

Skenario Uji 1 merupakan pengujian awal yang digunakan sebagai dasar untuk mengamati perilaku sistem dalam melakukan klasifikasi *Multi-Label* ujaran kebencian. Pada skenario ini digunakan *embedding* berukuran 50 dimensi, *LSTM* dengan 64 unit, *batch size* sebesar 32, serta nilai *threshold* 0,1. Kombinasi *hyperparameter* tersebut dirancang untuk menciptakan kondisi paling permisif dalam menetapkan label positif sehingga dapat menggambarkan kecenderungan awal sistem dalam menghasilkan prediksi. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.1.

Tabel 4.1 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 1

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1114	710	360	136	96	752	
TN	274	489	1153	2090	2323	594	
FP	1242	1427	1081	380	193	1279	
FN	4	8	40	28	22	9	
Matrix Evaluasi							
Precision (%)	47,28	33,22	24,98	26,36	33,22	37,03	33,68
Recall (%)	99,64	98,89	90,00	82,93	81,36	98,82	91,94
F1-Score (%)	64,13	49,74	39,11	40,00	47,17	53,87	49,00
Subset Accuracy (%)	14,92						
Hamming Loss (%)	36,15						

Berdasarkan Tabel 4.1, diperoleh nilai *subset accuracy* sebesar 14,92% dan *hamming loss* sebesar 36,15%. Nilai *subset accuracy* yang rendah menunjukkan bahwa sistem masih belum mampu memprediksi seluruh label secara tepat pada

satu data secara bersamaan. Sementara itu, nilai *hamming loss* yang relatif tinggi mengindikasikan bahwa tingkat kesalahan prediksi per label masih cukup besar.

Ditinjau dari metrik per label, sistem menunjukkan nilai *recall* yang sangat tinggi pada seluruh kategori, dengan *macro recall* mencapai 91,94%. Label *HS*, *HS_Individual*, dan *HS_Other* masing-masing memiliki nilai *recall* di atas 98%. Hal ini menunjukkan bahwa sistem hampir selalu berhasil mendeteksi keberadaan ujaran kebencian ketika label tersebut memang muncul pada data uji. Tingginya nilai *recall* ini selaras dengan penggunaan nilai *threshold* yang rendah, sehingga sistem cenderung mudah menetapkan label positif.

Namun demikian, capaian *recall* yang tinggi tersebut tidak diimbangi oleh nilai *precision* yang memadai. Berdasarkan Tabel 4.1, diperoleh nilai *macro precision* sebesar 33,68%, yang menunjukkan bahwa sebagian besar prediksi positif yang dihasilkan sistem tidak sepenuhnya tepat. Kondisi ini mengindikasikan bahwa sistem masih cenderung memberikan prediksi positif secara berlebihan, terutama pada label dengan distribusi data yang lebih kecil.

Ketidakseimbangan antara nilai *precision* dan *recall* tersebut berdampak langsung pada nilai *F1-score*. Berdasarkan Tabel 4.1, diperoleh nilai *macro F1-score* sebesar 49,00%. Label *HS* memiliki nilai *F1-score* tertinggi karena didukung oleh nilai *recall* yang sangat tinggi dan nilai *precision* yang relatif lebih baik dibandingkan label lainnya. Sebaliknya, beberapa label lain masih menunjukkan nilai *F1-score* yang rendah akibat rendahnya nilai *precision*.

Analisis *confusion matrix* pada Tabel 4.1 menunjukkan bahwa jumlah *false negative* pada seluruh label relatif kecil, yang menjelaskan tingginya nilai *recall*.

Sebaliknya, jumlah *false positive* tergolong besar pada hampir seluruh label, yang menjadi penyebab utama rendahnya nilai *precision* dan rendahnya nilai *subset accuracy*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 1 menunjukkan bahwa konfigurasi dengan nilai *threshold* rendah mampu meningkatkan sensitivitas sistem dalam mendeteksi ujaran kebencian, namun belum diikuti oleh kemampuan penyaringan prediksi yang baik. Kondisi ini menyebabkan banyaknya prediksi positif yang keliru dan performa klasifikasi yang belum optimal. Oleh karena itu, Skenario 1 digunakan sebagai acuan awal untuk melakukan penyesuaian *hyperparameter* pada skenario-skenario berikutnya guna memperoleh keseimbangan yang lebih baik antara nilai *precision* dan *recall*.

4.1.2 Hasil Skenario Uji 2

Skenario Uji 2 merupakan pengujian lanjutan dari Skenario 1 dengan mempertahankan konfigurasi *embedding* berukuran 50 dimensi, *LSTM* dengan 64 unit, dan *batch size* sebesar 32. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,2. Penyesuaian ini bertujuan untuk mengurangi kecenderungan sistem dalam menetapkan label positif secara berlebihan sekaligus mengamati perubahan keseimbangan antara nilai *precision* dan *recall*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.2.

Tabel 4.2 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 2

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1093	672	314	121	81	711	
TN	511	844	1625	2225	2409	959	

	Label						Metric
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
FP	1005	1072	609	245	107	914	
FN	25	46	86	43	37	50	
Matrix Evaluasi							
Precision (%)	52,10	38,53	34,02	33,06	43,09	43,75	40,76
Recall (%)	97,76	93,59	78,50	73,78	68,64	93,43	84,29
F1-Score (%)	67,97	54,59	47,47	45,66	52,94	59,60	54,70
Subset Accuracy (%)	29,54						
Hamming Loss (%)	26,82						

Berdasarkan Tabel 4.2, diperoleh nilai *subset accuracy* sebesar 29,54% dan *hamming loss* sebesar 26,82%. Dibandingkan dengan Skenario 1, nilai *subset accuracy* menunjukkan peningkatan yang cukup nyata, sementara nilai *hamming loss* mengalami penurunan. Hasil ini mengindikasikan bahwa peningkatan nilai *threshold* mulai memperbaiki kemampuan sistem dalam memprediksi kombinasi label secara tepat serta mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, sistem masih mempertahankan nilai *recall* yang relatif tinggi, dengan *macro recall* mencapai 84,29%. Label *HS* dan *HS_Other* masing-masing memiliki nilai *recall* sebesar 97,76% dan 93,43%, yang menunjukkan bahwa sebagian besar ujaran kebencian tetap dapat terdeteksi. Namun, jika dibandingkan dengan Skenario 1, nilai *recall* pada beberapa label mengalami penurunan, yang menandakan meningkatnya selektivitas sistem dalam menetapkan label positif.

Seiring dengan penurunan nilai *recall*, nilai *precision* pada Skenario 2 mengalami peningkatan. Berdasarkan Tabel 4.2, diperoleh nilai *macro precision* sebesar 40,76%, lebih tinggi dibandingkan Skenario 1. Peningkatan ini terlihat pada hampir seluruh label, yang menunjukkan bahwa sistem mulai mampu mengurangi jumlah prediksi positif yang tidak tepat (*false positive*).

Perbaikan keseimbangan antara nilai *precision* dan *recall* tersebut berdampak langsung pada nilai *F1-score*. Berdasarkan Tabel 4.2, diperoleh nilai *macro F1-score* sebesar 54,70%, yang menunjukkan peningkatan dibandingkan Skenario 1. Peningkatan nilai *F1-score* ini menandakan bahwa kualitas prediksi sistem secara keseluruhan menjadi lebih stabil dan seimbang.

Analisis *confusion matrix* pada Tabel 4.2 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 1, sementara jumlah *false negative* meningkat secara moderat. Pola ini menunjukkan bahwa sistem mulai mengurangi kesalahan prediksi positif, meskipun dengan konsekuensi meningkatnya sebagian kesalahan prediksi negatif.

Secara keseluruhan, hasil pengujian pada Skenario Uji 2 menunjukkan bahwa peningkatan nilai *threshold* dari 0,1 menjadi 0,2 mampu memperbaiki keseimbangan antara sensitivitas dan ketepatan prediksi. Sistem tidak lagi terlalu permisif dalam menetapkan label positif, sehingga nilai *subset accuracy* meningkat dan nilai *hamming loss* menurun. Dengan demikian, Skenario 2 menunjukkan perbaikan performa dibandingkan Skenario 1 dan menjadi pijakan yang lebih baik untuk pengujian pada skenario-skenario berikutnya.

4.1.3 Hasil Skenario Uji 3

Skenario Uji 3 melanjutkan pengujian dengan konfigurasi *embedding* berukuran 50 dimensi, *LSTM* dengan 64 unit, dan *batch size* sebesar 32, sama seperti Skenario 1 dan 2. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,3. Penyesuaian ini dilakukan untuk meningkatkan selektivitas sistem dalam menetapkan label positif sehingga diharapkan dapat menekan kesalahan prediksi positif. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.3.

Tabel 4.3 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 3

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1026	587	211	62	81	629	
TN	857	1281	2029	2434	2466	1354	
FP	659	635	205	36	50	519	
FN	92	131	189	102	37	132	
Matrix Evaluasi							
Precision (%)	60,89	48,04	50,72	63,27	61,83	54,79	56,59
Recall (%)	91,77	81,75	52,75	37,80	68,64	82,65	69,23
F1-Score (%)	73,21	60,52	51,72	47,33	65,06	65,90	60,62
Subset Accuracy (%)	50,46						
Hamming Loss (%)	17,63						

Berdasarkan Tabel 4.3, diperoleh nilai *subset accuracy* sebesar 50,46% dan *hamming loss* sebesar 17,63%. Dibandingkan dengan Skenario 2, nilai *subset accuracy* meningkat secara signifikan, sementara nilai *hamming loss* menurun. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* mampu memperbaiki

ketepatan prediksi kombinasi label sekaligus mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, peningkatan nilai *threshold* menghasilkan perubahan keseimbangan antara nilai *precision* dan *recall*. Berdasarkan Tabel 4.3, diperoleh nilai *macro precision* sebesar 56,59% dan *macro recall* sebesar 69,23%. Peningkatan nilai *precision* menunjukkan berkurangnya jumlah prediksi positif yang tidak tepat. Namun, penurunan nilai *recall* pada beberapa label mengindikasikan bahwa sistem menjadi lebih selektif sehingga sebagian data positif tidak lagi terdeteksi.

Pada label *HS*, sistem masih mampu mempertahankan nilai *recall* yang tinggi sebesar 91,77% dengan nilai *precision* sebesar 60,89%, sehingga menghasilkan nilai *F1-score* tertinggi sebesar 73,21%. Hal ini menunjukkan bahwa kategori ujaran kebencian utama tetap dapat dikenali dengan baik meskipun nilai *threshold* dinaikkan. Sebaliknya, label *HS_Religion* menunjukkan penurunan nilai *recall* yang cukup besar, yaitu 37,80%, meskipun nilai *precision*-nya relatif tinggi sebesar 63,27%. Kondisi ini menandakan meningkatnya selektivitas sistem pada kategori tersebut.

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut tercermin pada nilai *F1-score*. Berdasarkan Tabel 4.3, diperoleh nilai *macro F1-score* sebesar 60,62%, yang lebih tinggi dibandingkan Skenario 1 dan 2. Peningkatan ini menunjukkan bahwa kualitas prediksi sistem secara keseluruhan menjadi lebih seimbang meskipun sensitivitas pada beberapa label berkurang.

Analisis *confusion matrix* pada Tabel 4.3 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan skenario dengan nilai *threshold* yang lebih rendah. Sebaliknya, jumlah *false negative* meningkat pada beberapa label, yang menunjukkan bahwa sebagian data positif tidak terdeteksi akibat peningkatan nilai *threshold*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 3 menunjukkan bahwa peningkatan nilai *threshold* menjadi 0,3 mampu meningkatkan selektivitas sistem dan menekan kesalahan prediksi positif. Dampak ini tercermin pada peningkatan nilai *precision*, *F1-score*, dan *subset accuracy*, serta penurunan nilai *hamming loss*. Namun, peningkatan selektivitas tersebut juga menyebabkan penurunan nilai *recall* pada beberapa kategori. Dengan demikian, Skenario 3 menegaskan adanya *trade-off* antara sensitivitas dan ketepatan prediksi dalam klasifikasi *Multi-Label*, serta menjadi pijakan penting untuk evaluasi konfigurasi pada skenario berikutnya.

4.1.4 Hasil Skenario Uji 4

Skenario Uji 4 merupakan pengujian lanjutan yang bertujuan untuk mengamati pengaruh perubahan *batch size* terhadap hasil klasifikasi *Multi-Label*. Pada skenario ini digunakan *embedding* berukuran 50 dimensi, *LSTM* dengan 64 unit, nilai *threshold* sebesar 0,1, serta *batch size* yang ditingkatkan menjadi 64. Perubahan *batch size* ini dilakukan untuk mengevaluasi kestabilan proses pembelajaran model ketika jumlah data yang diproses dalam satu iterasi diperbesar. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.4.

Tabel 4.4 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 4

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1112	708	354	133	71	746	
TN	293	481	1138	2181	2412	605	
FP	1223	1435	1096	289	104	1268	
FN	6	10	46	31	47	15	
Matrix Evaluasi							
Precision (%)	47,62	33,04	24,41	31,52	40,57	37,04	35,70
Recall (%)	99,46	98,61	88,50	81,10	60,17	98,03	87,64
F1-Score (%)	64,41	49,49	38,27	45,39	48,46	53,77	49,97
Subset Accuracy (%)	16,36						
Hamming Loss (%)	35,24						

Berdasarkan Tabel 4.4, diperoleh nilai *subset accuracy* sebesar 16,36% dan *hamming loss* sebesar 35,24%. Jika dibandingkan dengan Skenario 1 yang menggunakan *batch size* lebih kecil, nilai *subset accuracy* hanya mengalami peningkatan yang sangat terbatas, sementara nilai *hamming loss* masih tergolong tinggi. Hal ini menunjukkan bahwa peningkatan *batch size* saja belum mampu memperbaiki ketepatan prediksi kombinasi label secara signifikan.

Ditinjau dari metrik per label, sistem kembali menunjukkan nilai *recall* yang sangat tinggi, dengan *macro recall* mencapai 87,64%. Label *HS* dan *HS_Individual* masing-masing memiliki nilai *recall* sebesar 99,46% dan 98,61%, yang menunjukkan bahwa sistem hampir selalu berhasil mendeteksi keberadaan ujaran kebencian. Tingginya nilai *recall* ini selaras dengan penggunaan nilai *threshold*

yang rendah, sehingga sistem tetap bersifat permisif dalam menetapkan label positif.

Namun demikian, nilai *precision* pada Skenario 4 masih relatif rendah. Berdasarkan Tabel 4.4, diperoleh nilai *macro precision* sebesar 35,70%, yang menunjukkan bahwa sistem masih menghasilkan banyak prediksi positif yang tidak tepat. Kondisi ini terlihat pada label *HS_Group* dan *HS_Religion* yang memiliki nilai *precision* rendah meskipun nilai *recall*-nya tinggi. Hal ini mengindikasikan bahwa peningkatan *batch size* belum mampu mengurangi kecenderungan sistem dalam menghasilkan *false positive*.

Ketidakseimbangan antara nilai *precision* dan *recall* tersebut berdampak pada nilai *F1-score*. Berdasarkan Tabel 4.4, diperoleh nilai *macro F1-score* sebesar 49,97%, yang relatif tidak jauh berbeda dibandingkan Skenario 1. Hal ini menunjukkan bahwa meskipun terdapat perubahan pada *batch size*, kualitas prediksi secara keseluruhan belum mengalami peningkatan yang berarti.

Analisis *confusion matrix* pada Tabel 4.4 memperkuat temuan tersebut. Jumlah *false negative* pada seluruh label masih relatif kecil, yang menjelaskan tingginya nilai *recall*. Sebaliknya, jumlah *false positive* tetap tinggi pada hampir seluruh label, yang menjadi penyebab utama rendahnya nilai *precision* dan rendahnya nilai *subset accuracy*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 4 menunjukkan bahwa peningkatan *batch size* dari 32 menjadi 64 pada nilai *threshold* yang rendah belum memberikan dampak signifikan terhadap peningkatan performa klasifikasi *Multi-Label*. Sistem masih bersifat sangat permisif dalam menetapkan label positif,

sehingga menghasilkan banyak prediksi positif yang keliru. Dengan demikian, Skenario 4 menunjukkan bahwa penyesuaian *batch size* perlu dikombinasikan dengan pengaturan *threshold* atau *hyperparameter* lain agar diperoleh keseimbangan yang lebih baik antara nilai *precision* dan *recall* pada skenario-skenario berikutnya.

4.1.5 Hasil Skenario Uji 5

Skenario Uji 5 merupakan pengujian lanjutan dengan mempertahankan *embedding* berukuran 50 dimensi dan *LSTM* dengan 64 unit, serta *batch size* sebesar 64 seperti pada Skenario 4. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,2. Penyesuaian ini dilakukan untuk mengurangi sifat permisif sistem dalam menetapkan label positif sekaligus mengevaluasi dampaknya terhadap keseimbangan nilai *precision* dan *recall*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.5.

Tabel 4.5 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 5

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1083	643	264	90	58	685	
TN	523	931	1787	2318	2418	1018	
FP	993	985	447	152	98	855	
FN	35	75	136	74	60	76	
Matrix Evaluasi							
Precision (%)	52,17	39,50	37,13	37,19	37,18	44,48	41,27
Recall (%)	96,87	89,55	66,00	54,88	49,15	90,01	74,41
F1-Score (%)	67,81	54,82	47,52	44,33	42,34	59,54	52,73
Subset Accuracy (%)	32,19						
Hamming Loss	25,22						

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
(%)							

Berdasarkan Tabel 4.5, diperoleh nilai *subset accuracy* sebesar 32,19% dan *hamming loss* sebesar 25,22%. Dibandingkan dengan Skenario 4, nilai *subset accuracy* meningkat secara nyata, sementara nilai *hamming loss* menurun. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada *batch size* yang lebih besar mulai memperbaiki ketepatan prediksi kombinasi label dan mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, sistem menunjukkan penurunan nilai *recall* dibandingkan Skenario 4, dengan *macro recall* sebesar 74,41%. Meskipun demikian, sebagian besar label utama masih mempertahankan nilai *recall* yang relatif baik, seperti *HS* sebesar 96,87% dan *HS_Other* sebesar 90,01%. Penurunan *recall* ini mencerminkan meningkatnya selektivitas sistem dalam menetapkan label positif seiring dengan kenaikan nilai *threshold*.

Seiring dengan penurunan *recall*, nilai *precision* pada Skenario 5 mengalami peningkatan. Berdasarkan Tabel 4.5, diperoleh nilai *macro precision* sebesar 41,27%, lebih tinggi dibandingkan Skenario 4. Peningkatan ini terlihat pada hampir seluruh label, yang menunjukkan bahwa sistem mulai mampu menekan jumlah prediksi positif yang tidak tepat (*false positive*), khususnya pada kategori dengan distribusi data yang lebih kecil.

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut berdampak langsung pada nilai *F1-score*. Berdasarkan Tabel 4.5, diperoleh nilai *macro F1-score* sebesar 52,73%, yang menunjukkan peningkatan dibandingkan

Skenario 4. Label *HS* dan *HS_Other* memperoleh nilai *F1-score* tertinggi karena masih mempertahankan nilai *recall* yang tinggi dengan nilai *precision* yang lebih baik.

Analisis *confusion matrix* pada Tabel 4.5 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 4, sementara jumlah *false negative* meningkat secara moderat. Pola ini menunjukkan bahwa peningkatan nilai *threshold* pada *batch size* yang lebih besar membantu sistem mengurangi kesalahan prediksi positif, meskipun dengan konsekuensi berkurangnya sensitivitas pada beberapa label.

Secara keseluruhan, hasil pengujian pada Skenario Uji 5 menunjukkan bahwa kombinasi *batch size* yang lebih besar dengan nilai *threshold* menengah mampu meningkatkan keseimbangan antara ketepatan dan sensitivitas prediksi. Peningkatan nilai *subset accuracy* dan penurunan *hamming loss* menandakan perbaikan performa klasifikasi *Multi-Label* dibandingkan Skenario 4. Dengan demikian, Skenario 5 menunjukkan bahwa penyesuaian nilai *threshold* pada *batch size* yang lebih besar memberikan dampak positif terhadap kualitas prediksi dan menjadi pijakan yang lebih baik untuk pengujian pada skenario berikutnya.

4.1.6 Hasil Skenario Uji 6

Skenario Uji 6 merupakan pengujian lanjutan dengan mempertahankan *embedding* berukuran 50 dimensi, *LSTM* dengan 64 unit, dan *batch size* sebesar 64 seperti pada Skenario 4 dan 5. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,3. Penyesuaian ini dilakukan untuk meningkatkan selektivitas sistem dalam menetapkan label positif dan mengevaluasi

dampaknya terhadap ketepatan prediksi klasifikasi *Multi-Label*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.6.

Tabel 4.6 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 6

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	986	514	192	82	42	596	
TN	905	1412	2027	2355	2455	1417	
FP	611	504	207	115	61	456	
FN	132	204	208	82	76	165	
Matrix Evaluasi							
Precision (%)	61,74	50,49	48,12	41,62	40,78	56,65	49,90
Recall (%)	88,19	71,59	48,00	50,00	35,59	78,32	61,95
F1-Score (%)	72,63	59,22	48,06	45,43	38,01	65,75	54,85
Subset Accuracy (%)	48,52						
Hamming Loss (%)	17,85						

Berdasarkan Tabel 4.6, diperoleh nilai *subset accuracy* sebesar 48,52% dan *hamming loss* sebesar 17,85%. Dibandingkan dengan Skenario 5, nilai *subset accuracy* meningkat secara signifikan, sementara nilai *hamming loss* menurun. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada *batch size* yang lebih besar mampu memperbaiki ketepatan prediksi kombinasi label sekaligus mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, sistem menunjukkan keseimbangan yang lebih baik antara nilai *precision* dan *recall*. Berdasarkan Tabel 4.6, diperoleh nilai *macro precision* sebesar 49,90% dan *macro recall* sebesar 61,95%. Peningkatan nilai *precision* menandakan berkurangnya jumlah prediksi positif yang tidak tepat

(*false positive*), sedangkan penurunan nilai *recall* menunjukkan meningkatnya selektivitas sistem dalam menetapkan label positif.

Pada label *HS*, sistem masih mampu mempertahankan nilai *recall* yang cukup tinggi sebesar 88,19% dengan nilai *precision* sebesar 61,74%, sehingga menghasilkan nilai *F1-score* sebesar 72,63%. Hal ini menunjukkan bahwa kategori ujaran kebencian utama tetap dapat dikenali dengan baik meskipun nilai *threshold* dinaikkan. Sebaliknya, pada label *HS_Race* dan *HS_Group*, nilai *recall* relatif rendah, masing-masing sebesar 35,59% dan 48,00%, yang mengindikasikan bahwa sebagian data positif pada kategori tersebut tidak terdeteksi akibat meningkatnya selektivitas sistem.

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut tercermin pada nilai *F1-score*. Berdasarkan Tabel 4.6, diperoleh nilai *macro F1-score* sebesar 54,85%, yang menunjukkan peningkatan dibandingkan Skenario 5. Peningkatan ini menandakan bahwa secara keseluruhan kualitas prediksi menjadi lebih stabil dan seimbang meskipun sensitivitas pada beberapa label menurun.

Analisis *confusion matrix* pada Tabel 4.6 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 5, yang menunjukkan berkurangnya prediksi positif yang keliru. Di sisi lain, jumlah *false negative* meningkat pada beberapa label, yang menandakan bahwa sebagian data positif tidak terdeteksi akibat peningkatan nilai *threshold*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 6 menunjukkan bahwa kombinasi *batch size* yang lebih besar dengan nilai *threshold* tinggi mampu

meningkatkan ketepatan prediksi dan menekan kesalahan prediksi positif secara signifikan. Peningkatan nilai *subset accuracy* dan penurunan *hamming loss* menegaskan adanya perbaikan performa klasifikasi *Multi-Label* dibandingkan Skenario 5. Dengan demikian, Skenario 6 menunjukkan konfigurasi yang lebih seimbang antara ketepatan dan sensitivitas prediksi serta menjadi pijakan yang kuat untuk pengujian pada skenario-skenario berikutnya.

4.1.7 Hasil Skenario Uji 7

Skenario Uji 7 merupakan pengujian lanjutan yang bertujuan untuk mengamati pengaruh peningkatan jumlah unit *LSTM* terhadap hasil klasifikasi *Multi-Label*. Pada skenario ini digunakan *embedding* berukuran 50 dimensi, jumlah unit *LSTM* ditingkatkan menjadi 128, *batch size* sebesar 32, serta nilai *threshold* 0,1. Perubahan jumlah unit *LSTM* ini dilakukan untuk meningkatkan kapasitas representasi temporal, sementara nilai *threshold* yang rendah dipertahankan untuk mengamati perilaku sistem pada kondisi yang masih permisif. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.7.

Tabel 4.7 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 7

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1111	705	354	127	88	741	
TN	299	474	1220	2145	2327	611	
FP	1217	1442	1014	325	189	1262	
FN	7	13	46	37	30	20	
Matrix Evaluasi							
Precision (%)	47,72	32,84	25,88	28,10	31,77	36,99	33,88
Recall (%)	99,37	98,19	88,50	77,44	74,58	97,37	89,24
F1-Score (%)	64,48	49,21	40,05	41,23	44,56	53,62	48,86

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Subset Accuracy (%)	16,32						
Hamming Loss (%)	35,45						

Berdasarkan Tabel 4.7, diperoleh nilai *subset accuracy* sebesar 16,32% dan *hamming loss* sebesar 35,45%. Nilai *subset accuracy* yang rendah menunjukkan bahwa peningkatan jumlah unit *LSTM* tanpa penyesuaian *threshold* belum mampu memperbaiki ketepatan prediksi kombinasi label. Sementara itu, nilai *hamming loss* yang tinggi mengindikasikan bahwa kesalahan prediksi per label masih sering terjadi.

Ditinjau dari metrik per label, sistem kembali menunjukkan nilai *recall* yang sangat tinggi, dengan *macro recall* mencapai 89,24%. Label *HS* dan *HS_Individual* masing-masing memiliki nilai *recall* sebesar 99,37% dan 98,19%, yang menunjukkan bahwa sistem hampir selalu berhasil mendeteksi keberadaan ujaran kebencian. Tingginya nilai *recall* ini konsisten dengan penggunaan nilai *threshold* yang rendah, sehingga sistem cenderung menetapkan label positif secara luas.

