Politeknik Caltex Riau

LAPORAN TESIS

KLASIFIKASI *CYBERBULLYING* PADA MEDIA SOSIAL MENGGUNAKAN MODEL *HYBRID* CNN-BiLSTM

Sari Fauzia Elza
NIM. 2356102011

DOSEN PEMBIMBING
Dr. Yohana Dewi Lulu Widyasari, S.Si., M.T.
Yuliska, S.T., M.Eng.

PROGRAM STUDI
MAGISTER TERAPAN TEKNIK KOMPUTER
POLITEKNIK CALTEX RIAU
2025

Politeknik Caltex Riau

TESIS

KLASIFIKASI *CYBERBULLYING* PADA MEDIA SOSIAL MENGGUNAKAN MODEL *HYBRID* CNN-BiLSTM

Sari Fauzia Elza
NIM. 2356102011

DOSEN PEMBIMBING
Dr. Yohana Dewi Lulu Widyasari, S.Si., M.T.
Yuliska, S.T., M.Eng.

PROGRAM STUDI
MAGISTER TERAPAN TEKNIK KOMPUTER
POLITEKNIK CALTEX RIAU
2025

HALAMAN PENGESAHAN

KLASIFIKASI *CYBERBULLYING* PADA MEDIA SOSIAL MENGGUNAKAN MODEL *HYBRID* CNN-BiLSTM

Oleh :

Sari Fauzia Elza
NIM. 2356102011

Tesis ini digunakan sebagai salah satu syarat untuk memperoleh Gelar Magister Terapan Teknik Komputer (M.Tr.Kom.)
di
Politeknik Caltex Riau
2025

Disetujui oleh :

Pembimbing Utama	:	Dr. Yohana Dewi Lulu Widyasari, S.Si., M.T. NIP. 007717
Pembimbing Pendamping	:	Yuliska, S.T., M.Eng. NIP. 199105
Penguji	:	Dr.Eng Yoanda Alim Syahbana, S.T., M.Sc. NIP. 148809
Penguji	:	Ananda, S.Kom, M.T., Ph.D. NIP. 108501
Penguji	:	Memen Akbar, S.Si., M.T. NIP. 078313

Mengetahui,
Ketua Program Studi Magister Terapan Teknik Komputer
Politeknik Caltex Riau

Dr.Eng Yoanda Alim Syahbana, S.T., M.Sc.
NIP. 148809

HALAMAN PERNYATAAN ORISINALITAS

Dengan ini saya menyatakan bahwa bagian atau keseluruhan tesis ini :

1. Adalah hasil karya sendiri dan tidak mengandung unsur plagiat dari pihak lain
2. Tidak pernah diajukan untuk mendapatkan gelar akademis pada suatu perguruan tinggi.
3. Tidak pernah dipublikasikan atau ditulis oleh pihak lain.
4. Mencantumkan rujukan dan kutipan dengan jujur dan benar terhadap sumber referensi lain yang menunjang pembahasan pada tesis ini.

Apabila ditemukan bukti bahwa pernyataan saya diatas tidak benar, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Politeknik Caltex Riau.

Pekanbaru, 10 Juli 2025
Yang menyatakan,

Sari Fauzia Elza

HALAMAN KESEPAKATAN PUBLIKASI

Demi pengembangan ilmu pengetahuan, dengan ini saya menyatakan :

1. Memberikan persetujuan kepada Politeknik Caltex Riau untuk menyimpan, mengolah dalam bentuk pangkalan data, merawat, mengalihmedia/ formatkan dan mempublikasikan tesis ini selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.
2. Tidak melakukan alihmedia/format dan publikasi dalam bentuk makalah ilmiah dari bagian atau keseluruhan tesis ini ke suatu publikasi ilmiah, pada seminar ataupun jurnal, skala nasional ataupun internasional, kecuali ada persetujuan dari saya dan Dosen Pembimbing Utama, dan mencantumkan nama saya, Dosen Pembimbing Utama dan nama-nama lain (jika ada) yang berkontribusi pada makalah.

Pekanbaru, 10 Juli 2025
Yang menyatakan,

Sari Fauzia Elza

KATA PENGANTAR

Bismillahirrahmanirrahim
Assalaamu'alaikum wa rahmatullahi wa barakaatuh

Alhamdulillah, segala puji dan syukur penulis panjatkan ke hadirat Allah Subhanahu wa Ta'ala, Tuhan semesta alam, atas limpahan rahmat, taufik, dan hidayah-Nya. Shalawat serta salam semoga selalu tercurah kepada Nabi Muhammad Shallallahu 'Alaihi Wasallam, keluarga, sahabat, dan seluruh umatnya. Amin. Sehingga penulis dapat menyelesaikan penulisan tesis ini dengan judul "Klasifikasi *Cyberbullying* Pada Media Sosial Menggunakan Model Hybrid CNN-BiLSTM."

Tesis ini disusun sebagai salah satu syarat untuk menyelesaikan jenjang Pendidikan Magister Terapan pada Program Studi Teknik Komputer Politeknik Caltex Riau. Pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada pihak yang telah banyak memberikan bantuan dan dukungan yang tiada terhingga, baik secara langsung maupun tidak langsung. Ucapan terima kasih penulis sampaikan kepada:

1. Kedua orang tua tercinta, adik, kakak dan keluarga yang selalu menjadi sumber doa, kasih sayang, dan kekuatan dalam setiap langkah hidup dan penyelesaian tesis ini.
2. Bapak Dr. Dadang Syarif Sihabudin Sahid, S.Si., M.Sc., selaku Direktur Politeknik Caltex Riau.
3. Bapak Dr.Eng Yoanda Alim Syahbana, S.T., M.Sc selaku Ketua Program Studi Magister Terapan Teknik Komputer Politeknik Caltex Riau dan juga selaku Penguji I.
4. Bapak Ananda, S.Kom., M.T., Ph.D., selaku Koordinator Tesis Program Studi Magister Terapan Teknik Komputer Politeknik Caltex Riau dan juga selaku Penguji II.
5. Ibu Dr. Yohana Dewi Lulu Widyasari, S.Si., M.T., selaku Pembimbing I, dan Ibu Yuliska, S.T., M.Eng., selaku Pembimbing II., yang telah membimbing dalam penelitian dan penulisan tesis ini.
6. Bapak Memen Akbar, S.Si., M.T. selaku Dosen Penguji III yang telah bersedia menguji dan memberikan saran dalam penyempurnaan tesis ini.
7. Bapak/Ibu Dosen Magister Terapan Teknik Komputer serta staf Politeknik Caltex Riau yang telah memberikan pengetahuan berharga dan pelayanan administrasi akademik selama masa perkuliahan.