Namun demikian, nilai *precision* pada Skenario 7 masih relatif rendah. Berdasarkan Tabel 4.7, diperoleh nilai *macro precision* sebesar 33,88%, yang menunjukkan bahwa peningkatan jumlah unit *LSTM* belum diikuti oleh kemampuan sistem dalam menyaring prediksi positif secara akurat. Kondisi ini terlihat pada label *HS_Group* dan *HS_Religion* yang memiliki nilai *precision* rendah meskipun nilai *recall*-nya tinggi.

Ketidakseimbangan antara nilai *precision* dan *recall* tersebut berdampak pada nilai *F1-score*. Berdasarkan Tabel 4.7, diperoleh nilai *macro F1-score* sebesar 48,86%, yang relatif tidak berbeda dibandingkan skenario dengan jumlah unit *LSTM* lebih kecil pada *threshold* yang sama. Hal ini menunjukkan bahwa peningkatan kapasitas model belum memberikan peningkatan performa yang signifikan tanpa penyesuaian parameter pengambilan keputusan.

Analisis *confusion matrix* pada Tabel 4.7 memperkuat temuan tersebut. Jumlah *false negative* pada seluruh label relatif kecil, yang menjelaskan tingginya nilai *recall*. Sebaliknya, jumlah *false positive* tetap tinggi pada hampir seluruh label, yang menjadi penyebab utama rendahnya nilai *precision* dan rendahnya nilai *subset accuracy*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 7 menunjukkan bahwa peningkatan jumlah unit *LSTM* menjadi 128 pada nilai *threshold* yang rendah belum mampu memperbaiki performa klasifikasi *Multi-Label* secara signifikan. Sistem masih bersifat sangat permisif dalam menetapkan label positif, sehingga menghasilkan banyak prediksi positif yang keliru. Dengan demikian, Skenario 7 menunjukkan bahwa peningkatan kapasitas model perlu diimbangi dengan penyesuaian nilai *threshold* atau *hyperparameter* lainnya agar diperoleh keseimbangan yang lebih baik antara nilai *precision* dan *recall*.

4.1.8 Hasil Skenario Uji 8

Skenario Uji 8 merupakan pengujian lanjutan dari Skenario 7 dengan mempertahankan *embedding* berukuran 50 dimensi, *LSTM* dengan 128 unit, dan *batch size* sebesar 32. Perbedaan utama pada skenario ini terletak pada peningkatan

nilai *threshold* menjadi 0,2. Penyesuaian ini dilakukan untuk mengurangi sifat permisif sistem dalam menetapkan label positif yang masih terlihat pada Skenario 7, serta untuk mengevaluasi dampaknya terhadap keseimbangan nilai *precision* dan *recall*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.8.

Tabel 4.8 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 8

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1093	664	296	106	89	698	
TN	492	867	1703	2281	2442	1027	
FP	1024	1049	531	189	74	846	
FN	25	54	104	58	29	63	
Matrix Evaluasi							
Precision (%)	51,63	38,76	35,79	35,93	54,60	45,21	43,65
Recall (%)	97,76	92,48	74,00	64,63	75,42	91,72	82,67
F1-Score (%)	67,57	54,63	48,25	46,19	63,35	60,56	56,76
Subset Accuracy (%)	31,55						
Hamming Loss (%)	25,60						

Berdasarkan Tabel 4.8, diperoleh nilai *subset accuracy* sebesar 31,55% dan *hamming loss* sebesar 25,60%. Dibandingkan dengan Skenario 7, nilai *subset accuracy* meningkat secara nyata, sementara nilai *hamming loss* menurun. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* mulai memperbaiki ketepatan prediksi kombinasi label serta mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, sistem masih mempertahankan nilai *recall* yang relatif tinggi, dengan *macro recall* mencapai 82,67%. Label *HS* dan *HS_Other* masing-masing memiliki nilai *recall* sebesar 97,76% dan 91,72%, yang

menunjukkan bahwa sebagian besar ujaran kebencian masih dapat terdeteksi dengan baik. Meskipun demikian, nilai *recall* pada beberapa label, seperti *HS_Group* dan *HS_Religion*, mengalami penurunan dibandingkan Skenario 7, yang menandakan meningkatnya selektivitas sistem dalam menetapkan label positif.

Seiring dengan penurunan nilai *recall*, nilai *precision* pada Skenario 8 mengalami peningkatan. Berdasarkan Tabel 4.8, diperoleh nilai *macro precision* sebesar 43,65%, yang lebih tinggi dibandingkan Skenario 7. Peningkatan ini terlihat pada hampir seluruh label, khususnya pada *HS_Race* yang mencapai nilai *precision* sebesar 54,60%. Kondisi ini menunjukkan bahwa sistem mulai mampu menekan jumlah prediksi positif yang tidak tepat (*false positive*).

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut berdampak langsung pada nilai *F1-score*. Berdasarkan Tabel 4.8, diperoleh nilai *macro F1-score* sebesar 56,76%, yang menunjukkan peningkatan dibandingkan Skenario 7. Peningkatan ini menandakan bahwa kualitas prediksi sistem secara keseluruhan menjadi lebih stabil dan seimbang setelah penyesuaian nilai *threshold*.

Analisis *confusion matrix* pada Tabel 4.8 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 7, sementara jumlah *false negative* meningkat secara moderat. Pola ini menunjukkan bahwa sistem mulai mengurangi kesalahan prediksi positif dengan konsekuensi berkurangnya sensitivitas pada sebagian kategori.

Secara keseluruhan, hasil pengujian pada Skenario Uji 8 menunjukkan bahwa peningkatan nilai *threshold* menjadi 0,2 pada konfigurasi *LSTM* dengan 128 unit mampu memperbaiki keseimbangan antara nilai *precision* dan *recall*.

Peningkatan nilai *subset accuracy* dan penurunan *hamming loss* menegaskan adanya perbaikan performa klasifikasi *Multi-Label* dibandingkan Skenario 7. Dengan demikian, Skenario 8 menunjukkan bahwa penyesuaian nilai *threshold* berperan penting dalam memaksimalkan manfaat peningkatan kapasitas model terhadap kualitas prediksi.

4.1.9 Hasil Skenario Uji 9

Skenario Uji 9 merupakan kelanjutan dari Skenario 7 dan 8 dengan mempertahankan *embedding* berukuran 50 dimensi, *LSTM* dengan 128 unit, serta *batch size* sebesar 32. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,3. Penyesuaian ini dilakukan untuk meningkatkan selektivitas sistem dalam menetapkan label positif setelah peningkatan kapasitas model pada skenario sebelumnya. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.9.

Tabel 4.9 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 9

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1028	567	168	69	67	622	
TN	792	1283	2053	2353	2432	1308	
FP	724	633	181	117	84	565	
FN	90	151	232	95	51	139	
Matrix Evaluasi							
Precision (%)	58,68	47,25	48,14	37,10	44,37	52,40	47,99
Recall (%)	91,95	78,97	42,00	42,07	56,78	81,73	65,58
F1-Score (%)	71,64	59,12	44,86	39,43	49,81	63,86	54,79
Subset Accuracy (%)	46,92						
Hamming Loss	19,37						

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
(%)							

Berdasarkan Tabel 4.9, diperoleh nilai *subset accuracy* sebesar 46,92% dan *hamming loss* sebesar 19,37%. Dibandingkan dengan Skenario 8, nilai *subset accuracy* meningkat secara signifikan, sementara nilai *hamming loss* mengalami penurunan. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada konfigurasi *LSTM* dengan kapasitas lebih besar mampu memperbaiki ketepatan prediksi kombinasi label sekaligus mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, peningkatan nilai *threshold* kembali memengaruhi keseimbangan antara nilai *precision* dan *recall*. Berdasarkan Tabel 4.9, diperoleh nilai *macro precision* sebesar 47,99% dan *macro recall* sebesar 65,58%. Peningkatan nilai *precision* dibandingkan Skenario 8 menunjukkan berkurangnya jumlah prediksi positif yang tidak tepat. Namun, nilai *recall* pada beberapa label mengalami penurunan, yang mengindikasikan meningkatnya selektivitas sistem dalam menetapkan label positif.

Pada label *HS*, sistem masih mampu mempertahankan nilai *recall* yang tinggi sebesar 91,95% dengan nilai *precision* sebesar 58,68%, sehingga menghasilkan nilai *F1-score* sebesar 71,64%. Hal ini menunjukkan bahwa kategori ujaran kebencian utama tetap dapat dikenali dengan baik meskipun nilai *threshold* dinaikkan. Sebaliknya, pada label *HS_Group* dan *HS_Religion*, nilai *recall* relatif rendah, masing-masing sebesar 42,00% dan 42,07%, yang menunjukkan bahwa sebagian data positif pada kategori tersebut tidak terdeteksi akibat meningkatnya selektivitas sistem.

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut tercermin pada nilai *F1-score*. Berdasarkan Tabel 4.9, diperoleh nilai *macro F1-score* sebesar 54,79%. Nilai ini relatif stabil dibandingkan Skenario 8, yang menunjukkan bahwa peningkatan selektivitas sistem mampu menjaga kualitas prediksi secara keseluruhan meskipun sensitivitas pada beberapa label menurun.

Analisis *confusion matrix* pada Tabel 4.9 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 8, yang menandakan berkurangnya prediksi positif yang keliru. Di sisi lain, jumlah *false negative* meningkat pada beberapa label, yang menunjukkan bahwa sebagian data positif tidak terdeteksi akibat peningkatan nilai *threshold*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 9 menunjukkan bahwa peningkatan nilai *threshold* menjadi 0,3 pada konfigurasi *LSTM* dengan 128 unit dan *batch size* 32 mampu meningkatkan ketepatan prediksi dan mengurangi kesalahan prediksi positif. Dampak ini tercermin pada peningkatan nilai *subset accuracy* dan penurunan nilai *hamming loss*. Namun, peningkatan selektivitas tersebut juga menyebabkan penurunan nilai *recall* pada beberapa kategori. Dengan demikian, Skenario 9 menegaskan adanya *trade-off* antara sensitivitas dan ketepatan prediksi dalam klasifikasi *Multi-Label* serta menjadi pijakan penting untuk evaluasi konfigurasi pada skenario berikutnya.

4.1.10 Hasil Skenario Uji 10

Skenario Uji 10 merupakan pengujian lanjutan yang bertujuan untuk mengamati pengaruh peningkatan *batch size* pada konfigurasi *LSTM* dengan 128

unit. Pada skenario ini digunakan *embedding* berukuran 50 dimensi, *LSTM* dengan 128 unit, *batch size* sebesar 64, serta nilai *threshold* 0,1. Perubahan *batch size* dilakukan untuk mengevaluasi kestabilan proses pembelajaran ketika jumlah data dalam satu iterasi diperbesar, sementara nilai *threshold* yang rendah dipertahankan untuk melihat perilaku sistem pada kondisi yang masih permisif. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.10.

Tabel 4.10 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 10

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1114	708	342	129	90	749	
TN	277	491	1378	2136	2349	608	
FP	1239	1425	856	334	167	1265	
FN	4	10	58	35	28	12	
Matrix Evaluasi							
Precision (%)	47,34	33,19	28,55	27,86	35,02	37,19	34,86
Recall (%)	99,64	98,61	85,50	78,66	76,27	98,42	89,52
F1-Score (%)	64,19	49,67	42,80	41,15	48,00	53,98	49,96
Subset Accuracy (%)	18,03						
Hamming Loss (%)	34,38						

Berdasarkan Tabel 4.10, diperoleh nilai *subset accuracy* sebesar 18,03% dan *hamming loss* sebesar 34,38%. Jika dibandingkan dengan Skenario 9, nilai *subset accuracy* menurun cukup tajam, sementara nilai *hamming loss* kembali meningkat. Hasil ini menunjukkan bahwa peningkatan *batch size* tanpa penyesuaian nilai *threshold* belum mampu meningkatkan ketepatan prediksi kombinasi label.

Ditinjau dari metrik per label, sistem kembali menunjukkan nilai *recall* yang sangat tinggi, dengan *macro recall* mencapai 89,52%. Label *HS* dan *HS_Individual* masing-masing memiliki nilai *recall* sebesar 99,64% dan 98,61%, yang menunjukkan bahwa sistem hampir selalu berhasil mendeteksi keberadaan ujaran kebencian. Tingginya nilai *recall* ini konsisten dengan penggunaan nilai *threshold* yang rendah, sehingga sistem cenderung menetapkan label positif secara luas.

Namun demikian, nilai *precision* pada Skenario 10 masih relatif rendah. Berdasarkan Tabel 4.10, diperoleh nilai *macro precision* sebesar 34,86%, yang menunjukkan bahwa sistem masih menghasilkan banyak prediksi positif yang tidak tepat. Kondisi ini terlihat pada label *HS_Group* dan *HS_Religion* yang memiliki nilai *precision* rendah meskipun nilai *recall*-nya tinggi. Hal ini mengindikasikan bahwa peningkatan *batch size* belum diikuti oleh kemampuan sistem dalam menyaring prediksi positif secara lebih akurat.

Ketidakseimbangan antara nilai *precision* dan *recall* tersebut berdampak pada nilai *F1-score*. Berdasarkan Tabel 4.10, diperoleh nilai *macro F1-score* sebesar 49,96%, yang relatif serupa dengan skenario pada kondisi permisif sebelumnya. Hal ini menunjukkan bahwa perubahan *batch size* saja belum memberikan peningkatan performa yang berarti tanpa penyesuaian parameter pengambilan keputusan.

Analisis *confusion matrix* pada Tabel 4.10 memperkuat temuan tersebut. Jumlah *false negative* pada seluruh label relatif kecil, yang menjelaskan tingginya nilai *recall*. Sebaliknya, jumlah *false positive* tetap tinggi pada hampir seluruh label,

yang menjadi penyebab utama rendahnya nilai *precision* dan rendahnya nilai *subset accuracy*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 10 menunjukkan bahwa peningkatan *batch size* menjadi 64 pada konfigurasi *LSTM* dengan 128 unit dan nilai *threshold* yang rendah belum mampu memperbaiki performa klasifikasi *Multi-Label*. Sistem masih bersifat sangat permisif dalam menetapkan label positif, sehingga menghasilkan banyak prediksi positif yang keliru. Dengan demikian, Skenario 10 menegaskan bahwa penyesuaian *batch size* perlu diimbangi dengan pengaturan nilai *threshold* atau *hyperparameter* lain agar diperoleh keseimbangan yang lebih baik antara nilai *precision* dan *recall* pada skenario-skenario berikutnya.

4.1.11 Hasil Skenario Uji 11

Skenario Uji 11 merupakan pengujian lanjutan dari Skenario 10 dengan mempertahankan *embedding* berukuran 50 dimensi, *LSTM* dengan 128 unit, serta *batch size* sebesar 64. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,2. Penyesuaian ini dilakukan untuk mengurangi sifat permisif sistem yang masih terlihat pada Skenario 10 serta untuk mengevaluasi dampaknya terhadap keseimbangan antara nilai *precision* dan *recall*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.11.

Tabel 4.11 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 11

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1064	619	292	115	84	644	
TN	659	1062	1766	2228	2423	1239	
FP	857	854	468	242	93	634	
FN	54	99	108	49	34	117	

	Label						Metric
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Matrix Evaluasi							
Precision (%)	55,39	42,02	38,42	32,21	47,46	50,39	44,32
Recall (%)	95,17	86,21	73,00	70,12	71,19	84,63	80,05
F1-Score (%)	70,02	56,50	50,34	44,15	56,95	63,17	56,86
Subset Accuracy (%)	35,91						
Hamming Loss (%)	22,84						

Berdasarkan Tabel 4.11, diperoleh nilai *subset accuracy* sebesar 35,91% dan *hamming loss* sebesar 22,84%. Dibandingkan dengan Skenario 10, nilai *subset accuracy* meningkat secara nyata, sementara nilai *hamming loss* mengalami penurunan. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* mulai memperbaiki ketepatan prediksi kombinasi label serta mengurangi kesalahan prediksi per label pada konfigurasi *batch size* yang lebih besar.

Ditinjau dari metrik per label, sistem masih mempertahankan nilai *recall* yang relatif tinggi dengan *macro recall* sebesar 80,05%. Label *HS* memiliki nilai *recall* sebesar 95,17%, yang menunjukkan bahwa sebagian besar ujaran kebencian utama masih dapat terdeteksi dengan baik. Namun, jika dibandingkan dengan Skenario 10, nilai *recall* pada beberapa label mengalami penurunan, yang mencerminkan meningkatnya selektivitas sistem dalam menetapkan label positif.

Seiring dengan penurunan nilai *recall*, nilai *precision* pada Skenario 11 mengalami peningkatan. Berdasarkan Tabel 4.11, diperoleh nilai *macro precision* sebesar 44,32%, lebih tinggi dibandingkan Skenario 10. Peningkatan ini terlihat

pada hampir seluruh label, khususnya pada label *HS_Race* dan *HS_Other*, yang menunjukkan bahwa sistem mulai mampu menekan jumlah prediksi positif yang tidak tepat (*false positive*).

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut berdampak langsung pada nilai *F1-score*. Berdasarkan Tabel 4.11, diperoleh nilai *macro F1-score* sebesar 56,86%, yang menunjukkan peningkatan dibandingkan Skenario 10. Peningkatan ini menandakan bahwa kualitas prediksi sistem secara keseluruhan menjadi lebih stabil dan seimbang setelah penyesuaian nilai *threshold*.

Analisis *confusion matrix* pada Tabel 4.11 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 10, sementara jumlah *false negative* meningkat secara moderat. Pola ini menunjukkan bahwa sistem mulai mengurangi kesalahan prediksi positif dengan konsekuensi berkurangnya sensitivitas pada sebagian kategori.

Secara keseluruhan, hasil pengujian pada Skenario Uji 11 menunjukkan bahwa peningkatan nilai *threshold* menjadi 0,2 pada konfigurasi *LSTM* dengan 128 unit dan *batch size* 64 mampu memperbaiki keseimbangan antara ketepatan dan sensitivitas prediksi. Peningkatan nilai *subset accuracy* dan penurunan *hamming loss* menegaskan adanya perbaikan performa klasifikasi *Multi-Label* dibandingkan Skenario 10. Dengan demikian, Skenario 11 menunjukkan bahwa penyesuaian nilai *threshold* berperan penting dalam memaksimalkan manfaat peningkatan *batch size* dan kapasitas model terhadap kualitas prediksi.

4.1.12 Hasil Skenario Uji 12

Skenario Uji 12 merupakan kelanjutan dari Skenario 11 dengan tetap menggunakan *embedding* berukuran 50 dimensi, *LSTM* dengan 128 unit, serta *batch size* sebesar 64. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,3. Penyesuaian ini dilakukan untuk meningkatkan selektivitas sistem dalam menetapkan label positif pada konfigurasi dengan kapasitas model dan *batch size* yang lebih besar. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.12.

Tabel 4.12 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 12

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1011	579	201	67	4	644	
TN	815	1231	1992	2381	2514	1240	
FP	701	685	242	89	2	633	
FN	107	139	199	97	114	117	
Matrix Evaluasi							
Precision (%)	59,05	45,81	45,37	42,95	66,67	50,43	51,71
Recall (%)	90,43	80,64	50,25	40,85	3,39	84,63	58,36
F1-Score (%)	71,45	58,43	47,69	41,88	6,45	63,20	48,18
Subset Accuracy (%)	46,09						
Hamming Loss (%)	19,77						

Berdasarkan Tabel 4.12, diperoleh nilai *subset accuracy* sebesar 46,09% dan *hamming loss* sebesar 19,77%. Dibandingkan dengan Skenario 11, nilai *subset accuracy* mengalami peningkatan, sementara nilai *hamming loss* menurun. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada konfigurasi ini mampu

memperbaiki ketepatan prediksi kombinasi label sekaligus mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, peningkatan nilai *threshold* kembali memengaruhi keseimbangan antara nilai *precision* dan *recall*. Berdasarkan Tabel 4.12, diperoleh nilai *macro precision* sebesar 51,71% dan *macro recall* sebesar 58,36%. Nilai *precision* menunjukkan peningkatan dibandingkan Skenario 11, yang mengindikasikan berkurangnya jumlah prediksi positif yang tidak tepat. Namun, nilai *recall* mengalami penurunan yang cukup signifikan pada beberapa label, yang menandakan meningkatnya selektivitas sistem.

Pada label *HS*, sistem masih mampu mempertahankan nilai *recall* yang tinggi sebesar 90,43% dengan nilai *precision* sebesar 59,05%, sehingga menghasilkan nilai *F1-score* sebesar 71,45%. Hal ini menunjukkan bahwa kategori ujaran kebencian utama tetap dapat dikenali dengan baik. Namun, pada label *HS_Race* terjadi penurunan nilai *recall* yang sangat drastis, yaitu hanya sebesar 3,39%, meskipun nilai *precision* mencapai 66,67%. Kondisi ini menunjukkan bahwa dengan nilai *threshold* yang tinggi, sistem menjadi sangat selektif pada kategori tersebut sehingga sebagian besar data positif tidak terdeteksi.

Dampak penurunan nilai *recall* tersebut tercermin pada nilai *F1-score*. Berdasarkan Tabel 4.12, label *HS_Race* memiliki nilai *F1-score* yang sangat rendah, yaitu 6,45%, yang menurunkan nilai *macro F1-score* menjadi 48,18%. Meskipun beberapa label lain, seperti *HS* dan *HS_Other*, masih menunjukkan nilai *F1-score* yang relatif baik, ketimpangan antarlabel menjadi lebih terlihat pada skenario ini.

Analisis *confusion matrix* pada Tabel 4.12 memperkuat temuan tersebut. Jumlah *false positive* pada sebagian label mengalami penurunan, yang menunjukkan peningkatan ketepatan prediksi positif. Namun, jumlah *false negative* meningkat secara signifikan pada label *HS_Race* , yang menegaskan bahwa sebagian besar data positif pada kategori tersebut tidak berhasil terdeteksi akibat meningkatnya nilai *threshold*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 12 menunjukkan bahwa peningkatan nilai *threshold* menjadi 0,3 pada konfigurasi *LSTM* dengan 128 unit dan *batch size* 64 mampu meningkatkan ketepatan prediksi secara umum, yang tercermin pada peningkatan nilai *subset accuracy* dan penurunan *hamming loss*. Namun, peningkatan selektivitas yang terlalu tinggi juga berdampak negatif pada sensitivitas sistem terhadap beberapa label tertentu, terutama *HS_Race* . Dengan demikian, Skenario 12 menegaskan bahwa pemilihan nilai *threshold* yang terlalu tinggi dapat menyebabkan ketidakseimbangan performa antarlabel dan perlu dipertimbangkan secara hati-hati pada skenario-skenario berikutnya.

4.1.13 Hasil Skenario Uji 13

Skenario Uji 13 merupakan pengujian lanjutan yang bertujuan untuk mengamati pengaruh peningkatan dimensi *embedding* terhadap hasil klasifikasi *Multi-Label*. Pada skenario ini digunakan *embedding* berukuran 100 dimensi, *LSTM* dengan 64 unit, *batch size* sebesar 32, serta nilai *threshold* 0,1. Perubahan dimensi *embedding* ini dilakukan untuk memperkaya representasi semantik teks, sementara nilai *threshold* yang rendah dipertahankan untuk melihat perilaku sistem pada

kondisi yang masih permisif. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.13.

Tabel 4.13 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 13

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1108	704	342	126	104	740	
TN	361	547	1397	2131	2309	741	
FP	1155	1369	837	339	207	1132	
FN	10	14	58	38	14	21	
Matrix Evaluasi							
Precision (%)	48,96	33,96	29,01	27,10	33,44	39,53	35,33
Recall (%)	99,11	98,05	85,50	76,83	88,14	97,24	90,81
F1-Score (%)	65,54	50,45	43,32	40,06	48,48	56,21	50,68
Subset Accuracy (%)	20,73						
Hamming Loss (%)	32,87						

Berdasarkan Tabel 4.13, diperoleh nilai *subset accuracy* sebesar 20,73% dan *hamming loss* sebesar 32,87%. Jika dibandingkan dengan Skenario 1 yang menggunakan dimensi *embedding* lebih kecil pada *threshold* yang sama, nilai *subset accuracy* menunjukkan peningkatan, meskipun masih tergolong rendah. Sementara itu, nilai *hamming loss* masih relatif tinggi, yang mengindikasikan bahwa kesalahan prediksi per label masih sering terjadi.

Ditinjau dari metrik per label, sistem kembali menunjukkan nilai *recall* yang sangat tinggi, dengan *macro recall* mencapai 90,81%. Label *HS*, *HS_Individual*, dan *HS_Other* masing-masing memiliki nilai *recall* di atas 97%. Hal ini menunjukkan bahwa peningkatan dimensi *embedding* mampu membantu sistem

dalam menangkap keberadaan ujaran kebencian pada berbagai kategori. Tingginya nilai *recall* ini juga dipengaruhi oleh penggunaan nilai *threshold* yang rendah, sehingga sistem tetap bersifat permisif dalam menetapkan label positif.

Namun demikian, nilai *precision* pada Skenario 13 masih relatif rendah. Berdasarkan Tabel 4.13, diperoleh nilai *macro precision* sebesar 35,33%, yang menunjukkan bahwa sistem masih menghasilkan banyak prediksi positif yang tidak tepat. Kondisi ini terlihat pada hampir seluruh label, khususnya *HS_Group* dan *HS_Religion*, yang memiliki nilai *precision* rendah meskipun nilai *recall*-nya tinggi. Hal ini mengindikasikan bahwa peningkatan dimensi *embedding* belum mampu secara signifikan mengurangi jumlah *false positive* pada kondisi *threshold* yang permisif.

Ketidakseimbangan antara nilai *precision* dan *recall* tersebut berdampak pada nilai *F1-score*. Berdasarkan Tabel 4.13, diperoleh nilai *macro F1-score* sebesar 50,68%, yang menunjukkan peningkatan dibandingkan Skenario 1, tetapi masih berada pada tingkat yang moderat. Label *HS* memiliki nilai *F1-score* tertinggi karena didukung oleh nilai *recall* yang sangat tinggi dan nilai *precision* yang relatif lebih baik dibandingkan label lainnya.

Analisis *confusion matrix* pada Tabel 4.13 memperkuat temuan tersebut. Jumlah *false negative* pada seluruh label relatif kecil, yang menjelaskan tingginya nilai *recall*. Sebaliknya, jumlah *false positive* masih cukup besar pada hampir seluruh label, yang menjadi penyebab utama rendahnya nilai *precision* dan rendahnya nilai *subset accuracy*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 13 menunjukkan bahwa peningkatan dimensi *embedding* menjadi 100 pada nilai *threshold* yang rendah mampu meningkatkan kemampuan sistem dalam mendeteksi keberadaan ujaran kebencian, yang tercermin pada tingginya nilai *recall*. Namun, sistem masih cenderung menghasilkan prediksi positif yang berlebihan sehingga nilai *precision* dan *subset accuracy* belum optimal. Dengan demikian, Skenario 13 menegaskan bahwa peningkatan dimensi *embedding* perlu diimbangi dengan penyesuaian nilai *threshold* atau *hyperparameter* lain agar diperoleh keseimbangan yang lebih baik antara ketepatan dan sensitivitas prediksi pada skenario-skenario berikutnya.