8. Teman seperjuangan 23MTTK A yang telah memberikan semangat, dukungan, dan kerja sama selama masa studi hingga penyelesaian tesis ini.

Penulis menyadari bahwa tesis ini masih jauh dari sempurna, oleh karena itu kritik dan saran yang membangun sangat diharapkan untuk perbaikan di masa mendatang.

Akhir kata, semoga tesis ini dapat memberikan manfaat bagi pengembangan ilmu pengetahuan dan menjadi referensi yang berguna bagi pembaca serta lain yang tertarik dalam bidang ini.

Wassalaamu'alaikum wa rahmatullahi wa barakaatuh

Pekanbaru, 10 Juli 2025
Yang menyatakan,

Sari Fauzia Elza

ABSTRAK

Cyberbullying merupakan salah satu dampak negatif dari perkembangan media sosial yang dapat menimbulkan gangguan psikologis dan sosial serius bagi korbannya. Platform X (sebelumnya Twitter) menjadi salah satu media sosial dengan tingkat kasus *cyberbullying* tertinggi, dengan karakteristik bahasa informal, singkatan, dan kata tidak baku yang menyulitkan deteksi otomatis. Penelitian ini bertujuan mengembangkan model klasifikasi komentar *cyberbullying* menggunakan pendekatan *hybrid* CNN–BiLSTM yang dikombinasikan dengan teknik *word embedding*. CNN digunakan untuk mengekstraksi fitur spasial, sedangkan BiLSTM memahami konteks kalimat secara dua arah. Eksperimen dilakukan menggunakan tiga jenis *word embedding*, yaitu FastText, GloVe, dan Word2Vec. Hasil penelitian menunjukkan bahwa model CNN–BiLSTM dengan *embedding* FastText berdimensi 300 memberikan performa terbaik, dengan *F1-score* sebesar 0,9238, AUC 0,9410, serta *precision* dan *recall* masing-masing sebesar 0,9238. FastText terbukti unggul dalam menangani kata-kata tidak baku melalui pendekatan *subword*, sehingga mampu meningkatkan representasi teks informal. Temuan ini menunjukkan bahwa kombinasi arsitektur *hybrid* CNN–BiLSTM dan *embedding* yang sesuai efektif dalam meningkatkan performa klasifikasi *cyberbullying* pada media sosial berbahasa Indonesia.

Kata kunci – Cyberbullying, Klasifikasi, CNN, BiLSTM*, Word embedding*

ABSTRACT

Cyberbullying represents a significant adverse effect of the rise of social media, leading to severe psychological and social issues for those affected. Among social media platforms, X (previously known as Twitter) has one of the highest occurrences of cyberbullying. This is largely due to its use of informal language, abbreviations, and non-standard vocabulary, which complicates automatic detection efforts. The objective of this research is to create a model for classifying comments related to cyberbullying by employing a hybrid approach that combines Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) networks, along with word embedding techniques. The CNN is responsible for extracting spatial features, while the BiLSTM captures the contextual meanings of sentences in both directions. We conducted experiments using three different types of word embeddings: FastText, GloVe, and Word2Vec. The findings reveal that the CNN–BiLSTM model utilizing 300-dimensional FastText embeddings delivered the highest performance, achieving an F1-score of 0.9238, an AUC of 0.9410, and both precision and recall scores of 0.9238. FastText demonstrated its effectiveness in managing non-standard words through its subword methodology, which improved the representation of informal text. These results suggest that the integration of the hybrid CNN–BiLSTM framework with suitable word embedding techniques significantly enhances the classification performance of cyberbullying on Indonesian-language social media.

Keywords – Cyberbullying, Classification, CNN, BiLSTM, Word Embedding

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

DAFTAR ISTILAH

Korpus	:	Kumpulan teks atau dokumen yang digunakan sebagai sumber data untuk pelatihan dan pengujian dalam pemrosesan bahasa alami (NLP).
Precission/presisi	:	Ukuran seberapa tepat model dalam mengklasifikasikan data positif.
Accuracy/ Akurasi	:	Proporsi total prediksi yang benar dibandingkan dengan seluruh data.
Recall	:	Ukuran seberapa baik model dalam menemukan semua data positif yang sebenarnya.
AUC	:	*Area Under Curve* Nilai yang menunjukkan seberapa baik model membedakan antara kelas positif dan negatif berdasarkan kurva ROC.
F1-Score	:	Harmonik rata-rata antara *precision* dan *recall*. Berguna ketika data tidak seimbang.
SMOTE	:	Teknik untuk menyeimbangkan data yang tidak seimbang dengan cara membuat contoh sintetis dari kelas minoritas, sehingga model tidak bias terhadap kelas mayoritas.
Epoch	:	Satu kali proses pelatihan penuh model terhadap seluruh data pelatihan.
Overfitting	:	Kondisi saat model terlalu menyesuaikan diri dengan data latih hingga kehilangan kemampuan generalisasi ke data baru.
Dropout	:	Teknik regularisasi yang secara acak "menghapus" sejumlah neuron saat pelatihan untuk mencegah overfitting dan meningkatkan kemampuan generalisasi model.
Training Time	:	Waktu yang dibutuhkan model untuk menyelesaikan proses pelatihan hingga selesai.
OOV	:	(*Out-of-Vocabulary*) Kata-kata yang tidak terdapat dalam kosakata *embedding* yang digunakan oleh model.