4.1.14 Hasil Skenario Uji 14

Skenario Uji 14 merupakan pengujian lanjutan dari Skenario 13 dengan tetap menggunakan *embedding* berukuran 100 dimensi, *LSTM* dengan 64 unit, serta *batch size* sebesar 32. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,2. Penyesuaian ini dilakukan untuk mengurangi sifat permisif sistem yang masih terlihat pada Skenario 13 serta untuk mengevaluasi dampaknya terhadap keseimbangan antara nilai *precision* dan *recall*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.14.

Tabel 4.14 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 14

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1061	635	292	112	96	676	
TN	718	1081	1746	2377	2408	1252	
FP	798	835	488	93	108	621	
FN	57	83	108	52	22	85	
Matrix Evaluasi							

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Precision (%)	57,07	43,20	37,44	54,63	47,06	52,12	48,59
Recall (%)	94,90	88,44	73,00	68,29	81,36	88,83	82,47
F1-Score (%)	71,28	58,04	49,49	60,70	59,63	65,69	60,81
Subset Accuracy (%)	39,33						
Hamming Loss (%)	**21,20**						

Berdasarkan Tabel 4.14, diperoleh nilai *subset accuracy* sebesar 39,33% dan *hamming loss* sebesar 21,20%. Dibandingkan dengan Skenario 13, nilai *subset accuracy* meningkat secara signifikan, sementara nilai *hamming loss* mengalami penurunan. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada dimensi *embedding* yang lebih besar mampu memperbaiki ketepatan prediksi kombinasi label sekaligus mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, sistem masih mempertahankan nilai *recall* yang relatif tinggi, dengan *macro recall* sebesar 82,47%. Label *HS* dan *HS_Other* masing-masing memiliki nilai *recall* sebesar 94,90% dan 88,83%, yang menunjukkan bahwa sebagian besar ujaran kebencian tetap dapat terdeteksi dengan baik. Namun, jika dibandingkan dengan Skenario 13, nilai *recall* pada beberapa label mengalami penurunan, yang mencerminkan meningkatnya selektivitas sistem dalam menetapkan label positif.

Seiring dengan penurunan nilai *recall*, nilai *precision* pada Skenario 14 mengalami peningkatan yang cukup nyata. Berdasarkan Tabel 4.14, diperoleh nilai *macro precision* sebesar 48,59%, yang menunjukkan bahwa sistem mulai mampu

menekan jumlah prediksi positif yang tidak tepat (*false positive*). Peningkatan nilai *precision* terlihat jelas pada label *HS_Religion* yang mencapai 54,63% serta *HS_Other* sebesar 52,12%.

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut berdampak langsung pada nilai *F1-score*. Berdasarkan Tabel 4.14, diperoleh nilai *macro F1-score* sebesar 60,81%, yang menunjukkan peningkatan dibandingkan Skenario 13. Peningkatan ini menandakan bahwa kualitas prediksi sistem secara keseluruhan menjadi lebih stabil dan seimbang setelah penyesuaian nilai *threshold*.

Analisis *confusion matrix* pada Tabel 4.14 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 13, sementara jumlah *false negative* meningkat secara moderat. Pola ini menunjukkan bahwa sistem mulai mengurangi kesalahan prediksi positif dengan konsekuensi berkurangnya sensitivitas pada sebagian kategori.

Secara keseluruhan, hasil pengujian pada Skenario Uji 14 menunjukkan bahwa peningkatan nilai *threshold* menjadi 0,2 pada konfigurasi *embedding* berukuran 100 dimensi mampu memperbaiki keseimbangan antara ketepatan dan sensitivitas prediksi. Peningkatan nilai *subset accuracy* dan penurunan *hamming loss* menegaskan adanya perbaikan performa klasifikasi *Multi-Label* dibandingkan Skenario 13. Dengan demikian, Skenario 14 menunjukkan bahwa kombinasi dimensi *embedding* yang lebih besar dengan nilai *threshold* menengah memberikan hasil yang lebih stabil dan menjadi pijakan penting untuk pengujian pada skenario berikutnya.

4.1.15 Hasil Skenario Uji 15

Skenario Uji 15 merupakan kelanjutan dari Skenario 14 dengan tetap menggunakan *embedding* berukuran 100 dimensi, *LSTM* dengan 64 unit, serta *batch size* sebesar 32. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,3. Penyesuaian ini dilakukan untuk meningkatkan selektivitas sistem dalam menetapkan label positif setelah pada Skenario 14 diperoleh keseimbangan yang lebih baik antara nilai *precision* dan *recall*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.15.

Tabel 4.15 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 15

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1026	572	227	66	78	625	
TN	859	1318	1979	2398	2456	1401	
FP	657	598	255	72	60	472	
FN	92	146	173	98	40	136	
Matrix Evaluasi							
Precision (%)	60,96	48,89	47,10	47,83	56,52	56,97	53,04
Recall (%)	91,77	79,67	56,75	40,24	66,10	82,13	69,44
F1-Score (%)	73,26	60,59	51,47	43,71	60,94	67,28	59,54
Subset Accuracy (%)	49,16						
Hamming Loss (%)	17,71						

Berdasarkan Tabel 4.15, diperoleh nilai *subset accuracy* sebesar 49,16% dan *hamming loss* sebesar 17,71%. Dibandingkan dengan Skenario 14, nilai *subset accuracy* kembali meningkat, sementara nilai *hamming loss* mengalami penurunan. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada konfigurasi ini

mampu memperbaiki ketepatan prediksi kombinasi label sekaligus mengurangi kesalahan prediksi per label secara lebih konsisten.

Ditinjau dari metrik per label, peningkatan nilai *threshold* menghasilkan keseimbangan yang lebih baik antara nilai *precision* dan *recall*. Berdasarkan Tabel 4.15, diperoleh nilai *macro precision* sebesar 53,04% dan *macro recall* sebesar 69,44%. Peningkatan nilai *precision* menunjukkan bahwa jumlah prediksi positif yang tidak tepat semakin berkurang, sementara nilai *recall* masih berada pada tingkat yang cukup baik untuk sebagian besar label.

Pada label *HS*, sistem mempertahankan nilai *recall* yang tinggi sebesar 91,77% dengan nilai *precision* sebesar 60,96%, sehingga menghasilkan nilai *F1-score* tertinggi sebesar 73,26%. Hal ini menunjukkan bahwa kategori ujaran kebencian utama tetap dapat dikenali dengan baik meskipun nilai *threshold* dinaikkan. Selain itu, label *HS_Individual* dan *HS_Other* juga menunjukkan nilai *F1-score* yang relatif tinggi, masing-masing sebesar 60,59% dan 67,28%, yang mencerminkan keseimbangan prediksi yang semakin stabil.

Perubahan keseimbangan tersebut tercermin pada nilai *F1-score* secara keseluruhan. Berdasarkan Tabel 4.15, diperoleh nilai *macro F1-score* sebesar 59,54%, yang menunjukkan peningkatan dibandingkan Skenario 14. Peningkatan ini menandakan bahwa kualitas prediksi sistem secara umum menjadi lebih baik setelah penyesuaian nilai *threshold*.

Analisis *confusion matrix* pada Tabel 4.15 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 14, yang menunjukkan peningkatan ketepatan prediksi

positif. Di sisi lain, jumlah *false negative* masih muncul pada beberapa label, namun berada pada tingkat yang lebih terkendali dibandingkan skenario dengan *threshold* yang lebih rendah.

Secara keseluruhan, hasil pengujian pada Skenario Uji 15 menunjukkan bahwa kombinasi *embedding* berukuran 100 dimensi dengan nilai *threshold* 0,3 mampu menghasilkan keseimbangan yang lebih optimal antara ketepatan dan sensitivitas prediksi. Peningkatan nilai *subset accuracy* dan penurunan *hamming loss* menegaskan adanya perbaikan performa klasifikasi *Multi-Label* dibandingkan Skenario 14. Dengan demikian, Skenario 15 menunjukkan konfigurasi yang lebih stabil dan menjadi salah satu kandidat konfigurasi terbaik pada kelompok pengujian dengan *embedding* 100 dimensi.

4.1.16 Hasil Skenario Uji 16

Skenario Uji 16 merupakan pengujian lanjutan yang bertujuan untuk mengamati pengaruh peningkatan *batch size* pada konfigurasi dengan *embedding* berukuran 100 dimensi dan *LSTM* dengan 64 unit. Pada skenario ini digunakan *batch size* sebesar 64 dengan nilai *threshold* 0,1. Perubahan *batch size* dilakukan untuk mengevaluasi kestabilan proses pembelajaran ketika jumlah data yang diproses dalam satu iterasi diperbesar, sementara nilai *threshold* yang rendah dipertahankan untuk mengamati perilaku sistem pada kondisi yang masih permisif. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.16.

Tabel 4.16 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 16

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1103	689	346	120	97	736	
TN	385	655	1380	2162	2369	734	
FP	1131	1261	854	308	147	1139	
FN	15	29	54	44	21	25	
Matrix Evaluasi							
Precision (%)	49,37	35,33	28,83	28,04	39,75	39,25	36,76
Recall (%)	98,66	95,96	86,50	73,17	82,20	96,71	88,87
F1-Score (%)	65,81	51,65	43,25	40,54	53,59	55,84	51,78
Subset Accuracy (%)	21,03						
Hamming Loss (%)	31,81						

Berdasarkan Tabel 4.16, diperoleh nilai *subset accuracy* sebesar 21,03% dan *hamming loss* sebesar 31,81%. Jika dibandingkan dengan Skenario 15, nilai *subset accuracy* menurun cukup tajam, sementara nilai *hamming loss* kembali meningkat. Hasil ini menunjukkan bahwa peningkatan *batch size* tanpa disertai penyesuaian nilai *threshold* belum mampu memperbaiki ketepatan prediksi kombinasi label.

Ditinjau dari metrik per label, sistem kembali menunjukkan nilai *recall* yang sangat tinggi, dengan *macro recall* mencapai 88,87%. Label *HS* dan *HS_Individual* masing-masing memiliki nilai *recall* sebesar 98,66% dan 95,96%, yang menunjukkan bahwa sistem hampir selalu berhasil mendeteksi keberadaan ujaran kebencian. Tingginya nilai *recall* ini konsisten dengan penggunaan nilai *threshold*

yang rendah, sehingga sistem kembali bersifat permisif dalam menetapkan label positif.

Namun demikian, nilai *precision* pada Skenario 16 masih relatif rendah. Berdasarkan Tabel 4.16, diperoleh nilai *macro precision* sebesar 36,76%, yang menunjukkan bahwa sistem masih menghasilkan banyak prediksi positif yang tidak tepat. Kondisi ini terlihat pada hampir seluruh label, khususnya *HS_Group* dan *HS_Religion*, yang memiliki nilai *precision* rendah meskipun nilai *recall*-nya tinggi. Hal ini mengindikasikan bahwa peningkatan *batch size* belum mampu mengurangi jumlah *false positive* secara signifikan pada kondisi *threshold* yang permisif.

Ketidakseimbangan antara nilai *precision* dan *recall* tersebut berdampak pada nilai *F1-score*. Berdasarkan Tabel 4.16, diperoleh nilai *macro F1-score* sebesar 51,78%, yang relatif lebih rendah dibandingkan Skenario 15. Hal ini menunjukkan bahwa perubahan *batch size* tanpa penyesuaian nilai *threshold* justru menurunkan kualitas prediksi secara keseluruhan.

Analisis *confusion matrix* pada Tabel 4.16 memperkuat temuan tersebut. Jumlah *false negative* pada seluruh label relatif kecil, yang menjelaskan tingginya nilai *recall*. Sebaliknya, jumlah *false positive* tetap tinggi pada hampir seluruh label, yang menjadi penyebab utama rendahnya nilai *precision* dan rendahnya nilai *subset accuracy*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 16 menunjukkan bahwa peningkatan *batch size* menjadi 64 pada konfigurasi *embedding* 100 dimensi dan nilai *threshold* yang rendah belum mampu meningkatkan performa klasifikasi

Multi-Label. Sistem kembali menunjukkan kecenderungan memberikan prediksi positif secara berlebihan, sehingga menghasilkan banyak kesalahan prediksi positif. Dengan demikian, Skenario 16 menegaskan bahwa peningkatan *batch size* perlu diimbangi dengan penyesuaian nilai *threshold* atau *hyperparameter* lain agar diperoleh keseimbangan yang lebih baik antara ketepatan dan sensitivitas prediksi pada skenario-skenario berikutnya.

4.1.17 Hasil Skenario Uji 17

Skenario Uji 17 merupakan pengujian lanjutan dari Skenario 16 dengan tetap menggunakan *embedding* berukuran 100 dimensi, *LSTM* dengan 64 unit, serta *batch size* sebesar 64. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,2. Penyesuaian ini dilakukan untuk mengurangi sifat permisif sistem yang masih terlihat pada Skenario 16 serta untuk mengevaluasi dampaknya terhadap keseimbangan antara nilai *precision* dan *recall*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.17.

Tabel 4.17 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 17

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1063	636	288	97	81	668	
TN	688	1029	1759	2323	2444	1225	
FP	828	887	475	147	72	648	
FN	55	82	112	67	37	93	
Matrix Evaluasi							
Precision (%)	56,21	41,76	37,75	39,75	52,94	50,76	46,53
Recall (%)	95,08	88,58	72,00	59,15	68,64	87,78	78,54
F1-Score (%)	70,65	56,76	49,53	47,55	59,78	64,32	58,10
Subset Accuracy (%)	39,10						

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Hamming Loss (%)	22,17						

Berdasarkan Tabel 4.17, diperoleh nilai *subset accuracy* sebesar 39,10% dan *hamming loss* sebesar 22,17%. Dibandingkan dengan Skenario 16, nilai *subset accuracy* meningkat secara signifikan, sementara nilai *hamming loss* mengalami penurunan. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada konfigurasi dengan *batch size* yang lebih besar mulai memperbaiki ketepatan prediksi kombinasi label dan mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, sistem masih mempertahankan nilai *recall* yang relatif tinggi dengan *macro recall* sebesar 78,54%. Label *HS* memiliki nilai *recall* sebesar 95,08%, yang menunjukkan bahwa sebagian besar ujaran kebencian utama masih dapat terdeteksi dengan baik. Namun, dibandingkan dengan Skenario 16, nilai *recall* pada beberapa label mengalami penurunan, yang mencerminkan meningkatnya selektivitas sistem dalam menetapkan label positif.

Seiring dengan penurunan nilai *recall*, nilai *precision* pada Skenario 17 mengalami peningkatan. Berdasarkan Tabel 4.17, diperoleh nilai *macro precision* sebesar 46,53%, lebih tinggi dibandingkan Skenario 16. Peningkatan ini terlihat pada hampir seluruh label, khususnya pada label *HS_Race* dan *HS_Other*, yang menunjukkan bahwa sistem mulai mampu menekan jumlah prediksi positif yang tidak tepat (*false positive*).

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut berdampak langsung pada nilai *F1-score*. Berdasarkan Tabel 4.17, diperoleh nilai

macro F1-score sebesar 58,10%, yang menunjukkan peningkatan dibandingkan Skenario 16. Peningkatan ini menandakan bahwa kualitas prediksi sistem secara keseluruhan menjadi lebih stabil dan seimbang setelah penyesuaian nilai *threshold*.

Analisis *confusion matrix* pada Tabel 4.17 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 16, sementara jumlah *false negative* meningkat secara moderat. Pola ini menunjukkan bahwa sistem mulai mengurangi kesalahan prediksi positif dengan konsekuensi berkurangnya sensitivitas pada sebagian kategori.

Secara keseluruhan, hasil pengujian pada Skenario Uji 17 menunjukkan bahwa peningkatan nilai *threshold* menjadi 0,2 pada konfigurasi *embedding* 100 dimensi dan *batch size* 64 mampu memperbaiki keseimbangan antara ketepatan dan sensitivitas prediksi. Peningkatan nilai *subset accuracy* dan penurunan *hamming loss* menegaskan adanya perbaikan performa klasifikasi *Multi-Label* dibandingkan Skenario 16. Dengan demikian, Skenario 17 menunjukkan bahwa penyesuaian nilai *threshold* merupakan langkah penting untuk memaksimalkan manfaat peningkatan *batch size* pada konfigurasi dengan dimensi *embedding* yang lebih besar.

4.1.18 Hasil Skenario Uji 18

Skenario Uji 18 merupakan kelanjutan dari Skenario 17 dengan tetap menggunakan *embedding* berukuran 100 dimensi, *LSTM* dengan 64 unit, serta *batch size* sebesar 64. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,3. Penyesuaian ini dilakukan untuk meningkatkan selektivitas sistem dalam menetapkan label positif setelah pada Skenario 17

diperoleh keseimbangan yang lebih baik antara nilai *precision* dan *recall*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.18.

Tabel 4.18 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 18

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	987	541	202	91	74	603	
TN	924	1360	2022	2400	2477	1413	
FP	592	556	212	70	39	460	
FN	131	177	198	73	44	158	
Matrix Evaluasi							
Precision (%)	62,51	49,32	48,79	56,52	65,49	56,73	56,56
Recall (%)	88,28	75,35	50,50	55,49	62,71	79,24	68,59
F1-Score (%)	73,19	59,61	49,63	56,00	64,07	66,12	61,44
Subset Accuracy (%)	52,16						
Hamming Loss (%)	17,15						

Berdasarkan Tabel 4.18, diperoleh nilai *subset accuracy* sebesar 52,16% dan *hamming loss* sebesar 17,15%. Dibandingkan dengan Skenario 17, nilai *subset accuracy* meningkat secara nyata, sementara nilai *hamming loss* mengalami penurunan. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada konfigurasi ini mampu memperbaiki ketepatan prediksi kombinasi label sekaligus mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, peningkatan nilai *threshold* menghasilkan keseimbangan yang lebih baik antara nilai *precision* dan *recall*. Berdasarkan Tabel 4.18, diperoleh nilai *macro precision* sebesar 56,56% dan *macro recall* sebesar 68,59%. Peningkatan nilai *precision* menunjukkan bahwa jumlah prediksi positif

yang tidak tepat semakin berkurang, sementara nilai *recall* masih berada pada tingkat yang memadai untuk sebagian besar label.

Pada label *HS*, sistem mempertahankan nilai *recall* yang tinggi sebesar 88,28% dengan nilai *precision* sebesar 62,51%, sehingga menghasilkan nilai *F1-score* sebesar 73,19%. Hal ini menunjukkan bahwa kategori ujaran kebencian utama tetap dapat dikenali dengan baik meskipun selektivitas sistem meningkat. Selain itu, label *HS_Individual*, *HS_Race* , dan *HS_Other* juga menunjukkan nilai *F1-score* yang relatif tinggi, yang mencerminkan kestabilan prediksi pada berbagai kategori.

Perubahan keseimbangan tersebut tercermin pada nilai *F1-score* secara keseluruhan. Berdasarkan Tabel 4.18, diperoleh nilai *macro F1-score* sebesar 61,44%, yang menunjukkan peningkatan dibandingkan Skenario 17. Peningkatan ini menandakan bahwa kualitas prediksi sistem secara umum menjadi lebih baik setelah penyesuaian nilai *threshold*.

Analisis *confusion matrix* pada Tabel 4.18 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 17, yang menunjukkan peningkatan ketepatan prediksi positif. Di sisi lain, jumlah *false negative* meningkat secara moderat pada beberapa label, namun masih berada pada tingkat yang dapat diterima.

Secara keseluruhan, hasil pengujian pada Skenario Uji 18 menunjukkan bahwa kombinasi *embedding* berukuran 100 dimensi dengan *batch size* 64 dan nilai *threshold* 0,3 mampu menghasilkan keseimbangan yang lebih optimal antara ketepatan dan sensitivitas prediksi. Peningkatan nilai *subset accuracy* dan

penurunan *hamming loss* menegaskan adanya perbaikan performa klasifikasi *Multi-Label* dibandingkan Skenario 17. Dengan demikian, Skenario 18 dapat dipandang sebagai salah satu konfigurasi dengan performa terbaik pada kelompok pengujian menggunakan *embedding* 100 dimensi dan *batch size* 64.

4.1.19 Hasil Skenario Uji 19

Skenario Uji 19 merupakan pengujian lanjutan yang bertujuan untuk mengamati pengaruh peningkatan jumlah unit *LSTM* pada konfigurasi dengan *embedding* berukuran 100 dimensi. Pada skenario ini digunakan *LSTM* dengan 128 unit, *batch size* sebesar 32, serta nilai *threshold* 0,1. Peningkatan jumlah unit *LSTM* dilakukan untuk memperbesar kapasitas pemodelan ketergantungan sekuensial, sementara nilai *threshold* yang rendah dipertahankan untuk melihat perilaku sistem pada kondisi yang masih permisif. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.19.

Tabel 4.19 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 19

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1112	706	350	134	99	751	
TN	270	598	1344	2127	2341	693	
FP	1246	1318	890	343	175	1180	
FN	6	12	50	30	19	10	
Matrix Evaluasi							
Precision (%)	47,16	34,88	28,23	28,09	36,13	38,89	35,56
Recall (%)	99,46	98,33	87,50	81,71	83,90	98,69	91,60
F1-Score (%)	63,98	51,50	42,68	41,81	50,51	55,79	51,05
Subset Accuracy (%)	17,08						
Hamming Loss	33,40						

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
(%)							

Berdasarkan Tabel 4.19, diperoleh nilai *subset accuracy* sebesar 17,08% dan *hamming loss* sebesar 33,40%. Jika dibandingkan dengan Skenario 18 yang menggunakan nilai *threshold* lebih tinggi, nilai *subset accuracy* mengalami penurunan yang cukup signifikan, sementara nilai *hamming loss* kembali meningkat. Hasil ini menunjukkan bahwa peningkatan jumlah unit *LSTM* tanpa penyesuaian nilai *threshold* belum mampu meningkatkan ketepatan prediksi kombinasi label.

Ditinjau dari metrik per label, sistem kembali menunjukkan nilai *recall* yang sangat tinggi, dengan *macro recall* mencapai 91,60%. Label *HS* dan *HS_Other* masing-masing memiliki nilai *recall* sebesar 99,46% dan 98,69%, yang menunjukkan bahwa sistem hampir selalu berhasil mendeteksi keberadaan ujaran kebencian. Tingginya nilai *recall* ini selaras dengan penggunaan nilai *threshold* yang rendah, sehingga sistem cenderung menetapkan label positif secara luas.

Namun demikian, nilai *precision* pada Skenario 19 masih relatif rendah. Berdasarkan Tabel 4.19, diperoleh nilai *macro precision* sebesar 35,56%, yang menunjukkan bahwa sistem menghasilkan banyak prediksi positif yang tidak tepat. Kondisi ini terlihat pada hampir seluruh label, khususnya *HS_Group* dan *HS_Religion*, yang memiliki nilai *precision* rendah meskipun nilai *recall*-nya tinggi. Hal ini mengindikasikan bahwa peningkatan kapasitas *LSTM* belum mampu menekan jumlah *false positive* pada kondisi *threshold* yang permisif.

Ketidakseimbangan antara nilai *precision* dan *recall* tersebut berdampak pada nilai *F1-score*. Berdasarkan Tabel 4.19, diperoleh nilai *macro F1-score* sebesar 51,05%, yang relatif serupa dengan skenario-skenario sebelumnya pada kondisi permisif. Hal ini menunjukkan bahwa peningkatan jumlah unit *LSTM* saja belum cukup untuk meningkatkan kualitas prediksi tanpa penyesuaian parameter pengambilan keputusan.

Analisis *confusion matrix* pada Tabel 4.19 memperkuat temuan tersebut. Jumlah *false negative* pada seluruh label relatif kecil, yang menjelaskan tingginya nilai *recall*. Sebaliknya, jumlah *false positive* masih sangat tinggi pada hampir seluruh label, yang menjadi penyebab utama rendahnya nilai *precision* dan rendahnya nilai *subset accuracy*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 19 menunjukkan bahwa peningkatan jumlah unit *LSTM* menjadi 128 pada *embedding* 100 dimensi dan nilai *threshold* yang rendah belum mampu memperbaiki performa klasifikasi *Multi-Label*. Sistem kembali menunjukkan kecenderungan menetapkan label positif secara berlebihan, sehingga menghasilkan banyak kesalahan prediksi positif. Dengan demikian, Skenario 19 menegaskan bahwa peningkatan kapasitas model perlu diimbangi dengan penyesuaian nilai *threshold* atau *hyperparameter* lain agar diperoleh keseimbangan yang lebih baik antara ketepatan dan sensitivitas prediksi pada skenario-skenario berikutnya.

4.1.20 Hasil Skenario Uji 20

Skenario Uji 20 merupakan pengujian lanjutan dari Skenario 19 dengan tetap menggunakan *embedding* berukuran 100 dimensi dan *LSTM* dengan 128 unit,

serta *batch size* sebesar 32. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,2. Penyesuaian ini dilakukan untuk mengurangi sifat permisif sistem yang masih terlihat pada Skenario 19 serta untuk mengevaluasi dampaknya terhadap keseimbangan antara nilai *precision* dan *recall*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.20.

Tabel 4.20 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 20

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1074	645	303	104	93	677	
TN	652	1024	1695	2247	2394	1151	
FP	864	892	539	223	122	722	
FN	44	73	97	60	25	84	
Matrix Evaluasi							
Precision (%)	55,42	41,96	35,99	31,80	43,26	48,39	42,80
Recall (%)	96,06	89,83	75,75	63,41	78,81	88,96	82,14
F1-Score (%)	70,29	57,21	48,79	42,36	55,86	62,69	56,20
Subset Accuracy (%)	34,66						
Hamming Loss (%)	23,70						

Berdasarkan Tabel 4.20, diperoleh nilai *subset accuracy* sebesar 34,66% dan *hamming loss* sebesar 23,70%. Dibandingkan dengan Skenario 19, nilai *subset accuracy* meningkat secara nyata, sementara nilai *hamming loss* mengalami penurunan. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada konfigurasi dengan kapasitas *LSTM* yang lebih besar mulai memperbaiki ketepatan prediksi kombinasi label dan mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, sistem masih mempertahankan nilai *recall* yang relatif tinggi dengan *macro recall* sebesar 82,14%. Label *HS* dan *HS_Other* masing-masing memiliki nilai *recall* sebesar 96,06% dan 88,96%, yang menunjukkan bahwa sebagian besar ujaran kebencian utama tetap dapat terdeteksi dengan baik. Namun, dibandingkan dengan Skenario 19, nilai *recall* pada beberapa label mengalami penurunan, yang mencerminkan meningkatnya selektivitas sistem dalam menetapkan label positif.

Seiring dengan penurunan nilai *recall*, nilai *precision* pada Skenario 20 mengalami peningkatan. Berdasarkan Tabel 4.20, diperoleh nilai *macro precision* sebesar 42,80%, lebih tinggi dibandingkan Skenario 19. Peningkatan ini terlihat pada hampir seluruh label, khususnya pada *HS* dan *HS_Other*, yang menunjukkan bahwa sistem mulai mampu menekan jumlah prediksi positif yang tidak tepat (*false positive*).

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut berdampak langsung pada nilai *F1-score*. Berdasarkan Tabel 4.20, diperoleh nilai *macro F1-score* sebesar 56,20%, yang menunjukkan peningkatan dibandingkan Skenario 19. Peningkatan ini menandakan bahwa kualitas prediksi sistem secara keseluruhan menjadi lebih stabil dan seimbang setelah penyesuaian nilai *threshold*.

Analisis *confusion matrix* pada Tabel 4.20 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 19, sementara jumlah *false negative* meningkat secara moderat. Pola ini menunjukkan bahwa sistem mulai mengurangi kesalahan prediksi positif dengan konsekuensi berkurangnya sensitivitas pada sebagian kategori.

Secara keseluruhan, hasil pengujian pada Skenario Uji 20 menunjukkan bahwa peningkatan nilai *threshold* menjadi 0,2 pada konfigurasi *LSTM* dengan 128 unit dan *embedding* 100 dimensi mampu memperbaiki keseimbangan antara ketepatan dan sensitivitas prediksi. Peningkatan nilai *subset accuracy* dan penurunan *hamming loss* menegaskan adanya perbaikan performa klasifikasi *Multi-Label* dibandingkan Skenario 19. Dengan demikian, Skenario 20 menunjukkan bahwa penyesuaian nilai *threshold* merupakan langkah penting untuk memanfaatkan peningkatan kapasitas model secara lebih optimal.

4.1.21 Hasil Skenario Uji 21

Skenario Uji 21 merupakan kelanjutan dari Skenario 20 dengan tetap menggunakan *embedding* berukuran 100 dimensi, *LSTM* dengan 128 unit, serta *batch size* sebesar 32. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,3. Penyesuaian ini dilakukan untuk meningkatkan selektivitas sistem dalam menetapkan label positif setelah pada Skenario 20 diperoleh keseimbangan yang lebih baik antara nilai *precision* dan *recall*. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.21.

Tabel 4.21 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 21

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1075	572	134	40	71	665	
TN	534	1150	2091	2390	2432	1136	
FP	982	766	143	80	84	737	
FN	43	146	266	124	47	96	
Matrix Evaluasi							
Precision (%)	52,26	42,75	48,38	33,33	45,81	47,43	44,99
Recall (%)	96,15	79,67	33,50	24,39	60,17	87,39	63,54

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
F1-Score (%)	67,72	55,64	39,59	28,17	52,01	61,49	50,77
Subset Accuracy (%)	38,15						
Hamming Loss (%)	22,23						

Berdasarkan Tabel 4.21, diperoleh nilai *subset accuracy* sebesar 38,15% dan *hamming loss* sebesar 22,23%. Dibandingkan dengan Skenario 20, nilai *subset accuracy* relatif stabil dengan sedikit peningkatan, sementara nilai *hamming loss* berada pada kisaran yang hampir sama. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada konfigurasi ini belum memberikan peningkatan yang signifikan terhadap ketepatan prediksi kombinasi label, namun tetap mampu menjaga tingkat kesalahan per label pada level yang relatif terkendali.

Ditinjau dari metrik per label, peningkatan nilai *threshold* berdampak pada penurunan nilai *recall* pada beberapa kategori, terutama *HS_Group* dan *HS_Religion*, yang masing-masing memiliki nilai *recall* sebesar 33,50% dan 24,39%. Penurunan ini menunjukkan bahwa sistem menjadi lebih selektif dan tidak lagi menetapkan label positif secara luas seperti pada kondisi *threshold* yang lebih rendah. Meskipun demikian, label *HS* dan *HS_Other* masih mempertahankan nilai *recall* yang tinggi, masing-masing sebesar 96,15% dan 87,39%, sehingga kemampuan sistem dalam mendeteksi ujaran kebencian utama tetap terjaga.

Seiring dengan meningkatnya selektivitas sistem, nilai *precision* pada Skenario 21 berada pada tingkat yang relatif moderat. Berdasarkan Tabel 4.21, diperoleh nilai *macro precision* sebesar 44,99%. Nilai ini menunjukkan bahwa

sebagian prediksi positif yang dihasilkan sistem sudah lebih tepat dibandingkan skenario dengan *threshold* rendah, meskipun masih terdapat kesalahan prediksi positif pada beberapa label.

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut berdampak pada nilai *F1-score*. Berdasarkan Tabel 4.21, diperoleh nilai *macro F1-score* sebesar 50,77%. Nilai ini sedikit lebih rendah dibandingkan Skenario 20, yang mengindikasikan bahwa peningkatan nilai *threshold* menjadi 0,3 pada konfigurasi ini cenderung menurunkan sensitivitas sistem secara lebih dominan dibandingkan peningkatan ketepatan prediksi.

Analisis *confusion matrix* pada Tabel 4.21 memperkuat temuan tersebut. Jumlah *false positive* pada beberapa label mengalami penurunan, yang menunjukkan peningkatan selektivitas sistem. Namun, jumlah *false negative* meningkat cukup signifikan pada label *HS_Group* dan *HS_Religion*, yang menjadi penyebab utama turunnya nilai *recall* dan *F1-score* pada kedua kategori tersebut.

Secara keseluruhan, hasil pengujian pada Skenario Uji 21 menunjukkan bahwa peningkatan nilai *threshold* menjadi 0,3 pada konfigurasi *LSTM* dengan 128 unit dan *embedding* 100 dimensi menghasilkan sistem yang lebih selektif, namun dengan konsekuensi berkurangnya kemampuan dalam menangkap beberapa kategori ujaran kebencian secara konsisten. Kondisi ini menyebabkan nilai *subset accuracy* tidak meningkat secara signifikan dan nilai *F1-score* cenderung menurun. Dengan demikian, Skenario 21 menunjukkan bahwa pada konfigurasi ini, peningkatan nilai *threshold* yang terlalu tinggi dapat mengurangi sensitivitas sistem

dan belum menghasilkan keseimbangan optimal antara ketepatan dan cakupan prediksi.

4.1.22 Hasil Skenario Uji 22

Skenario Uji 22 merupakan pengujian lanjutan yang bertujuan untuk mengamati pengaruh peningkatan *batch size* pada konfigurasi dengan *embedding* berukuran 100 dimensi dan *LSTM* dengan 128 unit. Pada skenario ini digunakan *batch size* sebesar 64 dengan nilai *threshold* yang kembali diturunkan menjadi 0,1. Penurunan nilai *threshold* dilakukan untuk melihat kembali perilaku sistem pada kondisi yang permisif setelah sebelumnya diuji pada *threshold* yang lebih tinggi. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.22.

Tabel 4.22 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 22

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1109	699	347	134	99	733	
TN	332	576	1402	2124	2316	741	
FP	1184	1340	832	346	200	1132	
FN	9	19	53	30	19	28	
Matrix Evaluasi							
Precision (%)	48,36	34,28	29,43	27,92	33,11	39,30	35,40
Recall (%)	99,19	97,35	86,75	81,71	83,90	96,32	90,87
F1-Score (%)	65,02	50,71	43,95	41,61	47,48	55,83	50,77
Subset Accuracy (%)	19,25						
Hamming Loss (%)	32,85						

Berdasarkan Tabel 4.22, diperoleh nilai *subset accuracy* sebesar 19,25% dan *hamming loss* sebesar 32,85%. Jika dibandingkan dengan Skenario 21, nilai

subset accuracy mengalami penurunan yang cukup signifikan, sementara nilai *hamming loss* kembali meningkat. Hasil ini menunjukkan bahwa kombinasi *batch size* yang lebih besar dengan nilai *threshold* rendah belum mampu meningkatkan ketepatan prediksi kombinasi label.

Ditinjau dari metrik per label, sistem kembali menunjukkan nilai *recall* yang sangat tinggi dengan *macro recall* mencapai 90,87%. Label *HS* dan *HS_Individual* masing-masing memiliki nilai *recall* sebesar 99,19% dan 97,35%, yang menunjukkan bahwa hampir seluruh ujaran kebencian pada data uji berhasil terdeteksi. Tingginya nilai *recall* ini selaras dengan penggunaan nilai *threshold* yang rendah, sehingga sistem cenderung menetapkan label positif secara luas.

Namun demikian, capaian *recall* yang tinggi tersebut tidak diimbangi oleh nilai *precision*. Berdasarkan Tabel 4.22, diperoleh nilai *macro precision* sebesar 35,40%, yang menunjukkan bahwa sebagian besar prediksi positif yang dihasilkan sistem masih belum tepat. Kondisi ini terlihat jelas pada seluruh label, khususnya *HS_Group* dan *HS_Religion*, yang memiliki nilai *precision* rendah meskipun nilai *recall*-nya tinggi. Hal ini mengindikasikan bahwa sistem kembali menghasilkan banyak prediksi positif yang tidak sesuai dengan label sebenarnya.

Ketidakseimbangan antara nilai *precision* dan *recall* tersebut berdampak langsung pada nilai *F1-score*. Berdasarkan Tabel 4.22, diperoleh nilai *macro F1-score* sebesar 50,77%, yang relatif serupa dengan skenario-skenario permisif sebelumnya. Nilai ini menunjukkan bahwa meskipun sistem sangat sensitif dalam mendeteksi keberadaan ujaran kebencian, ketepatan prediksi masih menjadi permasalahan utama.

Analisis *confusion matrix* pada Tabel 4.22 memperkuat temuan tersebut. Jumlah *false negative* pada seluruh label relatif kecil, yang menjelaskan tingginya nilai *recall*. Sebaliknya, jumlah *false positive* sangat besar pada hampir seluruh label, seperti pada label *HS*, *HS_Individual*, dan *HS_Other*. Tingginya jumlah *false positive* inilah yang menjadi penyebab utama rendahnya nilai *precision*, rendahnya *subset accuracy*, serta meningkatnya nilai *hamming loss*.

Secara keseluruhan, hasil pengujian pada Skenario Uji 22 menunjukkan bahwa peningkatan *batch size* menjadi 64 pada konfigurasi *LSTM* dengan 128 unit dan *embedding* 100 dimensi, ketika dipadukan dengan nilai *threshold* yang rendah, kembali menghasilkan sistem yang sangat permisif. Kondisi ini menyebabkan sistem mampu menangkap hampir seluruh ujaran kebencian, namun dengan tingkat kesalahan prediksi positif yang tinggi. Dengan demikian, Skenario 22 menegaskan bahwa peningkatan *batch size* perlu disertai dengan penyesuaian nilai *threshold* agar diperoleh keseimbangan yang lebih baik antara sensitivitas dan ketepatan prediksi pada skenario-skenario berikutnya.

4.1.23 Hasil Skenario Uji 23

Skenario Uji 23 merupakan pengujian lanjutan dari Skenario 22 dengan tetap menggunakan *embedding* berukuran 100 dimensi dan *LSTM* dengan 128 unit, serta *batch size* sebesar 64. Perbedaan utama pada skenario ini terletak pada peningkatan nilai *threshold* menjadi 0,2. Penyesuaian ini dilakukan untuk mengurangi sifat permisif sistem pada Skenario 22 serta untuk mengevaluasi perbaikan keseimbangan antara ketepatan dan sensitivitas prediksi. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.23.

Tabel 4. 23 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 23

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1070	650	288	105	82	682	
TN	618	893	1753	2282	2394	1078	
FP	898	1023	481	188	122	795	
FN	48	68	112	59	36	79	
Matrix Evaluasi							
Precision (%)	54,37	38,85	37,45	35,84	40,20	46,17	42,15
Recall (%)	95,71	90,53	72,00	64,02	69,49	89,62	80,23
F1-Score (%)	69,35	54,37	49,27	45,95	50,93	60,95	55,14
Subset Accuracy (%)	34,97						
Hamming Loss (%)	24,73						

Berdasarkan Tabel 4.23, diperoleh nilai *subset accuracy* sebesar 34,97% dan *hamming loss* sebesar 24,73%. Dibandingkan dengan Skenario 22, nilai *subset accuracy* meningkat secara signifikan, sementara nilai *hamming loss* menurun. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada konfigurasi dengan *batch size* yang lebih besar mulai memperbaiki ketepatan prediksi kombinasi label dan mengurangi kesalahan prediksi per label.

Ditinjau dari metrik per label, sistem masih mempertahankan nilai *recall* yang relatif tinggi dengan *macro recall* sebesar 80,23%. Label *HS* dan *HS_Other* masing-masing memiliki nilai *recall* sebesar 95,71% dan 89,62%, yang menunjukkan bahwa sebagian besar ujaran kebencian utama tetap dapat terdeteksi dengan baik. Namun, dibandingkan dengan kondisi *threshold* rendah, nilai *recall*

pada beberapa label menurun, yang mencerminkan meningkatnya selektivitas sistem dalam menetapkan label positif.

Seiring dengan meningkatnya selektivitas tersebut, nilai *precision* pada Skenario 23 menunjukkan perbaikan. Berdasarkan Tabel 4.23, diperoleh nilai *macro precision* sebesar 42,15%, lebih tinggi dibandingkan Skenario 22. Peningkatan ini terlihat pada hampir seluruh label, yang menandakan berkurangnya jumlah prediksi positif yang tidak tepat (*false positive*), meskipun nilainya masih bervariasi antar kategori.

Perubahan keseimbangan antara nilai *precision* dan *recall* tersebut berdampak pada nilai *F1-score*. Berdasarkan Tabel 4.23, diperoleh nilai *macro F1-score* sebesar 55,14%, yang menunjukkan peningkatan dibandingkan Skenario 22. Nilai ini mengindikasikan bahwa kualitas prediksi sistem secara keseluruhan menjadi lebih stabil setelah penyesuaian nilai *threshold*.

Analisis *confusion matrix* pada Tabel 4.23 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 22, sementara jumlah *false negative* meningkat secara moderat. Pola ini menunjukkan bahwa sistem mulai mengurangi kesalahan prediksi positif dengan konsekuensi berkurangnya sensitivitas pada sebagian kategori, namun masih berada pada tingkat yang dapat diterima.

Secara keseluruhan, hasil pengujian pada Skenario Uji 23 menunjukkan bahwa peningkatan nilai *threshold* menjadi 0,2 pada konfigurasi *LSTM* dengan 128 unit, *embedding* 100 dimensi, dan *batch size* 64 mampu memperbaiki keseimbangan antara ketepatan dan sensitivitas prediksi dibandingkan Skenario 22.

Peningkatan nilai *subset accuracy* dan penurunan *hamming loss* menegaskan adanya perbaikan performa klasifikasi *Multi-Label*, meskipun hasilnya belum seoptimal konfigurasi dengan nilai *threshold* yang lebih tinggi pada beberapa skenario sebelumnya.

4.1.24 Hasil Skenario Uji 24

Skenario Uji 24 merupakan pengujian terakhir pada penelitian ini yang menggunakan *embedding* berukuran 100 dimensi, *LSTM* dengan 128 unit, serta *batch size* sebesar 64. Perbedaan utama pada skenario ini terletak pada penggunaan nilai *threshold* tertinggi, yaitu 0,3. Pengaturan ini bertujuan untuk menguji tingkat selektivitas maksimum sistem dalam menetapkan label positif serta melihat dampaknya terhadap keseimbangan antara ketepatan dan sensitivitas prediksi. Laporan hasil klasifikasi pada skenario ini disajikan pada Tabel 4.24.

Tabel 4.24 Laporan Hasil Klasifikasi *Multi-Label* pada Skenario 24

	Label						***Metric***
	HS	*HS_ Individual*	*HS_ Group*	*HS_ Religion*	*HS_ Race*	*HS_ Other*	*Macro avg (%)*
Confusion Matrix							
TP	1015	556	232	39	68	624	
TN	757	1310	1893	2352	2445	1315	
FP	759	606	341	118	71	558	
FN	103	162	168	125	50	137	
Matrix Evaluasi							
Precision (%)	57,22	47,85	40,49	24,84	48,92	52,79	45,35
Recall (%)	90,79	77,44	58,00	23,78	57,63	82,00	64,94
F1-Score (%)	70,19	59,15	47,69	24,30	52,92	64,23	53,08
Subset Accuracy (%)	42,98						
Hamming Loss (%)	20,24						

Berdasarkan Tabel 4.24 diperoleh nilai *subset accuracy* sebesar 42,98% dan *hamming loss* sebesar 20,24%. Dibandingkan dengan Skenario 23, nilai *subset accuracy* mengalami peningkatan, sementara nilai *hamming loss* menurun. Hasil ini menunjukkan bahwa peningkatan nilai *threshold* pada konfigurasi ini mampu memperbaiki ketepatan prediksi kombinasi label dan mengurangi kesalahan prediksi per label secara keseluruhan.

Ditinjau dari metrik per label, peningkatan nilai *threshold* menyebabkan penurunan nilai *recall* pada beberapa kategori, terutama *HS_Religion* yang hanya memiliki nilai *recall* sebesar 23,78%. Hal ini menunjukkan bahwa sistem menjadi sangat selektif dalam menetapkan label pada kategori tersebut. Namun demikian, label *HS* dan *HS_Other* masih mempertahankan nilai *recall* yang relatif memadai, masing-masing sebesar 90,79% dan 82,00%, sehingga kemampuan sistem dalam mendeteksi ujaran kebencian utama tetap terjaga.

Seiring dengan meningkatnya selektivitas sistem, nilai *precision* pada Skenario 24 berada pada tingkat yang relatif lebih baik dibandingkan beberapa skenario sebelumnya. Berdasarkan Tabel 4.24, diperoleh nilai *macro precision* sebesar 45,35%. Nilai ini menunjukkan bahwa proporsi prediksi positif yang tepat semakin meningkat, meskipun masih terdapat variasi antar kategori. Label *HS* dan *HS_Other* menunjukkan nilai *precision* yang cukup tinggi, yang berkontribusi positif terhadap kestabilan hasil klasifikasi.

Keseimbangan antara nilai *precision* dan *recall* tersebut tercermin pada nilai *F1-score*. Berdasarkan Tabel 4.24, diperoleh nilai *macro F1-score* sebesar 53,08%. Nilai ini menunjukkan bahwa kualitas prediksi sistem secara keseluruhan berada

pada tingkat yang cukup stabil, meskipun tidak setinggi skenario dengan konfigurasi *threshold* 0,3 dan jumlah unit *LSTM* yang lebih kecil pada kelompok pengujian sebelumnya.

Analisis *confusion matrix* pada Tabel 4.24 memperkuat temuan tersebut. Jumlah *false positive* pada hampir seluruh label mengalami penurunan dibandingkan Skenario 23, yang menunjukkan peningkatan ketepatan prediksi positif. Namun, jumlah *false negative* meningkat cukup signifikan pada beberapa label, khususnya *HS_Religion*, yang menjadi penyebab utama menurunnya nilai *recall* pada kategori tersebut.

Secara keseluruhan, hasil pengujian pada Skenario Uji 24 menunjukkan bahwa penggunaan nilai *threshold* tertinggi pada konfigurasi *embedding* 100 dimensi, LSTM 128 unit, dan *batch size* 64 menghasilkan sistem yang sangat selektif dalam menetapkan label positif. Kondisi ini mampu meningkatkan ketepatan prediksi dan nilai *subset accuracy*, namun dengan konsekuensi berkurangnya kemampuan sistem dalam menangkap beberapa kategori ujaran kebencian secara konsisten. Dengan demikian, Skenario 24 menegaskan bahwa peningkatan nilai *threshold* perlu dipertimbangkan secara hati-hati agar tidak menurunkan sensitivitas sistem secara berlebihan, sekaligus menjadi penutup yang menunjukkan batas keseimbangan optimal antara ketepatan dan cakupan prediksi dalam penelitian ini.

4.2 Pembahasan

Berdasarkan hasil uji coba yang telah dijabarkan secara rinci pada sub-bab 4.1.1 hingga 4.1.24, bagian ini menyajikan analisis perbandingan menyeluruh.

Analisis ini bertujuan untuk mengidentifikasi konfigurasi *hyperparameter* yang paling optimal serta memahami pengaruh dari setiap parameter dan karakteristik dataset terhadap performa model.

4.2.1 Analisis Performa Model Terbaik

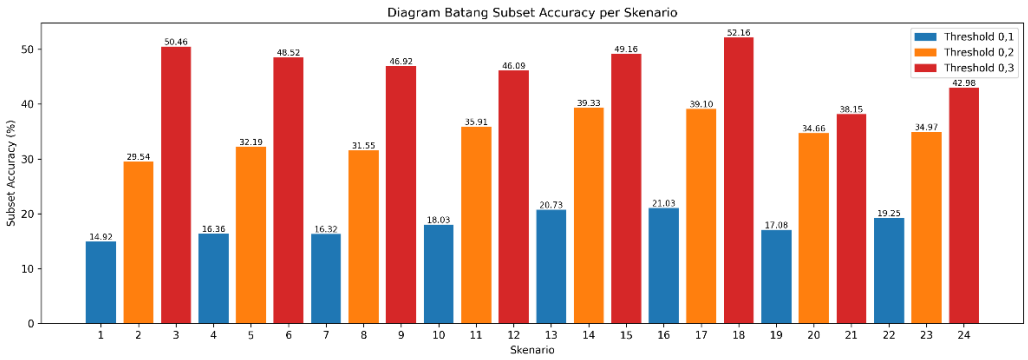
Pemilihan model terbaik pada penelitian ini dilakukan melalui evaluasi menyeluruh terhadap 24 skenario pengujian. Setiap skenario menggunakan kombinasi *embedding dimension*, jumlah unit LSTM, *batch size*, dan nilai *threshold* yang berbeda, sehingga menghasilkan variasi performa pada seluruh metrik evaluasi. Proses penilaian tidak hanya berfokus pada satu indikator, tetapi mempertimbangkan lima metrik utama pada klasifikasi multilabel, yaitu *subset accuracy*, *hamming loss*, *precision*, *recall*, dan *F1-score*. Oleh karena itu, penentuan skenario terbaik tidak dilakukan hanya dengan mempertimbangkan satu metrik tertentu, melainkan melalui evaluasi menyeluruh terhadap keseimbangan performa pada seluruh metrik tersebut. Rekapitulasi lengkap performa seluruh skenario disajikan dalam Tabel 4.25.

Tabel 4.25 Rekapitulasi Performa 24 Skenario Pengujian

Skenario	**Konfigurasi (d-H-B-t)**	***Subset Accuracy (%)***	***Hamming Loss (%)***	***Precision (%) (Macro Avg)***	***Recall (%) (Macro Avg)***	***F1-Score (%) (Macro Avg)***
1	50d-64H-32B-0.1t	14,92	36,15	33,68	91,94	49,00
2	50d-64H-32B-0.2t	29,54	26,82	40,76	84,29	54,70
3	50d-64H-32B-0.3t	50,46	17,63	56,59	69,23	60,62
4	50d-64H-64B-0.1t	16,36	35,24	35,70	87,64	49,97
5	50d-64H-64B-0.2t	32,19	25,22	41,27	74,41	52,73
6	50d-64H-64B-0.3t	48,52	17,85	49,90	61,95	54,85
7	50d-128H-32B-0.1t	16,32	35,45	33,88	89,24	48,86
8	50d-128H-32B-0.2t	31,55	25,60	43,65	82,67	56,76
9	50d-128H-32B-0.3t	46,92	19,37	47,99	65,58	54,79
10	50d-128H-64B-0.1t	18,03	34,38	34,86	89,52	49,96
11	50d-128H-64B-0.2t	35,91	22,84	44,32	80,05	56,86
12	50d-128H-64B-0.3t	46,09	19,77	51,71	58,36	48,18

Skenario	Konfigurasi (d-H-B-t)	*Subset Accuracy (%)*	*Hamming Loss (%)*	*Precision (%) (Macro Avg)*	*Recall (%) (Macro Avg)*	*F1-Score (%) (Macro Avg)*
13	100d-64H-32B-0.1t	20,73	32,87	35,33	90,81	50,68
14	100d-64H-32B-0.2t	39,33	21,20	48,59	82,47	60,81
15	100d-64H-32B-0.3t	49,16	17,71	53,04	69,44	59,54
16	100d-64H-64B-0.1t	21,03	31,81	36,76	88,87	51,78
17	100d-64H-64B-0.2t	39,10	22,17	46,53	78,54	58,10
18	100d-64H-64B-0.3t	52,16	17,15	56,56	68,59	61,44
19	100d-128H-32B-0.1t	17,08	33,40	35,56	91,60	51,05
20	100d-128H-32B-0.2t	34,66	23,70	42,80	82,14	56,20
21	100d-128H-32B-0.3t	38,15	22,23	44,99	63,54	50,77
22	100d-128H-64B-0.1t	19,25	32,85	35,40	90,87	50,77
23	100d-128H-64B-0.2t	34,97	24,73	42,15	80,23	55,14
24	100d-128H-64B-0.3t	42,98	20,24	45,35	64,94	53,08

Untuk memperjelas perbedaan performa antarskenario, hasil pada Tabel 4.25 divisualisasikan dalam bentuk diagram batang sebagaimana ditunjukkan pada Gambar 4.1. Visualisasi ini menyajikan perbandingan nilai *subset accuracy*, *hamming loss*, *macro precision*, *macro recall*, dan *macro F1-score* untuk setiap skenario pengujian.