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Media sosial adalah media di internet yang memungkinkan pengguna merepresentasikan dirinya maupun berinteraksi, bekerja sama, berbagi, berkomunikasi dengan pengguna lain membentuk ikatan sosial secara *virtual* (Fauziah et al., 2024) . Namun, media sosial juga membawa dampak negatif yang tidak dapat diabaikan (Silitonga, 2023). Salah satu dampak negatifnya adalah maraknya kejahatan di dunia maya (*cybercrime*), termasuk *cyberbullying* (Dwipayana et al., 2020). Platform X (sebelumnya dikenal sebagai Twitter), merupakan salah satu media sosial yang memiliki jumlah pengguna aktif yang tinggi dan banyak digunakan di seluruh dunia. Hingga April 2024, platform ini memiliki sekitar 611 juta pengguna aktif dan menempati peringkat ke-12 sebagai media sosial paling banyak digunakan di dunia (DataReportal, 2024). Direktur Jenderal Sumber Daya Perangkat Pos dan Informatika (SDPP) Kementerian Komunikasi dan Informatika, Budi Setiawan, menyatakan bahwa Indonesia menempati urutan ke-5 dunia sebagai pengguna terbanyak platform X setelah Inggris (Informatika, 2024) Sayangnya, *cyberbullying* melalui media sosial paling sering terjadi pada platform X (Twitter), yang menempati posisi tertinggi dengan persentase sebesar 45,5%. Disusul oleh Facebook sebesar 38,6% dan Instagram sebesar 13,7% (Whittaker & Kowalski, 2015).

Menurut *Think Before Text* pada laman online UNICEF menjelaskan bahwa *cyberbullying* merupakan perilaku agresif secara berulang melalui media elektronik yang dilakukan kepada seseorang atau sekelompok orang yang dianggap sulit melawan (UNICEF, 2020). Kementerian Komunikasi dan Informatika Republik Indonesia bersama UNICEF pernah melakukan penelitian tentang *cyberbullying* di 11 Provinsi Indonesia dengan sampel sebanyak 400 orang usia 10-19 tahun (Martha, 2024). Hasil penelitian menunjukkan bahwa sebagian besar remaja Indonesia pernah menjadi korban *cyberbullying* berupa ancaman, hinaan, dan mempermalukan korban. Dampak dari *cyberbullying* juga dapat menimbulkan kerusakan psikologis yang parah dan berkelanjutan bagi korban antara lain, mudah depresi, marah, timbul perasaan gelisah, cemas, menyakiti diri sendiri dan percobaan bunuh diri. Sedangkan dampak sosial dari korban *cyberbullying* adalah menarik diri, kehilangan kepercayaan diri, lebih agresif kepada teman dan keluarga (Harinsa Putri & Ina Savira, 2022).

Melihat dampak serius yang ditimbulkan, melakukan deteksi indikasi potensi dan klasifikasi *cyberbullying* dan *non-cyberbullying* dalam penggunaan sosial media dapat menjadi langkah awal dalam penanganan kasus *cyberbullying* (Nugraha Manoppo & Hatta Fudholi, 2021).

Penelitian terkait klasifikasi *cyberbullying* pada media sosial telah dilakukan dengan berbagai pendekatan. Penelitian pertama membahas deteksi *cyberbullying* pada pemain sepak bola di platform media sosial "X" menggunakan metode *Long Short-Term Memory* (LSTM). Model ini dipilih karena kemampuannya dalam memproses data teks secara sekuensial dan mengatasi masalah *long-term dependencies*. Hasil penelitian menunjukkan akurasi sebesar 78%, namun model ini masih menghadapi kendala dalam memahami bahasa informal dan tidak baku yang umum digunakan pada platform X (Pawit Widiyantoro et al., 2025). Selanjutnya, penelitian kedua menggunakan model kombinasi CNN-LSTM untuk mendeteksi komentar *cyberbullying* pada platform YouTube. Kombinasi CNN dan LSTM digunakan karena CNN dapat mengekstraksi fitur spasial dari teks, sementara LSTM memahami urutan kata secara lebih baik. Hasil penelitian menunjukkan *F1-score* sebesar 0,841, namun model ini masih mengalami kendala dalam menangani kata-kata yang tidak dikenali (*Out-Of-Vocabulary/*OOV) dan keberagaman bahasa dan penggunaan istilah yang tidak baku (Andika et al., 2023). Permasalahan pada metode-metode sebelumnya, menunjukkan kebutuhan akan pendekatan yang lebih komprehensif dan adaptif dalam memahami konteks teks.

Permasalahan umum yang dihadapi dalam klasifikasi teks media sosial adalah tingginya keberagaman gaya bahasa, termasuk penggunaan singkatan, slang, dan kesalahan ketik (*typo*). Kondisi ini menyebabkan model kesulitan dalam merepresentasikan makna kata secara akurat, yang pada akhirnya dapat menurunkan performa klasifikasi (Lu et al., 2020) . Menurut (Khan et al., 2025) *word embedding* mampu mengurangi pengaruh dari *Out-Of-Vocabulary* (OOV) dengan merepresentasikan kata ke dalam bentuk vektor berdasarkan konteks atau struktur sub-kata, sehingga memungkinkan model tetap dapat memahami makna kata yang belum pernah muncul dalam data pelatihan.

Oleh karena itu, penggunaan metode *hybrid* CNN-BiLSTM dan penerapan teknik w*ord embedding* pada penelitian ini diharapkan mampu mengintegrasikan keunggulan CNN dalam mengekstraksi fitur spasial, dengan kemampuan BiLSTM yang mampu memahami konteks kalimat secara dua arah (*bidirectional*) yaitu dengan membaca urutan kata dari awal

ke akhir (*forward*) dan dari akhir ke awal (*backward*) (Setyaningrum & Nadhif, 2025). Selain itu, penggunaan teknik w*ord embedding* diharapakan mampu meningkatkan efektivitas model dalam mengidentifikasi komentar yang mengandung unsur *cyberbullying pada media sosial*. Penelitian ini akan menggunakan beberapa varian *word embedding* seperti *FastText***,** *Word2Vec***,** dan *GloVe***,** yang masing-masing akan di *fine-tune* agar lebih sesuai dengan karakteristik data berbahasa Indonesia. Sebagai pembanding, model lain seperti LSTM, CNN, dan CNN-LSTM juga diuji untuk memberikan analisis performa yang lebih menyeluruh dalam klasifikasi komentar *cyberbullying* berbahasa Indonesia.

1.2 Permasalahan

Berdasarkan latar belakang di atas, maka dapat dirumuskan beberapa permasalahan sebagai berikut:

1. Bagaimana membangun model *hybrid* CNN-BiLSTM dengan penerapan teknik *word embedding* dalam klasifikasi komentar *cyberbullying* pada media sosial?
2. Bagaimana kinerja model *hybrid* CNN-BiLSTM dalam melakukan klasifikasi komentar sebagai *cyberbullying* atau *non-cyberbullying* pada media sosial?

1.3 Tujuan

Adapun tujuan yang didapat dari penelitian ini adalah sebagai berikut:

1. Mengembangkan model *hybrid* CNN-BiLSTM dengan penerapan teknik *word embedding* dalam klasifikasi komentar *cyberbullying* pada media sosial.
2. Mengevaluasi kinerja model CNN-BiLSTM menggunakan metrik evaluasi, yaitu akurasi, *precision, recall, dan F1-score* dalam klasifikasi komentar *cyberbullying* dan *non-cyberbullying*.

1.4 Manfaat

Adapun manfaat yang akan didapat dari penelitian ini adalah sebagai berikut:

1. Menghasilkan model *hybrid* CNN-BiLSTM yang mampu melakukan klasifikasi komentar *cyberbullying* dan *non-cyberbullying.*
2. Model yang dihasilkan dapat digunakan sebagai bahan untuk *transfer learning* guna membangun sistem otomatisasi dalam mendeteksi dan klasifikasi komentar-komentar yang mengandung unsur *cyberbullying*.

3. Menjadi acuan bagi penelitian selanjutnya untuk melakukan pengembangan sistem ataupun peningkatan performa.

1.5 Sistematika Penulisan

Sistematika penulisan adalah struktur atau urutan yang digunakan dalam menyusun sebuah karya tulis. Dengan sistematika penulisan membantu pembaca untuk mengikuti alur pikiran penulis dan memahami informasi secara sistematis.

BAB 1 PENDAHULUAN

Bab ini memberikan gambaran umum tentang latar belakang penelitian, permasalahan yang akan diteliti, tujuan penelitian, ruang lingkup penelitian, serta metode penelitian yang akan digunakan. Pendahuluan ini bertujuan untuk memberikan pemahaman kepada pembaca tentang pentingnya penelitian tersebut dan apa yang diharapkan dari hasil penelitian tersebut.

BAB 2 KAJIAN PUSTAKA

Bab ini akan menyajikan tinjauan literatur terkait topik yang diteliti. Tinjauan literatur ini akan membahas penelitian-penelitian terdahulu yang relevan dengan topik, teori-teori yang mendukung penelitian, serta kerangka konseptual yang akan digunakan.

BAB 3 DESAIN SISTEM

Bab ini merupakan bagian penting dari sebuah penelitian tesis dimana menguraikan secara detail langkah-langkah yang akan diambil untuk menjalankan penelitian. Pada bab ini, akan menjelaskan tentang metode penelitian yang dipilih, alat atau teknik yang digunakan untuk pengumpulan data, serta prosedur analisis yang akan diterapkan

BAB 4 PEMBAHASAN DAN ANALISIS

Bab ini mencakup beberapa sub-bab yang menjelaskan secara rinci tentang prosedur eksperimen, metode pengumpulan data, analisis data, dan interpretasi hasil. Bagian analisis data merupakan inti dari bab ini, dimana melakukan pengolahan dan analisis terhadap data yang telah dikumpulkan. Hal ini bisa melibatkan berbagai teknik statistik, model matematis, atau algoritma komputasi tergantung pada jenis penelitian. Hasil dari analisis data ini kemudian disajikan dalam bentuk tabel, grafik, atau visualisasi lainnya untuk memudahkan pembaca dalam memahami temuan penelitian.

BAB 5 PENUTUP

Bab ini merupakan penutup yang mencakup kesimpulan dan saran. Pada bagian kesimpulan, berisikan rangkuman temuan utama penelitian yang menjawab pertanyaan atau hipotesis yang diajukan pada bab pendahuluan, memastikan bahwa kesimpulan tersebut didasarkan pada data dan analisis yang telah dipaparkan sebelumnya. Bagian saran memberikan rekomendasi berdasarkan temuan penelitian, baik dalam bentuk implikasi praktis, rekomendasi untuk penelitian lebih lanjut.

BAB 2
KAJIAN PUSTAKA

Bab ini membahas mengenai kajian teori dari permasalahan yang akan diteliti yaitu penerapan model *hybrid learning* menggunakan CNN-BiLSTM. Selain itu pada bab ini juga menjelaskan mengenai penelitian terdahulu yang memiliki keterkaitan dan menjadi landasan maupun kerangka konsep dari penelitian ini.

2.1 Teori Penunjang

Teori penunjang membahas mengenai kajian teori dari permasalahan yang akan diteliti, yaitu penerapan model *hybrid learning* menggunakan CNN-BiLSTM dan teknik *word embedding*. Selain itu, pada bab ini juga dijelaskan mengenai penelitian terdahulu yang memiliki keterkaitan dan menjadi landasan maupun kerangka konsep dari penelitian ini.