(a)

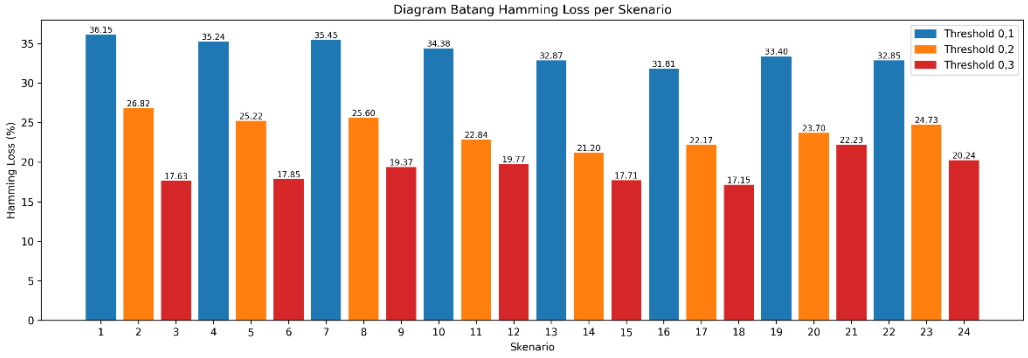


Diagram Batang Hamming Loss per Skenario
Threshold 0,1
Threshold 0,2
Threshold 0,3
Hamming Loss (%)
Skenario
36.15
26.82
17.63
35.24
25.22
17.85
35.45
25.60
19.37
34.38
22.84
19.77
32.87
21.20
17.71
31.81
22.17
17.15
33.40
23.70
22.23
32.85
24.73
20.24

(b)

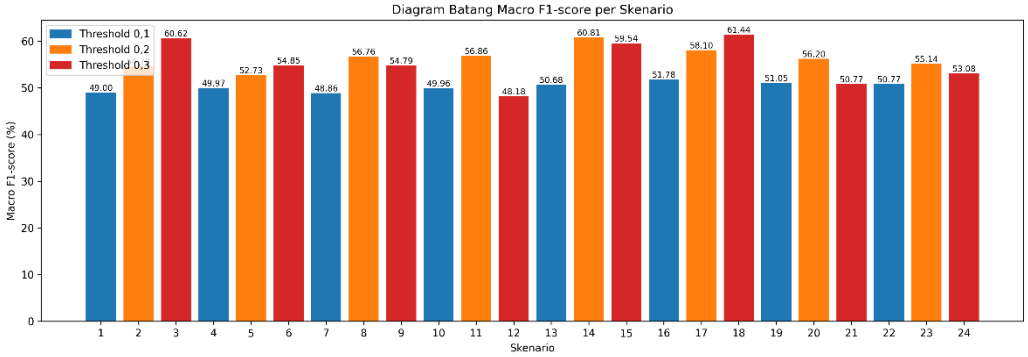


Diagram Batang Macro F1-score per Skenario
Threshold 0,1
Threshold 0,2
Threshold 0,3
Macro F1-score (%)
Skenario
49.00
60.62
49.97
52.73
54.85
48.86
56.76
54.79
49.96
56.86
48.18
50.68
60.81
59.54
51.78
58.10
61.44
51.05
56.20
50.77
50.77
55.14
53.08

(c)

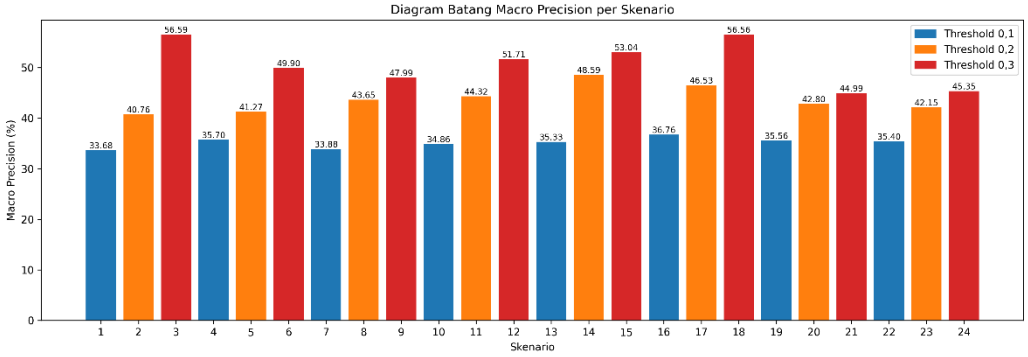
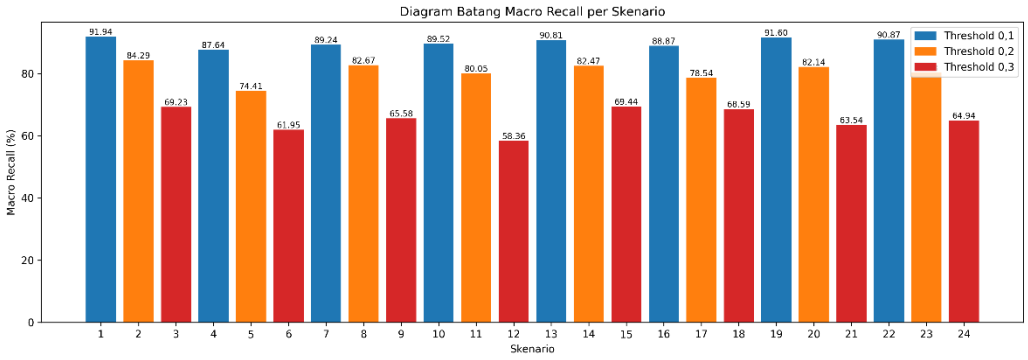


Diagram Batang Macro Precision per Skenario
Threshold 0,1
Threshold 0,2
Threshold 0,3
Macro Precision (%)
Skenario
33.68
40.76
56.59
35.70
41.27
49.90
33.88
43.65
47.99
34.86
44.32
51.71
35.33
48.59
53.04
36.76
46.53
56.56
35.56
42.80
44.99
35.40
42.15
45.35

(d)



(e)

Gambar 4.1 Diagram Batang (a) *'Subset Accuracy'*, (b) *'Hamming Loss'*, (c) *'Macro F1-score'*, (d) *'Macro Precision'*, dan (e) *'Macro Recall'* per Skenario

Pada Gambar 4.1, setiap skenario direpresentasikan oleh tiga batang berwarna yang menunjukkan variasi nilai *threshold*, yaitu warna biru untuk *threshold* 0,1, warna oranye untuk *threshold* 0,2, dan warna merah untuk *threshold* 0,3. Pola yang terlihat menunjukkan bahwa peningkatan nilai *threshold* cenderung meningkatkan *subset accuracy* dan *precision*, serta menurunkan *hamming loss*. Pola ini mengindikasikan bahwa model menjadi lebih selektif dalam mengaktifkan label, sehingga hanya label dengan tingkat keyakinan yang lebih tinggi yang diprediksi sebagai positif.

Berdasarkan hasil numerik pada Tabel 4.25 yang menandai skenario terbaik dengan *highlight* warna kuning, serta didukung oleh pola visualisasi pada Gambar 4.1, dapat disimpulkan bahwa Skenario 18 dengan konfigurasi 100d–64H–64B–0.3t merupakan skenario dengan kinerja paling baik dan paling konsisten. Penetapan ini didasarkan pada evaluasi menyeluruh terhadap seluruh metrik yang digunakan, mulai dari ketepatan prediksi kombinasi label, tingkat kesalahan per label, hingga keseimbangan antara *precision* dan *recall*.

Dari sisi *subset accuracy*, Skenario 18 mencapai nilai tertinggi dibandingkan seluruh skenario lainnya. Nilai ini menunjukkan bahwa model memiliki kemampuan paling baik dalam memprediksi kombinasi label secara lengkap pada satu instance. Kemampuan tersebut menjadi tantangan utama dalam klasifikasi multilabel karena adanya ketergantungan antarlabel, seperti hubungan antara label *HS*, *HS_Individual*, dan *HS_Group*.

Pada metrik *hamming loss*, Skenario 18 mencatat nilai terendah bersama dengan Skenario 12. Rendahnya nilai *hamming loss* menunjukkan bahwa model menghasilkan jumlah kesalahan prediksi per label yang lebih sedikit, baik berupa *false positive* maupun *false negative*. Temuan ini mengindikasikan bahwa model mampu menjaga ketepatan prediksi tanpa mengabaikan sensitivitas terhadap label yang relevan.

Dari perspektif *precision*, nilai *macro precision* pada Skenario 18 tergolong tinggi dan mendekati nilai tertinggi di antara seluruh skenario. Hal ini menunjukkan bahwa model tidak memberikan prediksi positif secara berlebihan, sehingga mampu meminimalkan *false positive*. Kondisi ini penting dalam konteks deteksi ujaran kebencian agar teks yang bersifat netral tidak salah diklasifikasikan sebagai ujaran bermasalah.

Sementara itu, nilai *macro recall* menunjukkan bahwa model cukup mampu menangkap instance positif tanpa menghasilkan aktivasi label yang berlebihan. Nilai *recall* yang moderat mencerminkan bahwa model bersifat sensitif terhadap keberadaan label positif, namun tetap berhati-hati dalam proses aktivasi label.

Keseimbangan antara *precision* dan *recall* tercermin pada nilai *macro F1-score* yang merupakan yang tertinggi di antara seluruh skenario. Karena *F1-score* merepresentasikan keseimbangan antara ketepatan dan cakupan deteksi, metrik ini menjadi indikator penting dalam evaluasi klasifikasi multilabel. Konsistensi nilai *F1-score* pada Skenario 18 menunjukkan bahwa model tidak hanya akurat, tetapi juga stabil dalam menangkap pola ujaran kebencian pada berbagai variasi label, termasuk label minoritas.

Berdasarkan keseluruhan hasil tersebut, Skenario 18 dinilai sebagai model terbaik dalam penelitian ini karena memenuhi beberapa kriteria utama, yaitu:

1. Memiliki *Subset accuracy* tertinggi, menunjukkan prediksi kombinasi label paling tepat,
2. *Hamming loss* terendah, menandakan jumlah kesalahan per label paling sedikit,
3. *Precision* stabil dan tidak berlebihan, mengurangi risiko false positive,
4. *Recall seimbang*, tidak memicu sensitivitas berlebih,
5. *Macro F1-score tertinggi*, menandakan model paling seimbang dan stabil.

Dengan demikian, konfigurasi 100d–64H–64B–0.3t ditetapkan sebagai model terbaik pada penelitian ini.

Untuk memberikan gambaran mengenai kinerja model pada data nyata, Gambar 4.2 menampilkan lima contoh hasil klasifikasi multilabel yang dihasilkan oleh Skenario 18.

>> SAMPEL DATASET AWAL (5 Baris Pertama CSV)

Tweet Asli	Label Aktual	Label Prediksi	Status
- disaat semua cowok berusaha melacak perhatian gue. loe lantas remehkan perhatian yg gue kasih khusus ke elo. basic elo cowok bego ! ! !'	{1 1 0 0 0 1}	{0 0 0 0 0 0}	Tidak Sesuai
RT USER: USER siapa yang telat ngasih tau elu?edan sarap gue bergaul dengan cigax jifla calis sama siapa noh licew juga'	{0 0 0 0 0 0}	{1 1 0 0 0 1}	Tidak Sesuai
41. Kadang aku berfikir, kenapa aku tetap percaya pada Tuhan padahal aku selalu jatuh berkali-kali. Kadang aku merasa Tuhan itu ninggalkan aku sendirian. Ketika orangtuaku berencana berpisah, ketika kakakku lebih memilih jadi Kristen. Ketika aku anak ter	{0 0 0 0 0 0}	{0 0 0 0 0 0}	Sesuai
USER USER AKU ITU AKU\n\nKU TAU MATAMU SIPIT TAPI DILIAT DARI MANA ITU AKU'	{0 0 0 0 0 0}	{0 0 0 0 0 0}	Sesuai
USER USER Kaum cebong kapir udah keliatan dongoknya dari awal tambah dongok lagi hahahah'	{1 0 1 1 0 0}	{1 1 1 0 0 1}	Tidak Sesuai

Gambar 4.2 Lima dataset teratas hasil klasifikasi *Multi-Label* dari Skenario 18

Pada Gambar 4.2 ditunjukkan perbandingan antara label aktual dan label prediksi, serta status kecocokan hasil prediksi. Beberapa *tweet* berhasil diklasifikasikan dengan tepat, terutama pada kasus ujaran kebencian dengan pola bahasa yang jelas. Namun, masih terdapat prediksi yang kurang tepat pada *tweet* dengan konteks yang lebih halus atau kategori yang jarang muncul. Meskipun hanya lima contoh yang ditampilkan, hasil ini menunjukkan bahwa model bekerja dengan cukup baik pada data nyata, namun tetap memiliki keterbatasan dalam menangani kasus tertentu.

Meskipun Skenario 18 menunjukkan performa terbaik berdasarkan evaluasi pada data uji, diperlukan pengujian tambahan untuk memastikan bahwa performa tersebut tidak bergantung pada satu pembagian data tertentu. Oleh karena itu, dilakukan evaluasi lanjutan menggunakan *K-Fold Cross Validation* dengan nilai *k* = 5 pada data latih untuk menilai kestabilan dan konsistensi performa model. Evaluasi ini bertujuan untuk memastikan bahwa performa tinggi yang diperoleh tidak hanya disebabkan oleh satu pembagian data tertentu, melainkan mampu dipertahankan pada berbagai variasi subset data latih.

Pada tahap ini, dataset dibagi menjadi data latih sebesar 80% dan data uji sebesar 20%. Proses *K-Fold Cross Validation* hanya diterapkan pada data latih, yang kemudian dibagi menjadi lima fold berukuran relatif sama. Pada setiap iterasi, satu fold digunakan sebagai data validasi dan empat fold lainnya digunakan sebagai data latih. Data uji tetap dipertahankan sebagai *hold-out test set* untuk mengevaluasi kemampuan generalisasi model terhadap data yang benar-benar baru.

Hasil evaluasi dari kelima iterasi *K-Fold Cross Validation* kemudian dirata-ratakan dan disertai dengan nilai standar deviasi untuk menggambarkan tingkat variasi performa antar fold. Rekapitulasi hasil *K-Fold Cross Validation* pada Skenario 18 disajikan pada Tabel 4.26.

Tabel 4.26 Rekapitulasi hasil *K-Fold Cross Validation* pada Skenario 18

Fold	***Subset Accuracy (%)***	***Hamming Loss (%)***	***Macro Precision (%)***	***Macro Recall (%)***	***Macro F1-Score (%)***
1	54,11	17,14	50,56	60,13	53,94
2	54,25	17,45	52,66	56,16	51,99
3	51,35	17,24	52,46	63,54	57,30
4	54,67	16,56	47,66	46,28	45,94
5	53,92	16,08	52,98	65,55	58,54
Rata-rata (Mean)	53,68	16,89	51,26	58,33	53,54
Standar Deviasi (Std)	1,32	0,56	2,23	7,62	4,99

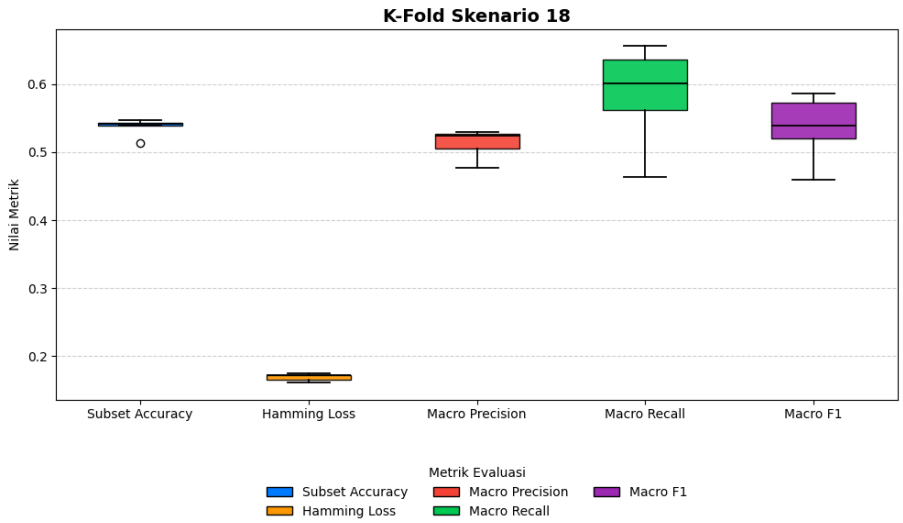
Berdasarkan Tabel 4.26, hasil evaluasi menggunakan *K-Fold Cross Validation* dengan nilai $k = 5$ pada data latih (80%) untuk Skenario 18, diperoleh nilai *Subset Accuracy* rata-rata sebesar 53,68% dengan standar deviasi 1,32%. Nilai standar deviasi yang relatif kecil menunjukkan bahwa performa model cenderung stabil pada setiap pembagian data latih dan data validasi. Hal ini mengindikasikan

bahwa model tidak bergantung pada satu pola pembagian data tertentu dan memiliki konsistensi performa yang baik selama proses pelatihan.

Selain itu, nilai *Hamming Loss* rata-rata sebesar 16,89% dengan standar deviasi 0,56% menunjukkan tingkat kesalahan prediksi per label yang relatif rendah dan konsisten di seluruh fold. Rendahnya variasi nilai *Hamming Loss* antar fold memperkuat temuan bahwa model mampu melakukan prediksi label secara seimbang pada berbagai subset data latih tanpa mengalami fluktuasi kesalahan yang signifikan.

Dari sisi performa klasifikasi, diperoleh nilai *Macro Precision* rata-rata sebesar 51,26%, *Macro Recall* sebesar 58,33%, dan *Macro F1-Score* sebesar 53,54%. Nilai *Macro F1-Score* yang cukup stabil dengan standar deviasi 4,99% menunjukkan bahwa model mampu menjaga keseimbangan antara *precision* dan *recall* pada seluruh label, termasuk label minoritas, meskipun distribusi data bersifat tidak seimbang. Variasi nilai *recall* yang cenderung lebih tinggi dibandingkan *precision* mengindikasikan bahwa model lebih sensitif dalam mendeteksi keberadaan label ujaran kebencian, yang sejalan dengan tujuan penelitian untuk meminimalkan kesalahan *false negative*.

Sebagai pelengkap analisis numerik pada Tabel 4.26, visualisasi boxplot pada Gambar 4.3 digunakan untuk menggambarkan sebaran nilai evaluasi model pada setiap metrik selama proses *5-Fold Cross Validation*. Visualisasi ini bertujuan untuk memberikan pemahaman yang lebih komprehensif mengenai kestabilan dan konsistensi performa model antar fold, yang tidak sepenuhnya dapat direpresentasikan hanya melalui nilai rata-rata dan standar deviasi.



Gambar 4.3 *Boxplot* Hasil 5-*Fold Cross Validation* pada Skenario 18

Berdasarkan Gambar 4.3, metrik *subset accuracy* menunjukkan sebaran nilai yang relatif sempit, dengan satu titik *outlier* yang muncul di bawah rentang utama data. Keberadaan *outlier* ini selaras dengan hasil pada Tabel 4.26, di mana terdapat satu fold yang memiliki nilai *subset accuracy* sedikit lebih rendah dibandingkan fold lainnya. Kondisi tersebut disebabkan oleh karakteristik *subset accuracy* yang bersifat ketat, yaitu prediksi hanya dianggap benar apabila seluruh label pada suatu data berhasil diprediksi secara tepat. Akibatnya, variasi kecil dalam distribusi kesalahan label pada satu fold dapat memberikan dampak yang relatif lebih besar pada nilai *subset accuracy* dan teridentifikasi sebagai *outlier* pada boxplot. Meskipun demikian, rentang sebaran yang sempit menunjukkan bahwa variasi tersebut masih berada dalam batas yang wajar dan tidak mencerminkan ketidakstabilan model.

Pada metrik *hamming loss*, boxplot memperlihatkan sebaran nilai yang sangat terkonsentrasi tanpa adanya *outlier*. Temuan ini konsisten dengan nilai standar deviasi yang rendah pada Tabel 4.26, yang mengindikasikan bahwa tingkat

kesalahan prediksi per label relatif seragam di seluruh fold. Tidak ditemukannya *outlier* pada metrik ini menunjukkan bahwa model mampu mempertahankan konsistensi kesalahan per label pada berbagai pembagian data latih.

Sementara itu, boxplot untuk metrik *macro precision*, *macro recall*, dan *macro F1-score* menunjukkan variasi yang lebih besar dibandingkan *subset accuracy* dan *hamming loss*. Variasi tersebut mencerminkan adanya perbedaan sensitivitas model terhadap distribusi label pada masing-masing fold, terutama pada metrik *macro recall* yang memiliki sebaran paling lebar. Temuan ini sejalan dengan nilai standar deviasi *macro recall* pada Tabel 4.26, yang menunjukkan bahwa kemampuan model dalam menangkap seluruh label relevan dipengaruhi oleh variasi data latih. Meskipun demikian, nilai median *macro F1-score* yang tetap berada pada kisaran yang relatif tinggi mengindikasikan bahwa keseimbangan antara *precision* dan *recall* masih dapat dipertahankan secara konsisten.

Secara keseluruhan, hasil *K-Fold Cross Validation* memperkuat temuan sebelumnya bahwa Skenario 18 tidak hanya memberikan performa terbaik pada pengujian *hold-out test*, tetapi juga menunjukkan kestabilan dan konsistensi performa yang baik ketika diuji pada berbagai pembagian data latih. Dengan demikian, Skenario 18 dapat dinyatakan sebagai konfigurasi model yang paling optimal dan andal untuk digunakan pada tugas klasifikasi multilabel ujaran kebencian pada *tweet* berbahasa Indonesia.

4.2.2 Analisis Pengaruh *Hyperparameter*

Setelah model terbaik ditentukan, tahap selanjutnya adalah melihat bagaimana setiap hyperparameter memengaruhi performa model secara

keseluruhan. Empat hyperparameter utama *threshold*, jumlah unit LSTM, *dimensi embedding*, dan *batch size* mempunyai peran yang berbeda dalam proses pembelajaran dan hasil prediksi. Oleh karena itu, penting untuk menganalisis perubahan performa ketika nilai *hyperparameter* tersebut divariasikan.

Untuk keperluan analisis, seluruh hasil dari 24 skenario dikelompokkan kembali berdasarkan variasi masing-masing *hyperparameter*. Pada setiap kelompok, nilai rata-rata dari lima metrik evaluasi *(subset accuracy, hamming loss, macro precision, macro recall, dan macro F1-score)* dihitung dan divisualisasikan dalam bentuk diagram batang. Pendekatan ini membantu melihat pola perubahan performa secara lebih jelas dan terstruktur.

4.2.2.1 Pengaruh *Threshold* (t)

Analisis pengaruh *hyperparameter threshold* dilakukan untuk memahami bagaimana nilai ambang keputusan memengaruhi hasil klasifikasi *Multi-Label*. *Threshold* berfungsi sebagai batas probabilitas minimum yang harus dilampaui agar suatu label dinyatakan aktif. Oleh karena itu, perubahan nilai *threshold* secara langsung berdampak pada jumlah label yang diprediksi positif, tingkat kesalahan per label, serta ketepatan prediksi kombinasi label secara keseluruhan.

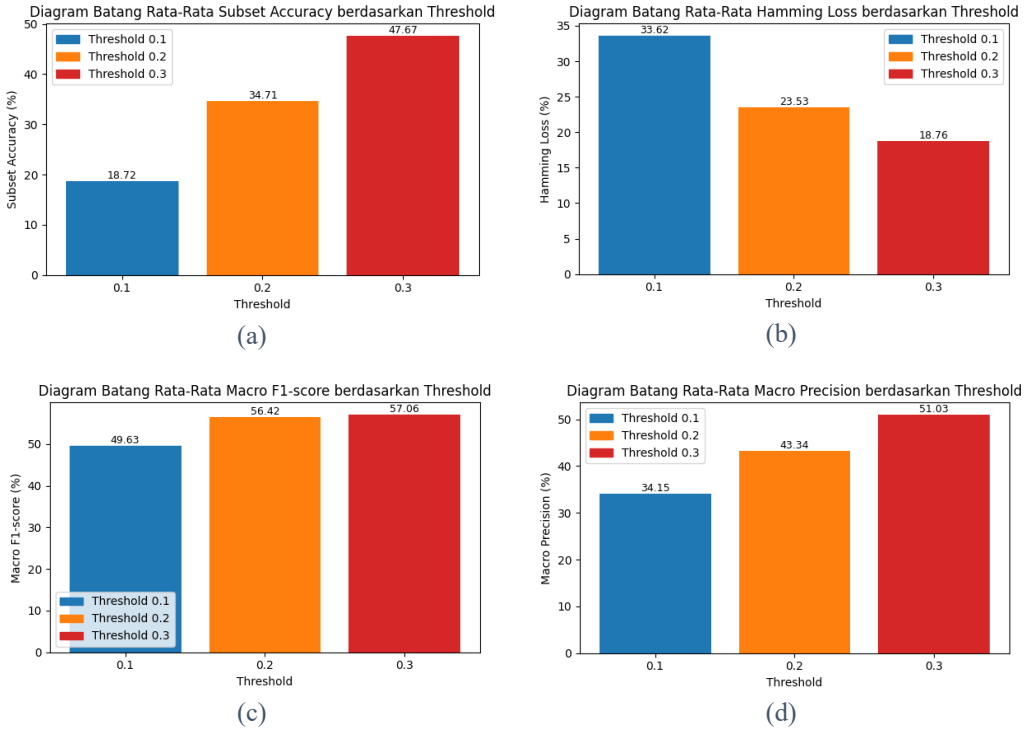
Pada penelitian ini, nilai *threshold* diuji pada tiga tingkat, yaitu 0,1, 0,2, dan 0,3. Setiap nilai *threshold* diterapkan secara konsisten pada berbagai kombinasi dimensi *embedding*, jumlah unit LSTM, dan *batch size*, sehingga menghasilkan total 24 skenario pengujian. Untuk memperoleh gambaran pengaruh *threshold* secara lebih objektif, seluruh hasil pengujian dikelompokkan berdasarkan nilai *threshold*, kemudian dihitung nilai rata-rata dari lima metrik evaluasi, yaitu *subset*

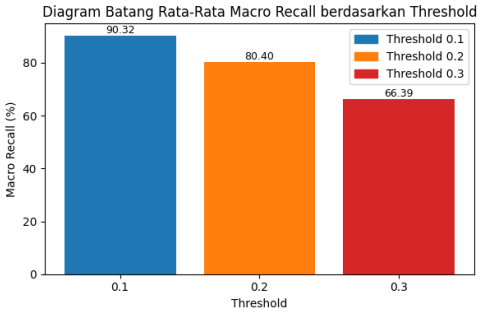
accuracy, *hamming loss*, *macro precision*, *macro recall*, dan *macro F1-score*. Hasil perhitungan tersebut disajikan pada Tabel 4.26.

Tabel 4.27 Rata-Rata Performa berdasarkan Nilai *Threshold*

Threshold	***Subset Accuracy (%)***	***Hamming Loss (%)***	***Macro Precision (%)***	***Macro Recall (%)***	***Macro F1-Score (%)***
0,1	18,72	33,62	34,15	90,32	49,63
0,2	34,71	23,53	43,34	80,40	56,42
0,3	47,67	18,76	51,03	66,39	57,06

Dan divisualisasikan dalam Gambar 4.27.