2.1.1 Cyberbullying

Menurut teori Willard dijelaskan bahwa *cyberbullying* adalah sebuah aktivitas mengunggah ataupun mengirimkan konten yang membahayakan/berupa agresi sosial melalui internet dan teknologi lain (Willard, 2005). *Cyberbullying* ialah *cyberbullying*atau perundungan dengan menggunakan teknologi digital. Hal ini dapat terjadi di medial sosial, *platform chatting, platform* bermain *game,* dan ponsel. *Cyberbullying* adalah perilaku agresif dan bertujuan yang dilakukan suatu kelompok atau individu, menggunakan media elektronik, secara berulang-ulang dari waktu ke waktu, terhadap seseorang yang dianggap tidak mudah melakukan perlawanan atas tindakan tersebut. Badan Pengembangan dan Pembinaan Bahasa pada tahun 2019 menyediakan daftar kosa kata perundungan dengan peringkat 1—20 berdasarkan penelitian forensik linguistik yang dilakukan. Kata-kata kunci yang digunakan dalam pencarian data tuturan tersebut meliputi: "*lu, gw, sok, otak, norak, ngaca, tolol, bangsat, tai, orang, anjir, dekil, mati, bodo, anak, miskin, doang, najis, dasar, dan jelek*" (Sukma et al., 2024).

Bentuk *cyberbullying* menurut (Willard, 2005) :

Berikut ini beberapa teknik yang dapat dilakukan dalam pemrosesan data :

a. *Flamming*

Flamming adalah pertarungan online atau pertengkaran sengit menggunakan pesan elektronik di ruang obrolan, melalui pesan instan atau melalui email dengan bahasa kasar dan vulgar. Penggunaan huruf kapital, gambar, dan simbol menambah emosi pada argumen mereka. Contoh: Seorang pengguna membalas cuitan orang lain dengan kalimat, "KAMU NGOMONG APA SIH? DASAR BODOH!!! COBA CERNA DULU BARU NGETIK!"

b. *Harrasment*
Harassment (Gangguan) berulang kali mengirim pesan yang menyinggung, kasar, dan menghina sering kali dikirim setiap saat, siang dan malam. Beberapa bahkan mungkin memposting pesan mereka ke forum publik, ruang obrolan atau papan buletin di mana orang lain dapat melihat ancaman. Contoh: Seseorang terus-menerus me-*mention* pengguna lain dengan kata-kata seperti "Dasar lo jelek! Mati aja!", baik siang maupun malam.

c. *Denigration*
Denigration (Fitnah) menyebarkan informasi tentang orang lain yang menghina dan tidak benar dengan mempostingnya di halaman Web, mengirimkannya ke orang lain melalui email atau pesan instan, atau memposting atau mengirim foto seseorang yang diubah secara digital. Contoh: Akun anonim membuat *thread* tentang seseorang dengan menulis "Dia itu pelakor, jangan percaya tampangnya," lalu menyertakan foto korban.

d. *Impersonation* (Peniruan) membobol email atau akun jejaring sosial dan menggunakan identitas online orang tersebut untuk mengirim atau memposting materi yang kejam atau memalukan kepada atau tentang orang lain. Contoh**:** Pelaku membuat akun dengan nama dan foto korban, lalu mencuit hal-hal kontroversial seperti hinaan terhadap kelompok tertentu, sehingga korban mendapatkan kecaman publik.

e. *Pseudonyms*
Pseudonyms Pseudonyms (Nama Samaran) menggunakan alias atau nama panggilan online untuk merahasiakan identitas mereka. Orang lain yang online hanya mengenal mereka dengan nama samaran ini yang mungkin tidak berbahaya atau menghina. Contoh: Akun bernama "@truthhurts123" secara rutin membalas cuitan korban dengan ejekan seperti, "Lo nggak layak jadi manusia, ngaca deh."

f. *Pseudonyms*
Pseudonyms Pseudonyms (Nama Samaran) menggunakan alias atau nama panggilan online untuk merahasiakan identitas mereka. Orang

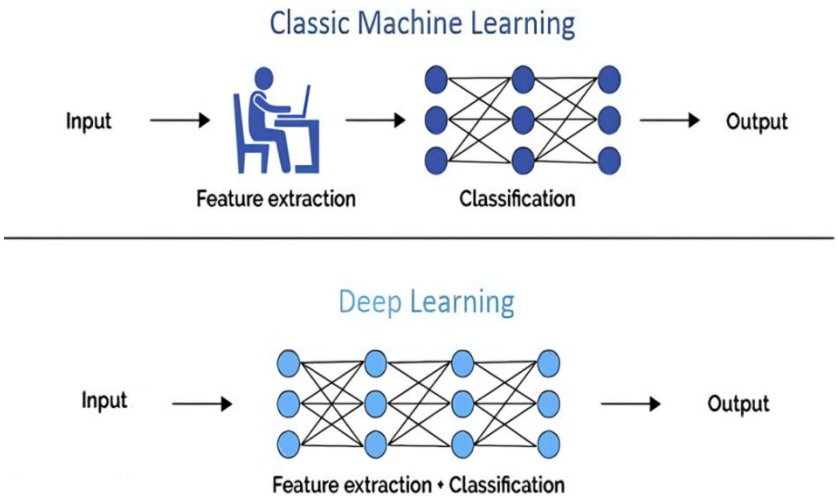
lain yang online hanya mengenal mereka dengan nama samaran ini yang mungkin tidak berbahaya atau menghina. Contoh: Akun bernama "@truthhurts123" secara rutin membalas cuitan korban dengan ejekan seperti, "Lo nggak layak jadi manusia, ngaca deh."

g. *Outing*
   Outing adalah membagikan rahasia atau informasi yang memalukan seseorang di media sosial. Seseorang memposting tangkapan layar DM dari korban yang berisi curhatan pribadi, lalu menambahkan caption yang mempermalukan seperti "Lihat nih, orang ini ternyata *insecure* banget."
h. *Cyberstalking*
   Cyberstalking merupakan bentuk pelecehan, berulang kali mengirim pesan yang mencakup ancaman bahaya atau sangat mengintimidasi, atau terlibat dalam aktivitas online lainnya yang membuat seseorang takut akan keselamatannya. Biasanya pesan dikirim melalui komunikasi pribadi seperti email atau pesan teks. Tergantung pada isi pesannya, mungkin juga illegal. Contoh: Seseorang membalas semua cuitan korban dengan pesan ancaman seperti, "Awas aja loo, kalo ketemu gue ancurin lo."
i. *Masquerading*
   Masquerading (Penyamaran) berpura-pura menjadi orang lain dengan membuat email palsu, alamat, atau nama pesan instan. Mereka mungkin juga menggunakan email atau ponsel orang lain sehingga seolah-olah ancaman tersebut dikirim oleh orang lain. Contoh: Seorang pelaku membuat akun palsu di media sosial X dengan menggunakan nama dan foto milik korban, lalu mencuit pernyataan kontroversial seperti ujaran kebencian terhadap kelompok tertentu. Akibatnya, banyak pengguna lain yang mengecam korban karena mengira akun tersebut benar-benar miliknya.

2.1.2 Deep Learning

Deep learning dipopulerkan oleh Geoffrey Hinton. Pada tahun 2006 ia memperkenalkan jaringan saraf tiruan *(neural network*) dan mempopulerkannya dengan istilah *deep learning*. *Deep learning* merupakan sub-bidang dari *machine learning* yang menggunakan algoritma *neural network* untuk memproses data dan belajar dari data tersebut. *Deep learning* memungkinkan komputer untuk memproses data yang lebih kompleks dan menemukan pola yang lebih rumit daripada *machine learning* tradisional. Berbagai macam pekerjaan memanfaatkan *deep learning* contohnya klasifikasi, memprediksi peluang atau kejadian,

mengenali objek, hingga mendiagnosa penyakit. Selain itu, *deep learning* dapat melakukan ekstraksi fitur, pengenalan pola dan klasifikasi. Gambar 2.1 merupakan ilustrasi perbedaan *machine learning* dan *deep learning*.



Gambar 2.1 Perbedaan *Machine Learning* dan *Deep Learning*
(Khudeyer & Al-Moosawi, 2022)

Deep Learning, bagian mendalam dari sebuah teknik *Machine Learning* dimana proses pengambilan keputusan dilakukan melalui beberapa sistem *artificial neural network*, Adanya teknik *Deep Learning* ini mampu meningkatkan performa sebuah program AI dengan cepat dan lebih akurat, jika dibandingkan dengan teknik AI atau *Machine Learning* konvensional. Tabel 2.1 menyajikan berbagai teknik algoritma *Deep Learning* beserta contoh aplikasinya.

Tabel 2.1 Algoritma *Deep Learning*

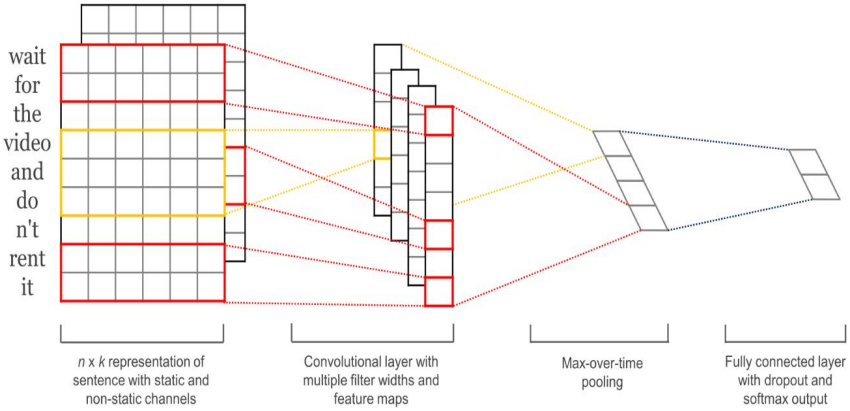
	Nama Teknik DL	**Aplikasi**
Supervised	*Artificial Neural Networks*	Regresi dan klasifikasi
	Convolutional Neural Networks	*Computer Vision,* ekstraksi fitur
	Recurrent Neural Networks	Analisis data kontinu

	Nama Teknik DL	Aplikasi
Unsupervised	*Self-Organizing Maps*	*Feature detection*
	Deep Boltzman Machines	Rekomendasi sistem
	Auto Encoders	Rekomendasi sistem

Penelitian ini menerapkan model *hybrid learning* yang akan menggabungkan dua model yaitu *Convolutional Neural Networks* dan *Bidirectional Long Short-Term Memory* (BiLSTM) yang merupakan pengembangan dari *Long Short-Term Memory* (LSTM).

2.1.3 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan jenis *Deep Learning* yang memanfaatkan *convolutional layer* sebagai penyusun *neural network* yang dibangun (Widhiyasana et al., 2021). *Convolutional Neural Network* (CNN), yang awalnya dikembangkan untuk pemrosesan citra, telah menunjukkan performa yang sangat baik dalam bidang Natural Language Processing (NLP**),** terutama dalam tugas seperti analisis sentimen dan klasifikasi**.** Dalam bidang *Natural Language Processing* (NLP), ekstraksi fitur dari teks merupakan komponen yang sangat esensial. Kemajuan dalam penelitian mengenai representasi terdistribusi, yang dikenal sebagai *word embedding*, telah memungkinkan inisialisasi vektor representasi kata menjadi lebih efisien melalui pemanfaatan perangkat terbuka seperti *Word2Vec* dan *GloVe*. Secara konseptual, proses *embedding* mengubah himpunan kosakata menjadi representasi vektorial dalam ruang berdimensi rendah, di mana jarak antarvektor mencerminkan kedekatan semantik antar kata. Meskipun demikian, beberapa studi mengindikasikan bahwa pendekatan berbasis kata tidak selalu optimal untuk data yang berasal dari media sosial, seperti Twitter, yang ditandai dengan penggunaan token yang bersifat kreatif dan tidak konvensional (Lu et al., 2020).