(e)

Gambar 4.4 Diagram Batang (a) *'Subset Accuracy'*, (b) *'Hamming Loss'*, (c) *'Macro F1-score'*, (d) *'Macro Precision'*, dan (e) *'Macro Recall'* per *Threshold*

Berdasarkan Tabel 4.26 terlihat bahwa penggunaan *threshold* 0,1 menghasilkan nilai *macro recall* tertinggi, yaitu sebesar 90,32%. Nilai ini menunjukkan bahwa sistem sangat sensitif dalam mendeteksi keberadaan label positif, sehingga sebagian besar ujaran kebencian berhasil teridentifikasi. Namun, sensitivitas yang tinggi tersebut diiringi oleh nilai *macro precision* yang rendah serta *hamming loss* yang tinggi. Hal ini mengindikasikan bahwa sistem cenderung terlalu permisif dalam mengaktifkan label, sehingga banyak terjadi kesalahan prediksi positif. Dampaknya, nilai *subset accuracy* pada *threshold* 0,1 relatif rendah karena kesalahan pada satu label saja menyebabkan prediksi kombinasi label dinilai tidak tepat.

Ketika nilai *threshold* dinaikkan menjadi 0,2, terlihat adanya perbaikan yang cukup signifikan pada hampir seluruh metrik evaluasi. Nilai *macro precision* meningkat, sementara *hamming loss* menurun dibandingkan *threshold* 0,1. Pada saat yang sama, *macro recall* masih berada pada tingkat yang relatif tinggi. Kondisi ini menunjukkan bahwa *threshold* 0,2 mampu membentuk keseimbangan antara ketepatan dan cakupan deteksi, yang tercermin dari peningkatan *macro F1-score* rata-rata menjadi 56,42% serta meningkatnya *subset accuracy*. Dengan demikian,

threshold 0,2 menghasilkan performa yang lebih stabil dan moderat tanpa kecenderungan terlalu permisif maupun terlalu selektif.

Penggunaan *threshold* 0,3 menunjukkan pola yang berbeda. Pada nilai ini, sistem menjadi lebih selektif dalam menetapkan label positif. Hal tersebut tercermin dari peningkatan nilai *macro precision* hingga 51,03% dan *subset accuracy* yang mencapai rata-rata tertinggi sebesar 47,67%, serta penurunan *hamming loss* menjadi 18,76%. Namun, peningkatan selektivitas ini juga menyebabkan penurunan *macro recall* menjadi 66,39%, yang menunjukkan bahwa sebagian ujaran kebencian dengan tingkat keyakinan rendah tidak lagi terdeteksi. Meskipun demikian, penurunan *recall* tersebut diimbangi oleh peningkatan ketepatan prediksi dan berkurangnya kesalahan per label.

Visualisasi diagram batang pada Gambar 4.2 memperkuat temuan pada Tabel 4.26. Terlihat bahwa peningkatan nilai *threshold* dari 0,1 ke 0,3 secara konsisten meningkatkan *subset accuracy* dan *macro precision*, serta menurunkan *hamming loss*. Sebaliknya, *macro recall* menunjukkan tren menurun seiring meningkatnya *threshold*. Pola ini menegaskan adanya hubungan *trade-off* antara ketepatan dan cakupan deteksi dalam klasifikasi *Multi-Label* yang sangat dipengaruhi oleh pemilihan nilai *threshold*.

Berdasarkan keseluruhan hasil tersebut, dapat disimpulkan bahwa *hyperparameter threshold* memiliki pengaruh yang sangat signifikan terhadap performa klasifikasi *Multi-Label*. *Threshold* 0,1 lebih menekankan sensitivitas, *threshold* 0,2 menghasilkan keseimbangan performa, sedangkan *threshold* 0,3 memberikan prediksi yang lebih selektif dan stabil. Dalam konteks penelitian ini,

threshold 0,3 dinilai paling optimal karena mampu meningkatkan *subset accuracy*, menurunkan *hamming loss*, serta menghasilkan nilai *macro F1-score* tertinggi secara rata-rata. Temuan ini juga selaras dengan hasil pemilihan model terbaik, di mana skenario dengan *threshold* 0,3 menunjukkan performa paling konsisten dibandingkan nilai *threshold* lainnya.

4.2.2.2 Pengaruh Pengaruh Unit LSTM (H)

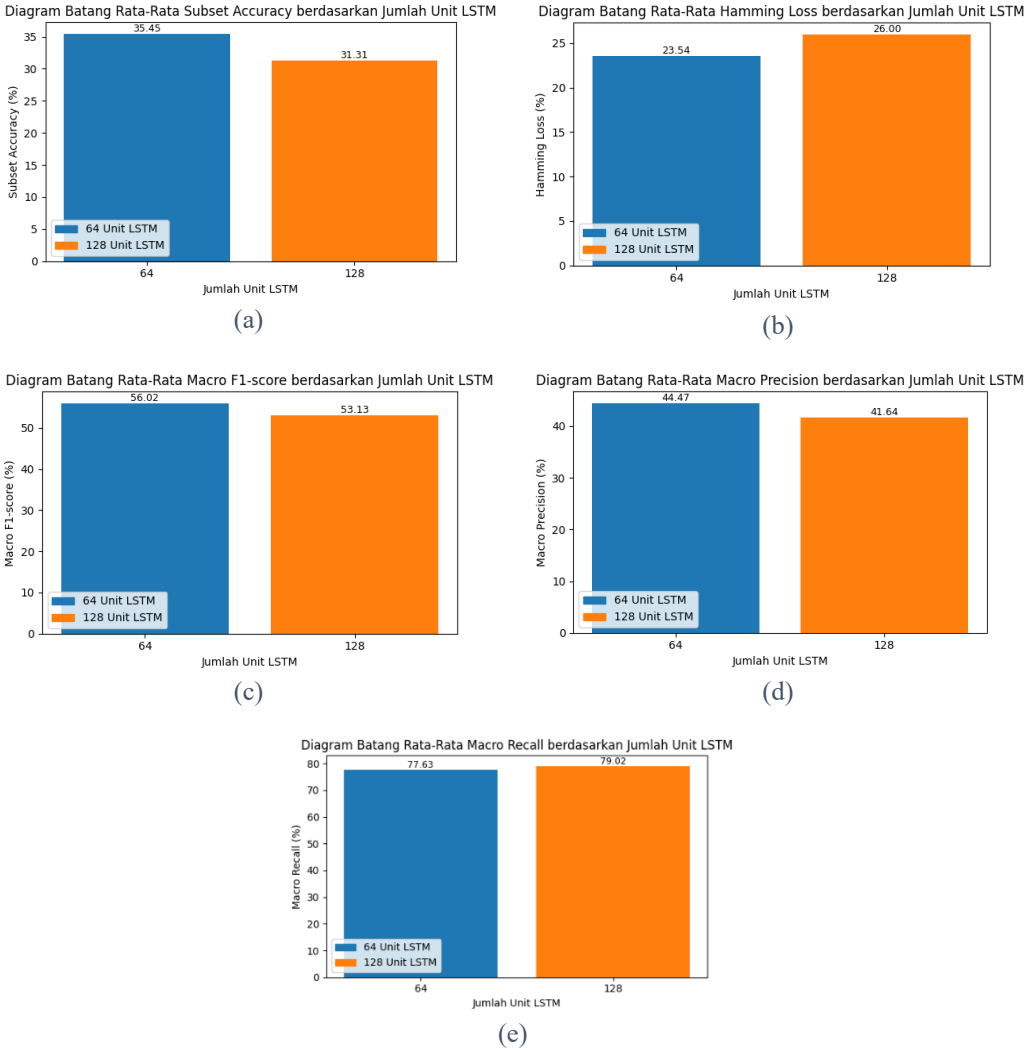
Jumlah unit pada LSTM menentukan kapasitas model dalam mempelajari hubungan berurutan dan menangkap konteks jangka panjang pada data teks. Semakin besar jumlah unit yang digunakan, semakin besar pula kapasitas representasi yang dimiliki model. Namun, peningkatan kapasitas ini juga berpotensi meningkatkan kompleksitas model dan risiko pembelajaran pola yang kurang relevan.

Pada penelitian ini digunakan dua variasi jumlah unit LSTM, yaitu 64 unit dan 128 unit, yang diterapkan secara merata pada seluruh konfigurasi skenario pengujian. Untuk menganalisis pengaruh jumlah unit LSTM terhadap performa model, seluruh hasil evaluasi dari skenario dengan 64 unit dan 128 unit dihitung nilai rata-ratanya untuk setiap metrik evaluasi. Hasil perbandingan tersebut disajikan pada Tabel 4.27.

Tabel 4.28 Rata-Rata Performa Berdasarkan Jumlah Unit LSTM

Jumlah Unit LSTM	***Subset Accuracy (%)***	***Hamming Loss (%)***	***Macro Precision (%)***	***Macro Recall (%)***	***Macro F1-Score (%)***
64	35,45	23,54	44,47	77,63	56,02
128	31,31	26,00	41,64	79,02	53,13

Sementara, visualisasinya ditampilkan pada Gambar 4.27.



Gambar 4.5 Diagram Batang (a) *'Subset Accuracy'*, (b) *'Hamming Loss'*, (c) *'Macro F1-score'*, (d) *'Macro Precision'*, dan (e) *'Macro Recall'* per Jumlah Unit LSTM

Hasil pada Tabel 4.27 serta seluruh visualisasi pada Gambar 4.4, menunjukkan bahwa jumlah unit LSTM berpengaruh terhadap performa model, meskipun perbedaannya tidak terlalu signifikan. Secara umum, konfigurasi dengan 64 unit LSTM menghasilkan performa yang lebih baik dibandingkan konfigurasi dengan 128 unit.

Hal ini pertama kali terlihat bahwa penggunaan 64 unit LSTM menghasilkan nilai *subset accuracy* yang lebih tinggi dibandingkan penggunaan 128 unit. Hal ini menunjukkan bahwa model dengan jumlah unit LSTM yang lebih moderat mampu menghasilkan prediksi kombinasi label yang lebih tepat secara keseluruhan. Peningkatan *subset accuracy* ini mengindikasikan bahwa kapasitas model sudah cukup untuk menangkap pola utama dalam data tanpa menimbulkan kompleksitas berlebih.

Selain itu, jumlah unit LSTM sebanyak 64 juga menghasilkan nilai *hamming loss* yang lebih rendah. Kondisi ini menunjukkan bahwa jumlah kesalahan prediksi per label lebih sedikit dibandingkan model dengan 128 unit LSTM. Dengan kata lain, peningkatan jumlah unit LSTM tidak selalu berbanding lurus dengan peningkatan ketepatan prediksi, terutama ketika kompleksitas model melebihi kebutuhan representasi data.

Dari sisi ketepatan prediksi, nilai *macro precision* pada konfigurasi 64 unit LSTM lebih tinggi dibandingkan konfigurasi 128 unit. Hal ini menunjukkan bahwa model dengan 64 unit LSTM cenderung lebih berhati-hati dalam mengaktifkan label positif, sehingga mampu menekan jumlah *false positive*. Kondisi ini penting dalam klasifikasi ujaran kebencian agar tidak terjadi pelabelan yang berlebihan pada teks yang sebenarnya tidak mengandung ujaran kebencian.

Sementara itu, konfigurasi 128 unit LSTM menghasilkan nilai *macro recall* yang sedikit lebih tinggi dibandingkan 64 unit LSTM. Hal ini menunjukkan bahwa peningkatan jumlah unit LSTM dapat meningkatkan sensitivitas model dalam mendeteksi label positif. Namun, peningkatan sensitivitas tersebut diiringi oleh

penurunan *precision* dan peningkatan *hamming loss*, sehingga secara keseluruhan keseimbangan performa menjadi kurang optimal.

Keseimbangan antara *precision* dan *recall* tercermin pada nilai *macro F1-score*. Berdasarkan Tabel 4.28, konfigurasi 64 unit LSTM menghasilkan nilai *macro F1-score* yang lebih tinggi dibandingkan konfigurasi 128 unit. Hal ini menunjukkan bahwa jumlah unit LSTM yang lebih kecil mampu memberikan performa yang lebih stabil dan seimbang dalam menangani klasifikasi *Multi-Label* dibandingkan jumlah unit LSTM yang lebih besar.

Secara keseluruhan, hasil analisis menunjukkan bahwa jumlah unit LSTM memiliki pengaruh yang signifikan terhadap performa klasifikasi *Multi-Label*. Penggunaan 64 unit LSTM memberikan keseimbangan terbaik antara ketepatan prediksi, sensitivitas deteksi, dan jumlah kesalahan per label. Sebaliknya, peningkatan jumlah unit LSTM menjadi 128 unit cenderung meningkatkan sensitivitas, namun menurunkan stabilitas dan ketepatan prediksi secara keseluruhan. Oleh karena itu, dalam konteks penelitian ini, penggunaan 64 unit LSTM dinilai lebih optimal dan konsisten, serta selaras dengan konfigurasi model terbaik yang telah ditentukan sebelumnya.

4.2.2.3 Pengaruh *Dimensi Embedding* (d)

Analisis pengaruh *hyperparameter* dimensi *embedding* dilakukan untuk mengetahui bagaimana ukuran representasi vektor kata memengaruhi performa klasifikasi *Multi-Label*. Dimensi *embedding* menentukan banyaknya informasi semantik yang dapat direpresentasikan oleh setiap kata dalam bentuk vektor numerik. Semakin besar dimensi *embedding*, semakin kaya informasi kontekstual

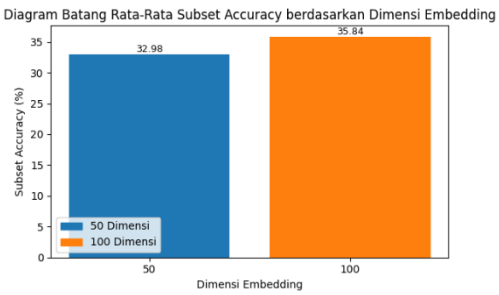
yang dapat ditangkap, namun hal tersebut juga berpotensi meningkatkan kompleksitas model dan risiko *overfitting*.

Pada penelitian ini, dimensi *embedding* diuji pada dua ukuran, yaitu 50 dimensi dan 100 dimensi. Kedua konfigurasi tersebut diterapkan secara konsisten pada berbagai kombinasi jumlah unit LSTM, *batch size*, dan nilai *threshold*, sehingga masing-masing dimensi *embedding* digunakan pada 12 skenario pengujian. Untuk memperoleh gambaran pengaruh dimensi *embedding* secara lebih objektif, seluruh hasil pengujian dikelompokkan berdasarkan ukuran *embedding*, kemudian dihitung nilai rata-rata dari lima metrik evaluasi, yaitu *subset accuracy*, *hamming loss*, *macro precision*, *macro recall*, dan *macro F1-score*. Hasil perhitungan tersebut disajikan pada Tabel 4.28.

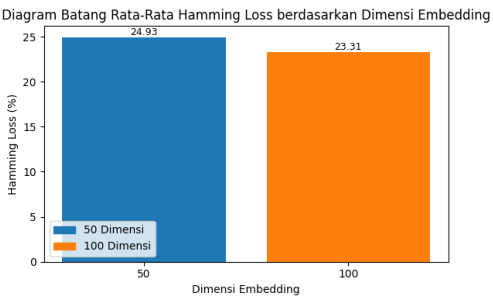
Tabel 4.29 Rata-Rata Performa Berdasarkan Dimensi *Embedding Word2Vec*

***Dimensi* Embedding**	***Subset Accuracy (%)***	***Hamming Loss (%)***	***Macro Precision (%)***	***Macro Recall (%)***	***Macro F1-Score (%)***
50 dimensi	32,98	24,93	42,15	79,62	54,48
100 dimensi	35,84	23,31	45,11	77,39	56,09

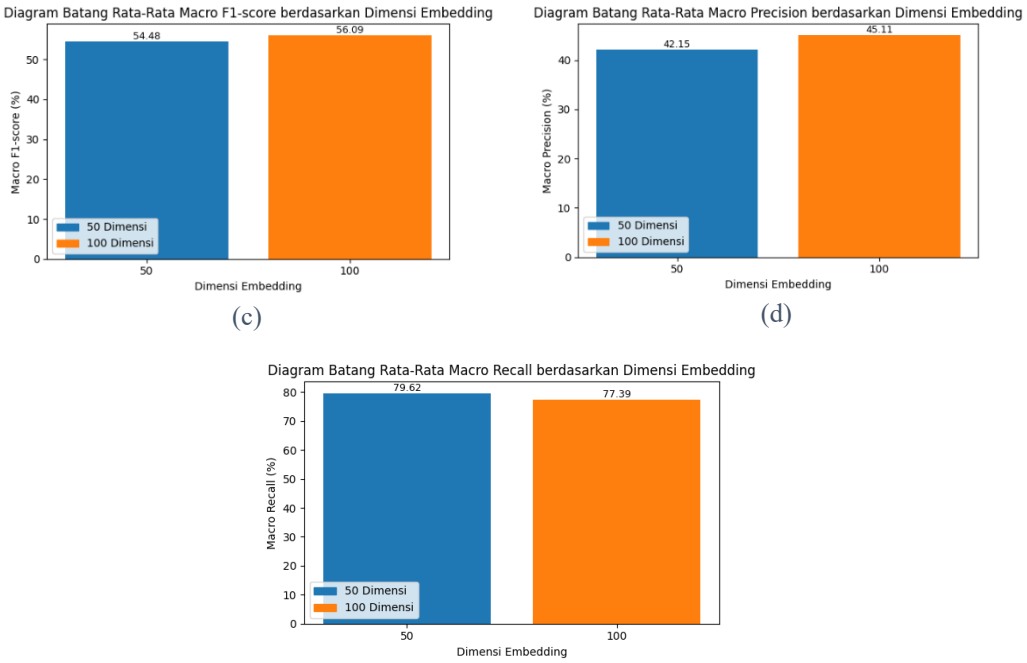
Sedangkan, visualisasi grafiknya disajikan pada Gambar 4.29.



(a)



(b)



Gambar 4.6 Diagram Batang (a) *'Subset Accuracy'*, (b) *'Hamming Loss'*, (c) *'Macro F1-score'*, (d) *'Macro Precision'*, dan (e) '*Macro Recall'* per *Dimensi Embedding*

Hasil pada Tabel 4.28 serta seluruh visualisasi pada Gambar 4.5, menunjukkan bahwa perbedaan dimensi *embedding* antara 50 dan 100 tidak memberikan perubahan performa yang signifikan pada model.

Embedding dengan dimensi 100 menghasilkan nilai *subset accuracy* yang lebih tinggi dibandingkan *embedding* 50 dimensi. Hal ini menunjukkan bahwa representasi kata dengan dimensi yang lebih besar mampu membantu sistem dalam memprediksi kombinasi label secara lebih tepat pada satu data. Peningkatan *subset accuracy* ini mengindikasikan bahwa informasi semantik tambahan yang terkandung dalam *embedding* 100 dimensi memberikan kontribusi positif terhadap kualitas prediksi.

Selain itu, *embedding* 100 dimensi juga menghasilkan nilai *hamming loss* yang lebih rendah. Kondisi ini menunjukkan bahwa jumlah kesalahan prediksi per

label lebih sedikit dibandingkan penggunaan *embedding* 50 dimensi. Dengan kata lain, peningkatan dimensi *embedding* tidak hanya meningkatkan ketepatan prediksi kombinasi label, tetapi juga mengurangi kesalahan klasifikasi pada tingkat label individual.

Dari sisi ketepatan prediksi, nilai *macro precision* pada *embedding* 100 dimensi lebih tinggi dibandingkan *embedding* 50 dimensi. Hal ini menunjukkan bahwa model dengan representasi kata yang lebih kaya cenderung lebih akurat dalam mengaktifkan label positif dan mampu menekan jumlah *false positive*. Kondisi ini penting dalam klasifikasi ujaran kebencian untuk menghindari pelabelan berlebihan terhadap teks yang sebenarnya tidak mengandung ujaran kebencian.

Sementara itu, *embedding* 50 dimensi menghasilkan nilai *macro recall* yang sedikit lebih tinggi dibandingkan *embedding* 100 dimensi. Hal ini menunjukkan bahwa dimensi *embedding* yang lebih kecil membuat model menjadi lebih sensitif dalam mendeteksi label positif. Namun, peningkatan sensitivitas tersebut diiringi oleh penurunan *precision* dan peningkatan *hamming loss*, sehingga secara keseluruhan keseimbangan performa menjadi kurang optimal.

Keseimbangan antara *precision* dan *recall* tercermin pada nilai *macro F1-score*. Berdasarkan Tabel 4.28, *embedding* 100 dimensi menghasilkan nilai *macro F1-score* yang lebih tinggi dibandingkan *embedding* 50 dimensi. Hal ini menunjukkan bahwa *embedding* dengan dimensi yang lebih besar mampu memberikan performa yang lebih stabil dan seimbang dalam menangani klasifikasi *Multi-Label*.

Secara keseluruhan, hasil analisis menunjukkan bahwa dimensi *embedding* memiliki pengaruh yang signifikan terhadap performa klasifikasi *Multi-Label*. Penggunaan *embedding* 50 dimensi cenderung meningkatkan sensitivitas deteksi, sedangkan *embedding* 100 dimensi memberikan prediksi yang lebih stabil dan akurat dengan jumlah kesalahan per label yang lebih rendah. Dalam konteks penelitian ini, *embedding* 100 dimensi dinilai lebih optimal karena mampu meningkatkan *subset accuracy*, menurunkan *hamming loss*, serta menghasilkan nilai *macro F1-score* yang lebih tinggi secara rata-rata. Temuan ini juga selaras dengan hasil pemilihan model terbaik, di mana konfigurasi dengan *embedding* 100 dimensi menunjukkan performa yang lebih unggul dibandingkan *embedding* 50 dimensi.

4.2.2.4 Pengaruh *Batch Size* (B)

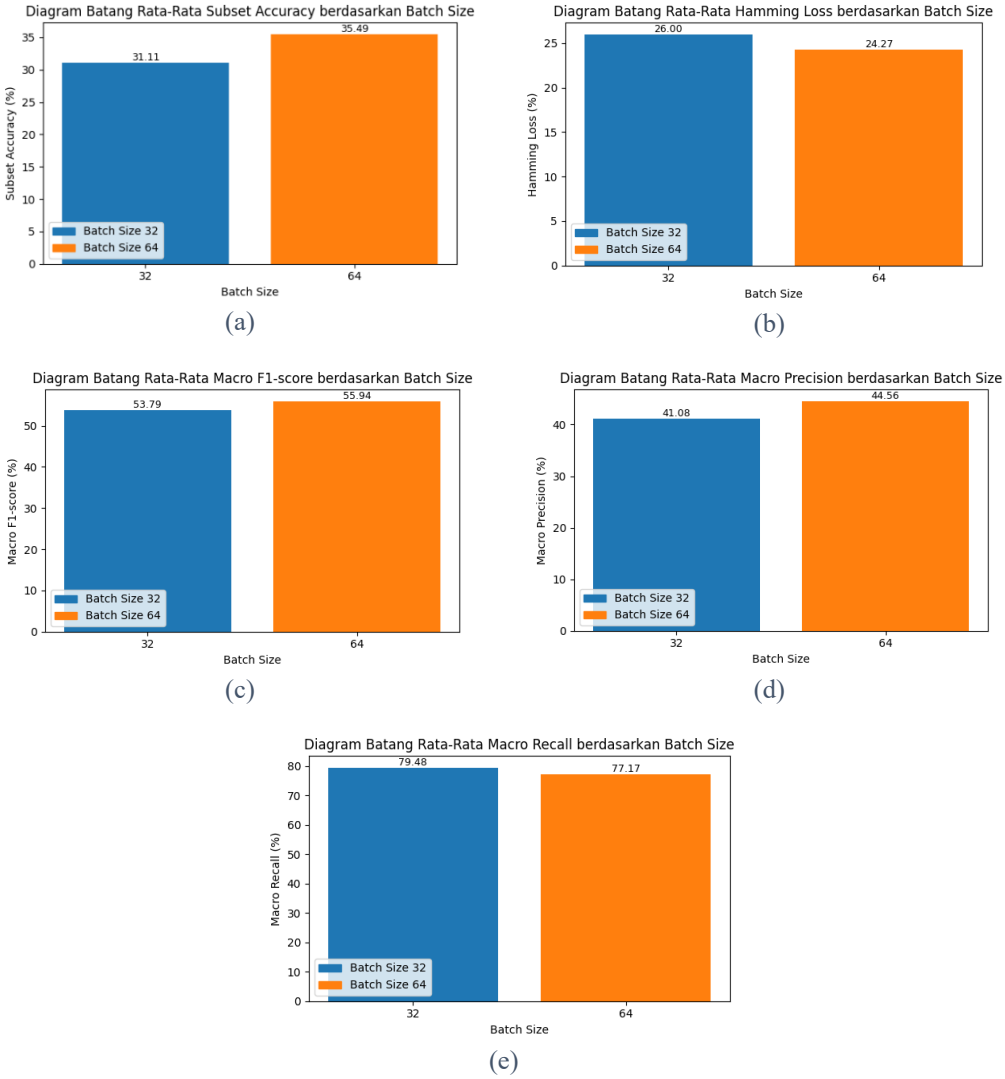
Pada penelitian ini, *batch size* diuji dalam dua nilai, yaitu 32 dan 64. Kedua nilai ini memengaruhi stabilitas proses pembelajaran model LSTM serta kemampuan model dalam generalisasi terhadap data uji.

Untuk mengetahui pengaruh batch size terhadap performa klasifikasi multilabel, seluruh hasil evaluasi dari skenario yang menggunakan batch size 32 dan 64 dihitung nilai rata-ratanya pada setiap metrik evaluasi. Rekapitulasi hasil tersebut disajikan pada Tabel 4.29 berikut.

Tabel 4.30 Rata-Rata Performa Model berdasarkan *Batch Size*

Batch Size	*Subset Accuracy (%)*	*Hamming Loss (%)*	*Macro Precision (%)*	*Macro Recall (%)*	*Macro F1-Score (%)*
32	31,11	26,00	41,08	79,48	53,79
64	35,49	24,27	44,56	77,17	55,94

Sementara, visualisasinya disajikan pada Gambar 4.6.



Gambar 4.7 Diagram Batang (a) *'Subset Accuracy'*, (b) *'Hamming Loss'*, (c) *'Macro F1-score'*, (d) *'Macro Precision'*, dan (e) *'Macro Recall'* per *Batch Size*

Hasil pada Tabel 4.29 serta seluruh visualisasi pada Gambar 4.6, dapat disimpulkan bahwa *batch size* memberikan pengaruh yang cukup signifikan terhadap performa model.