Gambar 2.2 *Convolutional Neural Networks* untuk klasifikasi (Kim, 2014)

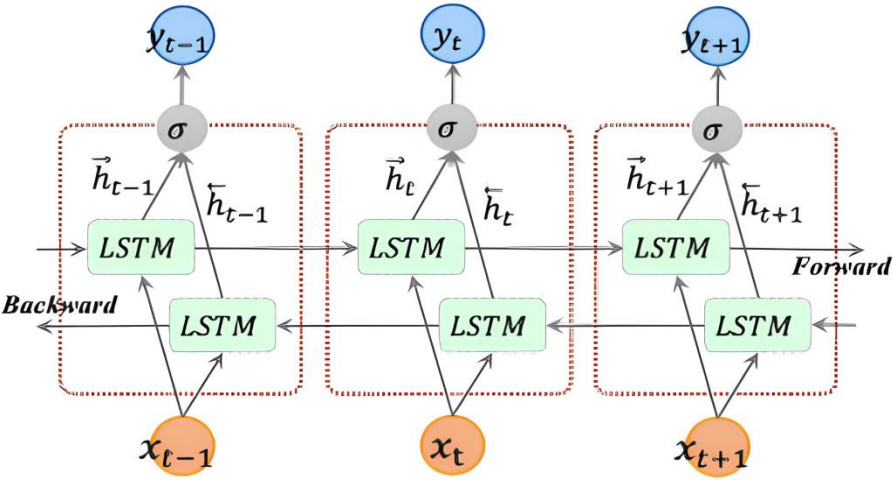
Gambar 2.2 menggambarkan arsitektur CNN yang digunakan untuk tugas klasifikasi teks. Kalimat yang menjadi input diubah terlebih dahulu menjadi bentuk matriks berukuran $n \times k$, di mana setiap barisnya merupakan representasi vektor dari satu kata dalam kalimat tersebut. Setelah itu, dilakukan proses konvolusi menggunakan berbagai ukuran filter untuk menangkap pola-pola penting seperti kombinasi kata *(n-gram*). Hasil konvolusi ini kemudian diringkas melalui proses *max-over-time pooling*, yaitu dengan mengambil nilai tertinggi dari setiap hasil filter untuk mewakili informasi paling kuat. Tahapan terakhir, hasil *pooling* ini diteruskan ke lapisan *fully connected* yang dilengkapi dengan teknik *dropout* untuk mengurangi *overfitting*, lalu diproses oleh fungsi *softmax* guna menentukan kelas dari teks yang dianalisis.

2.1.4 Bidirectional Long Short-Term Memory (Bi-LSTM)

Bidirectional Long Short-Term Memory (Bi-LSTM) adalah perkembangan dari model *Long Short-Term Memory* (LSTM), di dalam BiLSTM terdapat dua lapisan yang prosesnya saling berkebalikan arah. Dengan adanya lapisan dua arah yang saling berlawanan ini, maka model dapat memahami dan mengambil perspektif dari informasi terdahulu maupun informasi terdepan. Hal ini akan memungkinkan proses pembelajaran menjadi semakin dalam (Graves et al., 2005).

Bidirectional Long Short-Term Memory (Bi-LSTM) digunakan untuk mempelajari representasi matriks dari dokumen dan membangun model analisis. *Long Short-Term Memory* **(**LSTM) sendiri dikembangkan

sebagai pengembangan dari *recurrent neural network* (RNN) standar untuk mengatasi masalah *vanishing gradient*. Pada Bi-LSTM, masukan diproses dalam dua arah, yaitu maju (*forward*) dan mundur (*backward*) (Purwarianti & Crisdayanti, 2019).



Gambar 2.3 Arsitektur *hidden* Bi-LSTM (Ira et al., 2021)

Pada Gambar 2.2 dapat dilihat dari setiap *hidden layer* keluaran unit pada *layer* bawah dan atas disatukan sehingga membentuk nilai fitur yang lebih panjang dari pada LSTM biasa. Karena nilai fitur pada Bi-LSTM lebih panjang, maka informasi yang diproses pada selanjutnya yaitu *feed forward neural* akan mengklasifikasikan dengan lebih detail (Ira et al., 2021)**.**

Keterangan:

x_t: input pada waktu ke-**t**

LSTM : unit Long Short-Term Memory yang memproses data sekuensial

$h_t^{\rightarrow}$: hidden state dari LSTM arah *forward* pada waktu ke-t

$h_t^{\leftarrow}$: hidden state dari LSTM arah *backward* pada waktu ke-t

y_t: output model pada waktu ke-t

σ : fungsi aktivasi atau gabungan output dari dua arah

$Forward$: arah proses dari kiri ke kanan (dari masa lalu ke masa depan)

$Backward$: arah proses dari kanan ke kiri (dari masa depan ke masa lalu)

2.1.5 Confusion Matrix

Confusion Matrix digunakan untuk menghitung kinerja atau tingkat kebenaran akurasi proses klasifikasi. Keakuratan hasil diukur dengan nilai *recall, precission, accuracy*. Di mana *recall* (*True, Positive, Rate*) adalah rasio identifikasi benar positif dibandingkan keseluruhan data yang benar positif. *Precission* (*Positive Predictive Value*) adalah rasio identifikasi benar positif terhadap semua hasil identifikasi positif, dan akurasi adalah rasio identifikasi benar positif untuk semua data (Pratiwi et al., 2021). Tabel 2.2 berikut menunjukkan struktur umum dari *Confusion Matrix*:

Tabel 2.2 *Confusion Matrix*

		Kelas prediksi	
		Positif	Negatif
Aktual	Positif	TP	FP
	Negatif	FN	FP

Keterangan:

TP (*True Positive)* : data positif yang diprediksi benar
FN *(False Negative)* : data negatif yang diprediksi benar
FP *(False Positive)* : data negatif namun diprediksi sebagai data positif
FN *(False Negative)* : data positif namun diprediksi data negatif