Penggunaan *batch size* 64 menghasilkan nilai *subset accuracy* yang lebih tinggi dibandingkan *batch size* 32. Peningkatan nilai ini menunjukkan bahwa

pemrosesan data dalam kelompok yang lebih besar membantu model dalam menghasilkan prediksi kombinasi label yang lebih tepat secara keseluruhan.

Selain itu, *batch size* 64 juga menghasilkan nilai *hamming loss* yang lebih rendah. Hal ini mengindikasikan bahwa jumlah kesalahan prediksi per label lebih sedikit dibandingkan *batch size* 32. Dengan kata lain, penggunaan *batch size* yang lebih besar mampu meningkatkan ketepatan prediksi tanpa meningkatkan kesalahan secara signifikan.

Dari sisi ketepatan prediksi, *macro precision* pada *batch size* 64 menunjukkan nilai yang lebih tinggi dibandingkan *batch size* 32. Peningkatan *precision* ini menandakan bahwa model dengan *batch size* lebih besar cenderung tidak memberikan prediksi positif secara berlebihan, sehingga mampu menekan jumlah *false positive*. Kondisi ini sangat penting dalam konteks klasifikasi ujaran kebencian untuk menghindari pelabelan yang tidak tepat pada teks yang sebenarnya netral.

Sementara itu, *macro recall* pada *batch size* 32 sedikit lebih tinggi dibandingkan *batch size* 64. Hal ini menunjukkan bahwa *batch size* yang lebih kecil membuat model menjadi lebih sensitif dalam mendeteksi label positif. Namun, peningkatan sensitivitas tersebut diiringi oleh penurunan *precision* dan peningkatan *hamming loss*, sehingga secara keseluruhan keseimbangan performa menjadi kurang optimal.

Keseimbangan antara *precision* dan *recall* tercermin pada nilai *macro F1-score*. Berdasarkan Tabel 4.27, *batch size* 64 menghasilkan nilai *macro F1-score* yang lebih tinggi, yaitu sebesar 55,94%, dibandingkan *batch size* 32. Nilai ini

menunjukkan bahwa penggunaan *batch size* 64 mampu memberikan performa yang lebih stabil dan seimbang dalam menangani klasifikasi *Multi-Label*.

Secara keseluruhan, hasil analisis menunjukkan bahwa *batch size* memiliki pengaruh yang cukup signifikan terhadap performa model. *Batch size* 32 cenderung meningkatkan sensitivitas deteksi, sedangkan *batch size* 64 memberikan prediksi yang lebih stabil dan akurat dengan jumlah kesalahan per label yang lebih rendah. Dalam konteks penelitian ini, *batch size* 64 dinilai lebih optimal karena mampu meningkatkan *subset accuracy*, menurunkan *hamming loss*, serta menghasilkan nilai *macro F1-score* yang lebih tinggi secara rata-rata. Temuan ini juga konsisten dengan hasil pemilihan model terbaik, di mana skenario dengan *batch size* 64 menunjukkan performa yang lebih unggul dibandingkan *batch size* 32.

4.3 Integrasi Sains dan Islam Nilai

Penelitian ini tidak hanya berfokus pada pengembangan model LSTM untuk mengklasifikasikan ujaran kebencian pada *tweet* berbahasa Indonesia, tetapi juga berupaya mengintegrasikan nilai-nilai keislaman dalam keseluruhan proses ilmiah. Islam memberikan fondasi moral melalui tiga bentuk hubungan utama: hubungan manusia dengan Allah Swt (*Muamalah Ma'a Allah*), hubungan manusia dengan sesama (*Muamalah Ma'a An-Nas*), serta hubungan manusia dengan alam dan lingkungan (*Muamalah Ma'al Bi'ah*). Ketiga landasan ini memberikan arah dan makna dalam penelitian, terutama karena objek penelitian ini berkaitan dengan perilaku berbahasa yang berdampak langsung pada kualitas hubungan sosial di masyarakat.

4.3.1 *Muamalah Ma'a Allah*

Muamalah Ma'a Allah berkaitan dengan hubungan manusia dengan Allah Swt sebagai dasar dari seluruh aktivitas ilmiah. Seluruh rangkaian penelitian ini, mulai dari pengumpulan data, pemrosesan teks, perancangan arsitektur LSTM, hingga evaluasi performa model, diposisikan sebagai bentuk ibadah melalui proses menuntut ilmu dan menghasilkan karya yang bermanfaat bagi masyarakat. Upaya menghadirkan sistem klasifikasi ujaran kebencian bertujuan untuk menjaga kemaslahatan sosial dan mengurangi perilaku verbal yang tidak sesuai dengan nilai-nilai Islam.

Allah Swt berfirman dalam Q.S. Al-Mujādalah ayat 11:

يٰٓاَيُّهَا الَّذِيْنَ اٰمَنُوْٓا اِذَا قِيْلَ لَكُمْ تَفَسَّحُوْا فِى الْمَجٰلِسِ فَافْسَحُوْا يَفْسَحِ اللّٰهُ لَكُمْۚ وَاِذَا قِيْلَ انْشُزُوْا فَانْشُزُوْا يَرْفَعِ اللّٰهُ الَّذِيْنَ اٰمَنُوْا مِنْكُمْۙ وَالَّذِيْنَ اُوْتُوا الْعِلْمَ دَرَجٰتٍۗ وَاللّٰهُ بِمَا تَعْمَلُوْنَ خَبِيْرٌ ۝١١

"Wahai orang-orang yang beriman! Apabila dikatakan kepadamu, 'Berluas-lapanglah dalam majelis-majelis,' maka lapangkanlah, niscaya Allah akan memberi kelapangan untukmu. Dan apabila dikatakan, 'Berdirilah kamu,' maka berdirilah; niscaya Allah akan meninggikan orang-orang yang beriman di antaramu dan orang-orang yang diberi ilmu beberapa derajat. Dan Allah Mahateliti terhadap apa yang kamu kerjakan." (QS. Al- Mujādalah: 11).

Ayat tersebut mengandung pesan penting tentang adab seorang mukmin dalam menaati perintah Allah. Dalam tafsir as-Sa'di dijelaskan bahwa ayat ini mengajarkan adab bermajelis, di mana seorang hamba diperintahkan untuk melapangkan tempat bagi saudaranya apabila diminta, dan untuk berdiri apabila keadaan menuntutnya. Sikap taat terhadap perintah ini mencerminkan keimanan dan kerendahan hati seorang mukmin. Balasan Allah sesuai jenis amalnya: barang

siapa memberi kelapangan, maka Allah akan memberinya kelapangan di dunia dan akhirat.

Tafsir as-Sa'di juga menegaskan bahwa Allah Swt akan meninggikan derajat orang-orang yang beriman dan berilmu sesuai kadar iman dan ilmu mereka. Ilmu yang benar akan mendorong pemiliknya untuk beradab, patuh, dan menjalankan perintah Allah dengan penuh keikhlasan. Derajat yang Allah tinggikan mencakup kemuliaan di dunia dan pahala yang agung di akhirat. Dengan demikian, ayat ini menempatkan ilmu sebagai anugerah besar yang dimuliakan Allah, sekaligus menunjukkan bahwa buah dari ilmu adalah akhlak dan ketaatan (As-Sa'di, n.d.).

Dalam konteks penelitian ini, ayat tersebut menjadi landasan bahwa ilmu dan kemampuan berpikir merupakan amanah yang harus dijaga. Pemanfaatan teknologi seperti NLP dan model LSTM untuk menanggulangi masalah sosial seperti penyebaran ujaran kebencian merupakan bentuk pengamalan ilmu yang bernilai ibadah. Penelitian ini diarahkan untuk menghasilkan sistem yang dapat membantu mencegah perilaku verbal yang bertentangan dengan syariat, seperti menghina, mencela, atau merendahkan martabat manusia.

Prinsip amanah ilmu juga diwujudkan melalui kehati-hatian peneliti dalam setiap tahap penelitian. Pemilihan dataset, pengolahan data, penentuan parameter, dan evaluasi model dilakukan dengan penuh tanggung jawab agar tidak menimbulkan kesimpulan yang keliru dan tidak menimbulkan mudarat. Seluruh proses ini dijalankan dengan niat ibadah, rasa syukur atas kemampuan yang

diberikan, serta kesadaran bahwa setiap amal akan dimintai pertanggungjawaban oleh Allah Swt.

Dengan demikian, *Muamalah Ma'a Allah* tercermin dalam niat ibadah, syukur atas anugerah ilmu, dan amanah dalam penggunaan teknologi. Sistem klasifikasi ujaran kebencian yang dibangun tidak hanya berfungsi secara teknis, tetapi juga merupakan wujud *'ilmun nāfi'*, yaitu ilmu yang membawa manfaat, menjaga kemaslahatan, serta menjauhkan manusia dari perilaku verbal yang merusak dan tidak diridai Allah Swt.

4.3.2 *Muamalah Ma'a An-Nas*

Muamalah Ma'a An-Nas membahas hubungan manusia dengan sesama serta bagaimana seorang mukmin menjaga etika dalam komunikasi. Karena fokus penelitian ini adalah mengklasifikasikan berbagai bentuk ujaran kebencian, nilai Muamalah Ma'a An-Nas memiliki hubungan yang sangat erat dengan objek penelitian. Ujaran kebencian merupakan bentuk komunikasi yang bertentangan dengan syariat karena dapat menimbulkan permusuhan, memperbesar konflik, dan merusak keharmonisan sosial.

Allah Swt berfirman dalam Q.S. Al-Isrā' ayat 53:

وَقُلْ لِّعِبَادِيْ يَقُوْلُوا الَّتِيْ هِيَ اَحْسَنُۗ اِنَّ الشَّيْطٰنَ يَنْزَغُ بَيْنَهُمْۗ اِنَّ الشَّيْطٰنَ كَانَ لِلْاِنْسَانِ عَدُوًّا مُّبِيْنًا ﴿٥٣﴾

"Dan katakanlah kepada hamba-hamba-Ku yang beriman apabila mereka berkata kepada kaum musyrikin, "Hendaklah mereka mengucapkan perkataan yang lebih baik dan benar walaupun mereka bersikap keras dan berkata kasar kepadamu. Sungguh, setan itu senantiasa mencari peluang dan berusaha menimbulkan perselisihan di antara mereka, yakni orang-orang yang beriman. Sungguh, setan itu adalah musuh yang nyata bagi manusia." (QS. Al- Isrā': 53)

Ayat ini berisi perintah agar setiap hamba Allah mengucapkan perkataan yang terbaik dalam percakapan maupun interaksi. Dalam Tafsir Al-Muyassar dijelaskan bahwa Allah memerintahkan orang beriman untuk bertutur kata dengan baik dalam setiap bentuk komunikasi, karena ucapan yang buruk dapat memicu permusuhan, kerusakan, dan pertengkaran. Setan selalu berusaha menyalakan api konflik melalui kata-kata yang tidak dijaga, sehingga manusia dituntut untuk berhati-hati dalam berbicara dan memilih ucapan yang membawa kedamaian (Al-Muyassar, n.d.).

Pesan utama ayat ini menegaskan bahwa kualitas ucapan memiliki pengaruh besar terhadap hubungan antarmanusia. Komunikasi yang baik akan memperkuat harmoni, sedangkan kata-kata yang kasar atau penuh kebencian dapat memicu perpecahan. Nilai ini sejalan dengan tujuan penelitian ini, yaitu menghadirkan teknologi yang dapat membantu mengidentifikasi bentuk ucapan yang berpotensi menimbulkan kerusakan sosial.

Dalam konteks penelitian, model LSTM digunakan untuk mengenali berbagai kategori ujaran kebencian seperti penghinaan individu (*HS_Individual*), serangan terhadap kelompok (*HS_Group*), ujaran bernuansa ras, agama, fisik, maupun gender. Sistem klasifikasi ini dapat menjadi alat bantu bagi moderator platform digital, lembaga sosial, dan masyarakat dalam mencegah penyebaran ujaran yang melanggar adab komunikasi sesuai tuntunan syariat.

Model ini juga berfungsi sebagai sarana edukasi, karena membantu masyarakat memahami batas antara kritik yang dibolehkan dan ujaran kebencian yang dilarang. Dengan demikian, penelitian ini tidak hanya bersifat teknis, tetapi

juga membawa kontribusi nyata dalam menjaga hubungan antarmanusia di ruang digital agar tetap harmonis, santun, dan jauh dari provokasi negatif.

Melalui penerapan nilai *Muamalah Ma'a An-Nas*, penelitian ini menempatkan teknologi sebagai instrumen untuk menjaga kehormatan sesama, mengurangi potensi konflik, dan memperkuat etika komunikasi yang sejalan dengan ajaran Islam.

4.3.3 *Muamalah Ma'al Bi'ah*

Muamalah Ma'al Bi'ah berkaitan dengan hubungan manusia dengan lingkungan, baik lingkungan fisik maupun lingkungan sosial. Dalam konteks modern, lingkungan sosial tidak hanya mencakup interaksi langsung, tetapi juga ruang digital tempat masyarakat berkomunikasi melalui media sosial. Perilaku manusia di ruang digital dapat memberikan dampak besar terhadap kondisi sosial, termasuk munculnya kerusakan moral akibat penyebaran ujaran kebencian. Oleh karena itu, pengelolaan lingkungan termasuk lingkungan komunikasi digital merupakan bagian dari amanah yang harus dijaga oleh manusia sebagai khalifah di bumi.

Allah Swt menegaskan dalam Q.S. Hūd ayat 61:

وَاِلٰى ثَمُوْدَ اَخَاهُمْ صٰلِحًاۗ قَالَ يٰقَوْمِ اعْبُدُوا اللّٰهَ مَا لَكُمْ مِّنْ اِلٰهٍ غَيْرُهٗۗ هُوَ اَنْشَاَكُمْ مِّنَ الْاَرْضِ وَاسْتَعْمَرَكُمْ فِيْهَا ۞ فَاسْتَغْفِرُوْهُ ثُمَّ تُوْبُوْٓا اِلَيْهِۗ اِنَّ رَبِّيْ قَرِيْبٌ مُّجِيْبٌ ﴿٦١﴾

"Kepada (kaum) Samud (Kami utus) saudara mereka, Saleh. Dia berkata, "Wahai kaumku, sembahlah Allah! Sekali-kali tidak ada tuhan bagimu selain Dia. Dia telah menciptakanmu dari bumi (tanah) dan menjadikanmu pemakmurnya. Oleh karena itu, mohonlah ampunan kepada-Nya, kemudian bertobatlah kepada-Nya. Sesungguhnya Tuhanku sangat dekat lagi Maha Memperkenankan (doa hamba-Nya)." (QS. Hūd: 61)

Dalam Tafsir Al-Madinah Al-Munawwarah dijelaskan bahwa Allah Swt mengutus Nabi Shalih kepada kaum Tsamud untuk menyeru mereka agar hanya menyembah Allah semata, karena hanya Dia yang berhak disembah. Salah satu bukti keesaan dan kesempurnaan ketuhanan Allah adalah penciptaan manusia dari tanah serta pemberian kemampuan kepada mereka untuk memakmurkan bumi. Pemakmuran tersebut diwujudkan melalui berbagai aktivitas kehidupan, seperti mendirikan bangunan, memanfaatkan sumber daya alam, serta mengelola bumi sebagai tempat tinggal dan sumber penghidupan. Oleh karena itu, manusia tidak hanya diperintahkan untuk memanfaatkan bumi, tetapi juga untuk menjaganya agar tetap berfungsi secara berkelanjutan (Markaz Ta'dzhim al-Qur'an, n.d.). Ayat ini juga disertai perintah untuk memohon ampunan dan senantiasa berada di jalan tobat sebagai bentuk kesadaran moral atas tanggung jawab manusia terhadap amanah yang telah diberikan. Allah Swt menegaskan bahwa Dia Maha Dekat dengan hamba-hamba-Nya yang beriman dan senantiasa mengabulkan doa mereka yang kembali kepada-Nya dengan ketaatan dan keikhlasan.

Makna "menjadikan kamu pemakmurnya" dalam ayat tersebut tidak hanya berkaitan dengan pembangunan fisik, tetapi juga mencakup upaya menjaga tatanan kehidupan sosial agar tetap harmonis dan bermanfaat. Dalam kerangka ini, lingkungan sosial termasuk ruang digital merupakan bagian dari "bumi" yang harus dijaga agar tidak rusak akibat perilaku manusia. Penyebaran ujaran kebencian di media sosial berpotensi memicu konflik, memecah persatuan, serta merusak hubungan antarkelompok, sehingga dapat dikategorikan sebagai bentuk kerusakan sosial yang bertentangan dengan amanah pemakmuran bumi.

Penelitian ini berkontribusi dalam pelaksanaan Muamalah *Ma'al Bi'ah* melalui pengembangan aplikasi klasifikasi *Multi-Label* ujaran kebencian berbasis LSTM. Sistem yang dikembangkan bertujuan untuk mengidentifikasi dan memetakan konten bermuatan negatif sehingga pemangku kepentingan, seperti platform media sosial, lembaga pemerintah, dan organisasi sosial, dapat mengambil langkah preventif terhadap penyebaran ujaran yang merusak. Dengan menurunkan intensitas konten kebencian, sistem ini diharapkan mampu menjaga "kebersihan" ruang digital serta mendukung terciptanya lingkungan komunikasi yang lebih aman, santun, dan beradab.

Selain itu, keberadaan sistem ini turut mendorong peningkatan literasi digital masyarakat. Setiap individu diingatkan bahwa kata-kata yang disebarkan di ruang digital memiliki konsekuensi moral dan sosial. Dengan demikian, teknologi yang digunakan dalam penelitian ini tidak hanya berfungsi sebagai model klasifikasi berbasis LSTM, tetapi juga sebagai sarana untuk mengurangi kerusakan sosial, menjaga keharmonisan, dan menciptakan ruang interaksi yang lebih sehat. Penerapan Muamalah *Ma'al Bi'ah* dalam penelitian ini tercermin melalui pemanfaatan teknologi sebagai wujud pelaksanaan amanah manusia dalam menjaga keseimbangan dan kemaslahatan lingkungan sosial.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa model *Long Short-Term Memory* (LSTM) dapat digunakan untuk melakukan klasifikasi *multi-label* ujaran kebencian pada *tweet* berbahasa Indonesia. Penelitian ini menguji sebanyak 24 skenario dengan memvariasikan empat *hyperparameter* utama, yaitu *dimensi embedding* (50 dan 100), jumlah unit LSTM (64 dan 128), *batch size* (32 dan 64), serta nilai *threshold* (0,1, 0,2, dan 0,3). Variasi *hyperparameter* tersebut digunakan untuk melihat pengaruh setiap konfigurasi terhadap performa model pada tugas klasifikasi multi-label.

Perbedaan nilai pada setiap *hyperparameter* tersebut menghasilkan variasi performa model pada seluruh metrik evaluasi yang digunakan. Berdasarkan hasil evaluasi terhadap lima metrik utama, yaitu *subset accuracy, hamming loss, macro precision, macro recall,* dan *macro F1-score,* Skenario 18 dengan konfigurasi *embedding* 100 dimensi, 64 unit LSTM, *batch size* 64, dan *threshold* 0,3 (100d–64H–64B–0.3t) menunjukkan performa yang lebih baik dibandingkan skenario lainnya. Konfigurasi ini menghasilkan *subset accuracy* sebesar 52,16%, *hamming loss* sebesar 17,15%, serta *macro F1-score* sebesar 61,44%, yang menunjukkan keseimbangan antara ketepatan prediksi dan kemampuan model dalam mengenali kombinasi label secara bersamaan.

Untuk memastikan bahwa hasil tersebut tidak dipengaruhi oleh satu pembagian data tertentu, konfigurasi dengan performa terbaik diuji menggunakan metode K-Fold Cross Validation. Hasil pengujian menunjukkan bahwa performa model relatif stabil pada setiap fold, sehingga model memiliki kemampuan generalisasi yang konsisten terhadap variasi data uji.

Analisis pengaruh *hyperparameter* menunjukkan bahwa nilai *threshold* memiliki peran penting dalam mengatur keseimbangan performa model. *Threshold* yang rendah cenderung meningkatkan *recall* namun menurunkan *precision*, sedangkan peningkatan nilai *threshold* membuat model lebih selektif, meningkatkan *subset accuracy* dan *precision*, serta menurunkan *hamming loss*. Selain itu, penggunaan *batch size* 64 menghasilkan performa yang lebih stabil dibandingkan *batch size* 32, sementara penggunaan 64 unit LSTM menunjukkan hasil yang lebih seimbang dibandingkan 128 unit. Dari sisi representasi kata, *embedding* 100 dimensi memberikan hasil yang lebih konsisten dibandingkan 50 dimensi.

Berdasarkan keseluruhan hasil tersebut, dapat disimpulkan bahwa pemilihan kombinasi *hyperparameter* yang tepat berperan penting dalam menghasilkan performa model LSTM yang stabil dan seimbang untuk klasifikasi *multi-label* ujaran kebencian pada tweet berbahasa Indonesia.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, berikut adalah beberapa saran yang dapat dipertimbangkan untuk pengembangan lebih lanjut dari model LSTM untuk klasifikasi *Multi-Label* ujaran kebencian ini:

1. Eksplorasi Arsitektur Model

Meskipun LSTM telah diuji, arsitektur lain mungkin lebih cocok. Disarankan untuk mencoba model *Bidirectional* LSTM (Bi-LSTM) yang dapat membaca konteks dari dua arah, atau arsitektur *Transformer* (seperti IndoBERT) yang terbukti unggul dalam memahami konteks bahasa Indonesia yang kompleks.

2. Eksplorasi Penggunaan Ektrasi Fitur Lain

Meskipun penggunaan *Word2Vec* dengan model *Skip-Gram* telah memberikan hasil yang memadai, penggunaan teknik representasi kata lain seperti *FastText* atau *GloVe* bisa memberikan hasil yang lebih baik, terutama dalam menangani kata-kata yang jarang muncul atau kata-kata yang memiliki banyak bentuk (misalnya kata kerja dengan berbagai bentuk infleksi). Dengan mengganti teknik representasi kata ini, model dapat menangkap lebih banyak makna dari kata-kata yang tidak ditemukan dalam kamus standar.

DAFTAR PUSTAKA

Afif, M. F. A., Nurhamidah, Y., & Mashuri, M. F. (2021). Kematangan emosi dalam perilaku ujaran kebencian pada kebijakan politik. *Cognicia*, *9*(1), 25–30. https://doi.org/10.22219/cognicia.v9i1.14234

Akbar, I., Faisal, M., & Chamidy, T. (2024). Multi-Label Classification of Indonesian Qur'an Translation using Long Short-Term Memory Model. *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, 119–128. https://doi.org/10.22219/kinetik.v9i2.1901

Akbar, J., Hairul Fahmi, & Wafiah Murniati. (2025). Multi Label Klasifikasi Genre Film Berdasarkan Sinopsis Menggunakan Metode Long Short-Term Memory (lstm). *Jurnal Manajemen Informatika dan Sistem Informasi*, *8*(1), 110–119. https://doi.org/10.36595/misi.v8i1.1436

Alfaro, R., Allende-Cid, H., & Allende, H. (2023). Multilabel Text Classification with Label-Dependent Representation. *Applied Sciences*, *13*(6), 3594. https://doi.org/10.3390/app13063594

Almeida, F., & Xexéo, G. (2023). *Word Embeddings: A Survey* (No. arXiv:1901.09069). arXiv. https://doi.org/10.48550/arXiv.1901.09069

Al-Muyassar. (n.d.). *Surat Al-Isra Ayat 53 Arab, Latin, Terjemah dan Tafsir | Baca di TafsirWeb*. Retrieved December 22, 2025, from https://tafsirweb.com/4657-surat-al-isra-ayat-53.html

Al-Smadi, B. S. (2024). DeBERTa-BiLSTM: A multi-label classification model of Arabic medical questions using pre-trained models and deep learning. *Computers in Biology and Medicine*, *170*, 107921. https://doi.org/10.1016/j.compbiomed.2024.107921

Ambari, N., Puspitasari, N., & Septiarini, A. (2025). Prediction of Budget Planning Using the Long Short Term Memory. *Journal of Artificial Intelligence and Software Engineering (J-AISE)*, *5*(1), 116. https://doi.org/10.30811/jaise.v5i1.6428

Andriani, A. D., Fitri, S. A., & Muchtar, K. (2024). Model Komunikasi Literasi Digital Dalam Mengatasi Ujaran Kebencian Di Media Sosial. *Interaksi: Jurnal Ilmu Komunikasi*, *13*(2), 439–464. https://doi.org/10.14710/interaksi.13.2.439-464

Angger Saputra, R., & Sibaroni, Y. (2025). Multilabel Hate Speech Classification in Indonesian Political Discourse on X using Combined Deep Learning Models with Considering Sentence Length. *Jurnal Ilmu Komputer Dan Informasi*, *18*(1), 113–125. https://doi.org/10.21609/jiki.v18i1.1440

Ash-Shidiq, M. A., & Pratama, A. R. (2021). Ujaran Kebencian Di Kalangan Pengguna Media Sosial Di Indonesia: Agama Dan Pandangan Politik. *AUTOMATA*, *2*(1). https://journal.uii.ac.id/AUTOMATA/article/view/17286

As-Sa'di, A. bin N. (n.d.). *Surat Al-Mujadalah Ayat 11 Arab, Latin, Terjemah dan Tafsir | Baca di TafsirWeb*. Retrieved December 22, 2025, from https://tafsirweb.com/10765-surat-al-mujadalah-ayat-11.html

Ayo, F. E., Folorunso, O., Ibharalu, F. T., & Osinuga, I. A. (2020). Machine learning techniques for hate speech classification of twitter data: State-of-the-art, future challenges and research directions. *Computer Science Review*, *38*, 100311. https://doi.org/10.1016/j.cosrev.2020.100311

Badry Ali Mustofa & Wawan Laksito Yuly Saptomo. (2025). Use of Natural Language Processing in Social Media Text Analysis. *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, *4*(2), 1235–1238. https://doi.org/10.59934/jaiea.v4i2.875

Bisht, A., Singh, A., Bhadauria, H. S., Virmani, J., & Kriti. (2020). Detection of Hate Speech and Offensive Language in Twitter Data Using LSTM Model. In S. Jain & S. Paul (Eds.), *Recent Trends in Image and Signal Processing in Computer Vision* (Vol. 1124, pp. 243–264). Springer Singapore. https://doi.org/10.1007/978-981-15-2740-1_17

Cahyani, J., Mujahidin, S., & Fiqar, T. P. (2023). Implementasi Metode Long Short Term Memory (LSTM) untuk Memprediksi Harga Bahan Pokok Nasional. *JUSTIN (Jurnal Sistem dan Teknologi Informasi)*, *11*(2), 346–357. https://doi.org/10.26418/justin.v11i2.57395

Cunha, L. (2022). Deep learning with Python (2ª ed)—François Chollet—Manning, outubro 2021, 504 pp. *Interações: Sociedade e as novas modernidades*, *42*, 113–115. https://doi.org/10.31211/interacoes.n42.2022.r1

Dharmawan, S., Mawardi, V. C., & Perdana, N. J. (2023). Klasifikasi Ujaran Kebencian Menggunakan Metode FeedForward Neural Network (IndoBERT). *Jurnal Ilmu Komputer dan Sistem Informasi*, *11*(1). https://doi.org/10.24912/jiksi.v11i1.24066

Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark* (No. arXiv:2109.14545). arXiv. https://doi.org/10.48550/arXiv.2109.14545

Ezzat, I., Abdulqader, A. W., & Shaker, A. S. (2023). Effectual Text Classification in Data Mining: A Practical Approach. *Mesopotamian Journal of Big Data*, *2023*, 46–52. https://doi.org/10.58496/MJBD/2023/007

Handayani, T. P. & Hilmansyah Gani. (2023). Enhancing Multi-Label Hate Speech and Abusive Language Detection on Indonesian Twitter Using Recurrent Neural Networks with Hyperparameter Tuning. *Jurnal Ilmiah Teknik*

Mesin, Elektro Dan Komputer, *3*(3), 601–612. https://doi.org/10.51903/juritek.v3i3.3022

Huan, J. L., Sekh, A. A., Quek, C., & Prasad, D. K. (2022). Emotionally charged text classification with deep learning and sentiment semantic. *Neural Computing and Applications*, *34*(3), 2341–2351. https://doi.org/10.1007/s00521-021-06542-1

Isnain, A. R., Sihabuddin, A., & Suyanto, Y. (2020). Bidirectional Long Short Term Memory Method and Word2vec Extraction Approach for Hate Speech Detection. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *14*(2), 169–178. https://doi.org/10.22146/ijccs.51743

Jamilah, F., & Wahyuni, P. (2020). Ujaran Kebencian dalam Kolom Komentar YouTube pada Tahun Politik Pemilihan Presiden 2019. *Silampari Bisa: Jurnal Penelitian Pendidikan Bahasa Indonesia, Daerah, dan Asing*, *3*(2), 325–341. https://doi.org/10.31540/silamparibisa.v3i2.1109

Javid, A. M., Das, S., Skoglund, M., & Chatterjee, S. (2021). A ReLU Dense Layer to Improve the Performance of Neural Networks. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2810–2814. https://doi.org/10.1109/ICASSP39728.2021.9414269

Kartika, S., & Nurhayati, N. (2023). Ujaran Kebencian (Hate Speech) di Media Sosial dalam Konteks Hukum dan Perubahan Sosial (Studi Kasus pada Masyarakat Kota Medan). *JURNAL MERCATORIA*, *16*(1), 99–106. https://doi.org/10.31289/mercatoria.v16i1.7668

Kemp, S. (2023). *Digital 2023 Global Overview Report*. https://datareportal.com/reports/digital-2023-global-overview-report

Marietta, B., Santoso, R., & Sa'adah, A. A. (2025). Peramalan Nilai Tukar Rupiah Terhadap Dollar Amerika Dengan Metode Long Short-Term Memory Dan Gated Recurrent Unit. *Jurnal Gaussian*, *14*(1), 13–22. https://doi.org/10.14710/j.gauss.14.1.13-22

Markaz Ta'dzhim al-Qur'an. (n.d.). *Surat Hud Ayat 61 Arab, Latin, Terjemah dan Tafsir | Baca di TafsirWeb*. Retrieved December 22, 2025, from https://tafsirweb.com/3553-surat-hud-ayat-61.html

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space* (No. arXiv:1301.3781). arXiv. https://doi.org/10.48550/arXiv.1301.3781

Natzir, S. M., & Jatiprasetya, H. (2025). Prediksi Harga Cryptocurrency Xlm Menggunakan Metode Deep Learning Lstm Dan Gru: Predicting Xlm Cryptocurrency Prices Using Lstm and Gru Deep Learning Models. *HOAQ (High Education of Organization Archive Quality) : Jurnal Teknologi Informasi*, *16*(1), 49–58. https://doi.org/10.52972/hoaq.vol16no1.p49-58

Naveen P. (2025). Evaluating the Effectiveness of CNN, LSTM, and Bi-LSTM Models in Classifying Twitter Sentiments. *Journal of Information Systems Engineering and Management*, *10*(15s), 398–406. https://doi.org/10.52783/jisem.v10i15s.2475

Nida Ulhasanah, Yulison Herry Chrisnanto, Melina, & Julian Evan Chrisnanto. (2025). Klasifikasi Multi-Label Jenis dan Warna Buah Menggunakan Convolutional Neural Network (CNN) dengan Encoder Fitur. *TEMATIK*, *12*(1), 72–80. https://doi.org/10.38204/tematik.v12i1.2328

Nistor, S. C., Moca, M., Moldovan, D., Oprean, D. B., & Nistor, R. L. (2021). Building a Twitter Sentiment Analysis System with Recurrent Neural Networks. *Sensors*, *21*(7), 2266. https://doi.org/10.3390/s21072266

Noh, S.-H. (2021). Analysis of Gradient Vanishing of RNNs and Performance Comparison. *Information*, *12*(11), 442. https://doi.org/10.3390/info12110442

Rainio, O., Teuho, J., & Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, *14*(1), 6086. https://doi.org/10.1038/s41598-024-56706-x

Rawal, Ms. P. (2025). Hate Speech Detection Using LSTM and Machine Learning Models. *International Journal for Research in Applied Science and Engineering Technology*, *13*(2), 1–7. https://doi.org/10.22214/ijraset.2025.66791

Rizkyani, E., Ernawati, I., & Chamidah, N. (2022). Klasifikasi Multi-Label Menggunakan Metode Multi-Label K-Nearest Neighbor (ml-Knn) Pada Penyakit Kanker Serviks. *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, *7*(4), 1281–1293. https://doi.org/10.29100/jipi.v7i4.3260

Rong, X. (2016). *Word2vec Parameter Learning Explained* (No. arXiv:1411.2738). arXiv. https://doi.org/10.48550/arXiv.1411.2738

Siregar, F. S. (2023). Literasi Digital Sebagai Upaya Antisipasi Ujaran Kebencian di Media Sosial. *Jurnal SOMASI (Sosial Humaniora Komunikasi)*, *4*(1), 68–76. https://doi.org/10.53695/js.v4i1.929

Susanti, D. I. (2022). Kebebasan Berekspresi dan Ujaran Kebencian: Kajian Filsafat Hukum Terapan. *SAPIENTIA ET VIRTUS*, *7*(2), 100–125. https://doi.org/10.37477/sev.v7i2.363

Susilo, D., Ahmad Chusyairi, & Saputra, M. I. (2025). Prediksi Penjualan Untuk Optimasi Stock Produk Menggunakan Algoritma Long Short Term Memory. *Sistematis : Jurnal Ilmiah Sistem Informasi*, *1*(2), 119–134. https://doi.org/10.69533/56gyat30

Syahid, A., Muhlisin, M., & Fauzan, M. (2025). Tindak Tutur Ujaran Kebencian Bermuatan Antargolongan di Media Sosial Indonesia: Kajian Linguistik Forensik. *MABASAN*, *19*(1). https://doi.org/10.62107/mab.v19i1.1037

Syifa, A. S. A., Umam, K., Handayani, M. R., & Aini, S. N. (2025). Perbandingan Klasifikasi Single-Label dan Multi-Label Ulasan Pengguna Lapangan Futsal di Semarang Menggunakan SVM. *InComTech : Jurnal Telekomunikasi dan Komputer*, *15*(2), 168–185. https://doi.org/10.22441/incomtech.v15i2.33905

Tamam, A. B. (2021). Ujaran Kebencian Di Media Sosial Perspektif Hukum Islam dan Hukum Positif Di Indonesia. *Alamtara: Jurnal Komunikasi Dan Penyiaran Islam*, *5*(1), 1–10. https://doi.org/10.58518/alamtara.v5i1.678

Utama, H. (2023). Pendekatan Deep Learning Menggunakan Metode Lstm Untuk Prediksi Harga Bitcoin. *The Indonesian Journal of Computer Science Research*, *2*(2), 43–50. https://doi.org/10.59095/ijcsr.v2i2.77

Walsh, S., & Greaney, P. (2025). Multiclass hate speech detection with an aggregated dataset. *Natural Language Processing*, 1–17. https://doi.org/10.1017/nlp.2024.62

Yazid, R. M., Umbara, F. R., & Sabrina, P. N. (2024). Deteksi Ujaran Kebencian dengan Metode Klasifikasi Naïve Bayes dan Metode N-Gram pada Dataset Multi-Label Twitter Berbahasa Indonesia. *Informatics and Digital Expert (INDEX)*, *4*(2), 46–52. https://doi.org/10.36423/index.v4i2.894

Zhang, M.-L., & Zhou, Z.-H. (2014). A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819–1837. https://doi.org/10.1109/TKDE.2013.39

Zhang, Y., Li, X., Liu, Y., Li, A., Yang, X., & Tang, X. (2023). A Multilabel Text Classifier of Cancer Literature at the Publication Level: Methods Study of Medical Text Classification. *JMIR Medical Informatics*, *11*(1), e44892. https://doi.org/10.2196/44892

LAMPIRAN

Sourcecode Manual

```
# ================================================================
# Build Model LSTM (MANUAL) - Forward + BPTT + Update
# ================================================================

print_header("TAHAP 8: BUILD MODEL LSTM (MANUAL)")

LSTM_UNITS = 64
DENSE_UNITS = 32
LEARNING_RATE = 0.001

class ManualLSTMClassifier:
    def __init__(self, vocab_size, vector_size, embedding_matrix, max_len,
                 lstm_units=64, dense_units=32, n_labels=6, lr=0.001):

        self.vocab_size = vocab_size
        self.vector_size = vector_size
        self.embedding = embedding_matrix  # trainable=False (tidak kita update)
        self.max_len = max_len

        self.H = lstm_units
        self.D = vector_size
        self.dense_units = dense_units
        self.n_labels = n_labels
        self.lr = lr

        def init_mat(rows, cols, scale=0.08):
            return [[(random.random()*2-1)*scale for _ in range(cols)] for _ in range(rows)]

        def init_vec(n):
            return [0.0]*n

        # LSTM gate weights
        self.Wxi = init_mat(self.D, self.H); self.Whi = init_mat(self.H, self.H); self.bi = init_vec(self.H)
        self.Wxf = init_mat(self.D, self.H); self.Whf = init_mat(self.H, self.H); self.bf = init_vec(self.H)
        self.Wxo = init_mat(self.D, self.H); self.Who = init_mat(self.H, self.H); self.bo = init_vec(self.H)
        self.Wxg = init_mat(self.D, self.H); self.Whg = init_mat(self.H, self.H); self.bg = init_vec(self.H)

        # Dense 1: H -> dense_units
        self.W1 = init_mat(self.H, self.dense_units); self.b1 = init_vec(self.dense_units)
        # Output: dense_units -> n_labels
        self.W2 = init_mat(self.dense_units, self.n_labels); self.b2 = init_vec(self.n_labels)
```

```
    def embed_seq(self, seq):
        out = []
        for idx in seq:
            if 0 <= idx < self.vocab_size:
                out.append(self.embedding[idx])
            else:
                out.append([0.0]*self.D)
        return out

    def matvec(self, M, v):
        out_dim = len(M[0])
        res = [0.0]*out_dim
        for r in range(len(M)):
            vr = v[r]
            if vr == 0.0:
                continue
            row = M[r]
            for c in range(out_dim):
                res[c] += row[c]*vr
        return res

    def add_vec_inplace(self, a, b):
        for i in range(len(a)):
            a[i] += b[i]

    def gate_compute(self, x, hprev, Wx, Wh, b, act):
        z = self.matvec(Wx, x)
        zh = self.matvec(Wh, hprev)
        self.add_vec_inplace(z, zh)
        self.add_vec_inplace(z, b)
        if act == "sigmoid":
            return [sigmoid(v) for v in z], z
        else:
            t = [tanh(v) for v in z]
            return t, z
```

```
    def forward(self, seq):
        X = self.embed_seq(seq)
        T = len(X)

        h = [0.0]*self.H
        c = [0.0]*self.H
        cache = []

        for t in range(T):
            x_t = X[t]

            i_t, zi = self.gate_compute(x_t, h, self.Wxi, self.Whi, self.bi, "sigmoid")
            f_t, zf = self.gate_compute(x_t, h, self.Wxf, self.Whf, self.bf, "sigmoid")
            o_t, zo = self.gate_compute(x_t, h, self.Wxo, self.Who, self.bo, "sigmoid")
            g_t, zg = self.gate_compute(x_t, h, self.Wxg, self.Whg, self.bg, "tanh")

            c_new = [0.0]*self.H
            for j in range(self.H):
                c_new[j] = f_t[j]*c[j] + i_t[j]*g_t[j]

            tanh_c = [tanh(c_new[j]) for j in range(self.H)]
            h_new = [0.0]*self.H
            for j in range(self.H):
                h_new[j] = o_t[j] * tanh_c[j]

            cache.append((x_t, h, c, i_t, f_t, o_t, g_t, tanh_c))
            h, c = h_new, c_new

        # Dense 1
        z1 = [0.0]*self.dense_units
        for j in range(self.dense_units):
            s = self.b1[j]
            for i in range(self.H):
                s += h[i]*self.W1[i][j]
            z1[j] = s
        a1 = [relu(v) for v in z1]

        # Output sigmoid
        z2 = [0.0]*self.n_labels
        for k in range(self.n_labels):
            s = self.b2[k]
            for j in range(self.dense_units):
                s += a1[j]*self.W2[j][k]
            z2[k] = s
        y_prob = [sigmoid(v) for v in z2]

        return y_prob, (cache, h, c, z1, a1, z2)

    def bce_loss(self, y_prob, y_true):
        loss = 0.0
        for k in range(self.n_labels):
            p = y_prob[k]
            y = y_true[k]
            loss += -(y*safe_log(p) + (1-y)*safe_log(1-p))
        return loss

    def backward_and_update(self, seq, y_true):
        y_prob, ctx = self.forward(seq)
        cache, h_last, c_last, z1, a1, z2 = ctx
        T = len(cache)

        # dL/dz2 = p - y
        dz2 = [(y_prob[k] - y_true[k]) for k in range(self.n_labels)]

        # grads W2, b2
        dW2 = [[0.0]*self.n_labels for _ in range(self.dense_units)]
        db2 = dz2[:]
        for j in range(self.dense_units):
            for k in range(self.n_labels):
                dW2[j][k] = a1[j] * dz2[k]

        # grad to a1
        da1 = [0.0]*self.dense_units
        for j in range(self.dense_units):
            s = 0.0
            for k in range(self.n_labels):
                s += self.W2[j][k]*dz2[k]
            da1[j] = s
```

```
        # relu back
        dz1 = [da1[j]*drelu(z1[j]) for j in range(self.dense_units)]

        # grads W1, b1
        dW1 = [[0.0]*self.dense_units for _ in range(self.H)]
        db1 = dz1[:]
        for i in range(self.H):
            for j in range(self.dense_units):
                dW1[i][j] = h_last[i]*dz1[j]

        # grad to h_last
        dh = [0.0]*self.H
        for i in range(self.H):
            s = 0.0
            for j in range(self.dense_units):
                s += self.W1[i][j]*dz1[j]
            dh[i] = s

    # init LSTM grads
    def zero_mat(r,c): return [[0.0]*c for _ in range(r)]
    dWxi=zero_mat(self.D,self.H); dWhi=zero_mat(self.H,self.H); dbi=[0.0]*self.H
    dWxf=zero_mat(self.D,self.H); dWhf=zero_mat(self.H,self.H); dbf=[0.0]*self.H
    dWxo=zero_mat(self.D,self.H); dWho=zero_mat(self.H,self.H); dbo=[0.0]*self.H
    dWxg=zero_mat(self.D,self.H); dWhg=zero_mat(self.H,self.H); dbg=[0.0]*self.H

    dh_next = dh[:]
    dc_next = [0.0]*self.H

    for t in reversed(range(T)):
        x_t, h_prev, c_prev, i_t, f_t, o_t, g_t, tanh_c = cache[t]

        do = [dh_next[j]*tanh_c[j] for j in range(self.H)]
        dc = [0.0]*self.H
        for j in range(self.H):
            dc[j] = dh_next[j]*o_t[j]*dtanh_from_tanh(tanh_c[j]) + dc_next[j]

        df = [dc[j]*c_prev[j] for j in range(self.H)]
        di = [dc[j]*g_t[j] for j in range(self.H)]
        dg = [dc[j]*i_t[j] for j in range(self.H)]

        dzi = [di[j]*dsigmoid_from_sig(i_t[j]) for j in range(self.H)]
        dzf = [df[j]*dsigmoid_from_sig(f_t[j]) for j in range(self.H)]
        dzo = [do[j]*dsigmoid_from_sig(o_t[j]) for j in range(self.H)]
        dzg = [dg[j]*dtanh_from_tanh(g_t[j]) for j in range(self.H)]

        for j in range(self.H):
            dbi[j]+=dzi[j]; dbf[j]+=dzf[j]; dbo[j]+=dzo[j]; dbg[j]+=dzg[j]

        for i in range(self.D):
            xv = x_t[i]
            if xv == 0.0: continue
            for j in range(self.H):
                dWxi[i][j] += xv*dzi[j]
                dWxf[i][j] += xv*dzf[j]
                dWxo[i][j] += xv*dzo[j]
                dWxg[i][j] += xv*dzg[j]

        for i in range(self.H):
            hv = h_prev[i]
            if hv == 0.0: continue
            for j in range(self.H):
                dWhi[i][j] += hv*dzi[j]
                dWhf[i][j] += hv*dzf[j]
                dWho[i][j] += hv*dzo[j]
                dWhg[i][j] += hv*dzg[j]

        dh_prev = [0.0]*self.H
        for i in range(self.H):
            s = 0.0
            for j in range(self.H):
                s += self.Whi[i][j]*dzi[j] + self.Whf[i][j]*dzf[j] + self.Who[i][j]*dzo[j] + self.Whg[i][j]*dzg[j]
            dh_prev[i] = s

        dc_prev = [dc[j]*f_t[j] for j in range(self.H)]
        dh_next = dh_prev
        dc_next = dc_prev
```

```
        # SGD update
        lr = self.lr

        for i in range(self.H):
            for j in range(self.dense_units):
                self.W1[i][j] -= lr*dW1[i][j]
        for j in range(self.dense_units):
            self.b1[j] -= lr*db1[j]

        for j in range(self.dense_units):
            for k in range(self.n_labels):
                self.W2[j][k] -= lr*dW2[j][k]
        for k in range(self.n_labels):
            self.b2[k] -= lr*db2[k]

        def upd_mat(W, dW):
            for i in range(len(W)):
                for j in range(len(W[0])):
                    W[i][j] -= lr*dW[i][j]

        upd_mat(self.Wxi,dWxi); upd_mat(self.Whi,dWhi)
        upd_mat(self.Wxf,dWxf); upd_mat(self.Whf,dWhf)
        upd_mat(self.Wxo,dWxo); upd_mat(self.Who,dWho)
        upd_mat(self.Wxg,dWxg); upd_mat(self.Whg,dWhg)

        for j in range(self.H):
            self.bi[j] -= lr*dbi[j]
            self.bf[j] -= lr*dbf[j]
            self.bo[j] -= lr*dbo[j]
            self.bg[j] -= lr*dbg[j]

        return y_prob

    def predict_proba(self, X):
        out = []
        for seq in X:
            y_prob, _ = self.forward(seq)
            out.append(y_prob)
        return out
```

```
# Inisialisasi model
model = ManualLSTMClassifier(
    vocab_size=VOCAB_SIZE,
    vector_size=VECTOR_SIZE,
    embedding_matrix=embedding_matrix,
    max_len=max_len,
    lstm_units=LSTM_UNITS,
    dense_units=DENSE_UNITS,
    n_labels=len(LABEL_COLUMNS),
    lr=LEARNING_RATE
)

print("Model LSTM manual siap digunakan.")
```

```python
# ================================================================
# Training Model (MANUAL)
# ================================================================

print_header("TAHAP 9: TRAINING PROCESS (MANUAL)")

BATCH_SIZE = 64   # tetap ditampilkan agar konsisten dengan skenario, meski update kita per-sample
EPOCHS = 20        # naikkan kalau mau (akan sangat lama)

def train_model_manual(model, X_train, y_train, epochs=1):
    n = len(X_train)
    idx = list(range(n))

    for ep in range(1, epochs+1):
        random.shuffle(idx)
        t0 = time.time()
        loss_sum = 0.0

        for step, ii in enumerate(idx, start=1):
            seq = X_train[ii]
            yt  = y_train[ii]

            # forward untuk loss
            y_prob, _ = model.forward(seq)
            loss_sum += model.bce_loss(y_prob, yt)

            # backward + update
            model.backward_and_update(seq, yt)

            if step % 200 == 0:
                print(f"step {step}/{n} - avg_loss: {loss_sum/step:.6f}")

        dt = time.time() - t0
        print(f"[LSTM] Epoch {ep}/{epochs} - avg_loss: {loss_sum/max(1,n):.6f} - time: {dt:.1f}s")

train_model_manual(model, X_train, y_train, epochs=EPOCHS)
print("\nTraining selesai.")
```

```python
# ================================================================
# Evaluasi Performa Model (MANUAL)
# ================================================================

print_header("TAHAP 10: EVALUASI PERFORMA MODEL (MANUAL)")

THRESHOLD = 0.3
SKENARIO_NOMOR = 18
SKENARIO_ID = f"Skenario {SKENARIO_NOMOR} ({VECTOR_SIZE}d-{LSTM_UNITS}H-{BATCH_SIZE}B-{THRESHOLD}t)"
print("ID Skenario:", SKENARIO_ID)

def threshold_preds(y_prob_list, thr):
    return [[1 if p >= thr else 0 for p in probs] for probs in y_prob_list]

def subset_accuracy(y_true, y_pred):
    correct = 0
    for yt, yp in zip(y_true, y_pred):
        if yt == yp:
            correct += 1
    return correct / max(1, len(y_true))

def hamming_loss_manual(y_true, y_pred):
    if not y_true:
        return 0.0
    n = len(y_true)
    L = len(y_true[0])
    wrong = 0
    for yt, yp in zip(y_true, y_pred):
        for a, b in zip(yt, yp):
            if a != b:
                wrong += 1
    return wrong / (n * L)

def multilabel_confusion(y_true, y_pred, n_labels):
    cms = []
    for k in range(n_labels):
        TN = FP = FN = TP = 0
        for yt, yp in zip(y_true, y_pred):
            t = yt[k]
            p = yp[k]
            if t == 0 and p == 0: TN += 1
            elif t == 0 and p == 1: FP += 1
            elif t == 1 and p == 0: FN += 1
            else: TP += 1
        cms.append([[TN, FP], [FN, TP]])
    return cms
```

```python
    def precision_recall_f1(cm):
        TN, FP = cm[0]
        FN, TP = cm[1]
        prec = TP / (TP + FP) if (TP + FP) > 0 else 0.0
        rec  = TP / (TP + FN) if (TP + FN) > 0 else 0.0
        f1   = (2 * prec * rec / (prec + rec)) if (prec + rec) > 0 else 0.0
        sup  = TP + FN
        return prec, rec, f1, sup

    def print_report(cms, labels):
        print("\nLaporan Klasifikasi Per Label (manual):")
        print("Label".ljust(16), "precision".rjust(10), "recall".rjust(10), "f1-score".rjust(10), "support".rjust(10))

        macro_p = macro_r = macro_f = 0.0
        total_support = 0

        for i, cm in enumerate(cms):
            p, r, f1, sup = precision_recall_f1(cm)
            macro_p += p; macro_r += r; macro_f += f1
            total_support += sup
            print(labels[i].ljust(16), f"{p:10.4f}", f"{r:10.4f}", f"{f1:10.4f}", f"{sup:10d}")

        m = len(labels)
        if m > 0:
            macro_p /= m; macro_r /= m; macro_f /= m

        print("macro avg".ljust(16), f"{macro_p:10.4f}", f"{macro_r:10.4f}", f"{macro_f:10.4f}", f"{total_support:10d}")
    # ========================
    # A. Prediksi
    # ========================
    print(f"\nMenganalisis Data Test dengan Threshold {THRESHOLD} ...")
    y_pred_proba = model.predict_proba(X_test)
    y_pred_binary = threshold_preds(y_pred_proba, THRESHOLD)

    # ========================
    # B. Bukti kerja threshold (1 sampel)
    # ========================
    print("\n" + "-"*60)
    print(f"[BUKTI] Simulasi Threshold (Batas = {THRESHOLD})")
    print("-"*60)

    if len(y_pred_proba) > 0:
        sample_idx = 0
        raw_probs = y_pred_proba[sample_idx]
        final_pred = y_pred_binary[sample_idx]

        print(f"Sampel Data Test ke-{sample_idx}:")
        print(f"{'LABEL':<15} | {'PROBABILITAS':<12} | {'SYARAT'} | {'HASIL'}")
        print("-"*60)
        for i, label in enumerate(LABEL_COLUMNS):
            prob = raw_probs[i]
            decision = final_pred[i]
            tanda = ">= " if prob >= THRESHOLD else "<  "
            status = "YA (1)" if decision == 1 else "TIDAK (0)"
            print(f"{label:<15} | {prob:.5f}      | {tanda}{THRESHOLD}  | {status}")
        print("-"*60)

    # ========================
    # C. Metrik utama
    # ========================
    subset_acc = subset_accuracy(y_test, y_pred_binary)
    hamming = hamming_loss_manual(y_test, y_pred_binary)

    print("\nRingkasan Metrik Global (manual):")
    print(f"- Subset Accuracy (Exact Match): {subset_acc:.4f}")
    print(f"- Hamming Loss                 : {hamming:.4f}")

    # ========================
    # D. Confusion matrix + report
    # ========================
    cms = multilabel_confusion(y_test, y_pred_binary, n_labels=len(LABEL_COLUMNS))

    print_report(cms, LABEL_COLUMNS)

    print("\nConfusion Matrix (manual) per label format [[TN,FP],[FN,TP]]:")
    for i, cm in enumerate(cms):
        print(f"- {LABEL_COLUMNS[i]}: {cm}")
```