Dengan kata lain, nilai akurasi merupakan perbandingan antara data yang terklasifikasi benar dengan keseluruhan data. Nilai akurasi dapat diperoleh dengan persamaan:

$$Akurasi = \frac{(TP + TN)}{(TP + TN + FP + FN)} \times 100\%$$

Nilai presisi menggambarkan jumlah data kategori positif yang diklasifikasi secara benar dibagi dengan total data yang diklasifikasi positif, Presisi dapat diperoleh dengan persamaan:

$$Presisi = \frac{(TP)}{(TP + FP)} \times 100\%$$

Sementara itu, *recall* menunjukkan beberapa persen data kategori positif yang terklasifikasi dengan benar oleh sistem.

$$Recall = \frac{(TP)}{(TP + FN)} \times 100\%$$

Error adalah kasus yang diidentifikasi salah dalam sejumlah data, sehingga dapat dilihat seberapa besar tingkat kesalahan pada sistem yang digunakan. Persentase error dapat dilakukan perhitungan menggunakan persamaan 4 di bawah ini:

$$Error = \frac{FP}{TP} x\, 100\%$$

2.1.6 Word *Embedding*

Word embedding merupakan pendekatan representasi kata dalam bentuk vektor berdimensi tetap yang memungkinkan pemodelan makna kata berdasarkan konteks penggunaannya dalam suatu korpus teks (Jurafsky, Daniel S. and Martin, 2025). Teknik ini muncul sebagai solusi terhadap keterbatasan representasi tradisional seperti *one-hot encoding*, yang memperlakukan kata sebagai entitas diskrit dan tidak menyimpan informasi hubungan semantik antar kata (Jurafsky, Daniel S. and Martin, 2025)

Konsep *Word embedding* didasarkan pada hipotesis distribusional (*distributional hypothesis*), yaitu gagasan bahwa kata-kata yang memiliki makna serupa akan muncul dalam lingkungan atau konteks yang serupa (Jurafsky, Daniel S. and Martin, 2025). Oleh karena itu, *embedding* memungkinkan pembelajaran makna kata secara tidak langsung melalui pola distribusi kata dalam kalimat. Misalnya, kata “dokter” dan “perawat” cenderung muncul dalam konteks yang berkaitan dengan rumah sakit atau pasien, sehingga keduanya akan memiliki representasi vektor yang berdekatan dalam ruang vektor.

Dalam praktiknya, representasi vektor ini bersifat padat (*dense*) dan kontinu, serta memungkinkan pengukuran kedekatan semantik antar kata menggunakan metode seperti jarak kosinus atau operasi vektor linier. Hal ini tidak dimungkinkan pada pendekatan one-hot, di mana semua kata dianggap sepenuhnya berbeda dan tidak memiliki hubungan matematis yang bermakna (Jurafsky, Daniel S. and Martin, 2021).

Saat ini, *Word embedding* telah menjadi komponen penting dalam berbagai aplikasi pemrosesan bahasa alami (*Natural Language Processing*), seperti klasifikasi teks, analisis sentimen, sistem tanya jawab otomatis, dan penerjemahan mesin. Representasi kata yang lebih bermakna dan efisien menjadikan *embedding* sebagai fondasi dalam pengembangan berbagai model lanjutan dalam NLP, termasuk model berbasis *Deep Learning*.

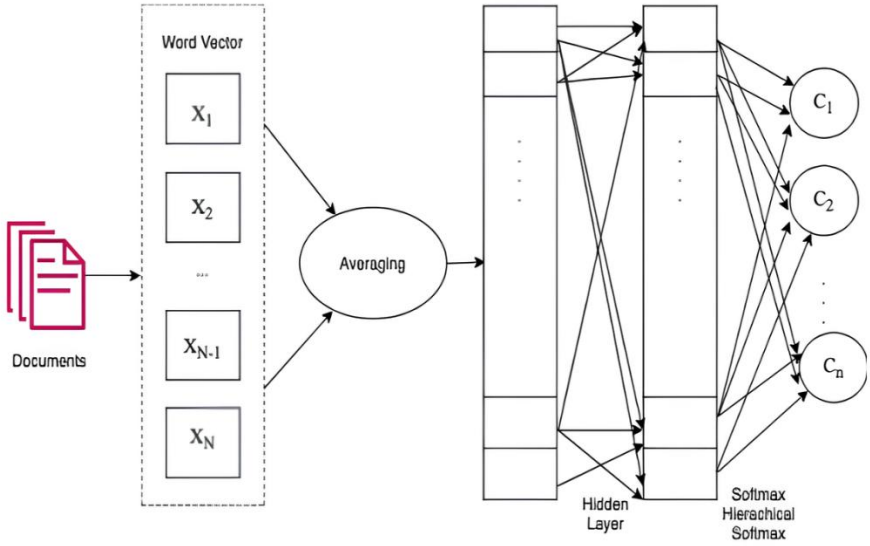
Word embedding merupakan istilah yang digunakan untuk representasi dari kata-kata yang terdistribusi. Teknik ini sering digunakan pada penggalian teks atau *Natural Language Processing* (Mikolov et al., 2013). Cara kerja dari teknik ini adalah dengan merepresentasikan kumpulan kata menjadi suatu vektor. Sehingga dari representasi tersebut dapat diolah lebih lanjut untuk mengetahui konteks dari suatu kalimat atau untuk memperoleh informasi mengenai kesamaan semantik dan sintaksis serta keterkaitannya dengan kata lain.

2.1.7 FastText

FastText dikembangkan oleh tim Riset AI Facebook yang digunakan untuk mempelajari representasi kalimat dan kata secara efisien (Bojanowski et al., 2017) . FastText merupakan pengembangan dari pendekatan *Skip-gram*. Tidak seperti Word2Vec yang menggunakan representasi tingkat kata yang memperlakukan setiap kata sebagai unit terkecil, FastText mempertimbangkan bagian-bagian penyusun suatu kata dengan cara membagi kata tersebut menjadi potongan-potongan kecil yang disebut *n-gram* karakter. Misalnya, sebuah kata dapat dipecah menjadi kombinasi huruf-huruf pendek seperti awalan, suku kata, atau akhiran. Setiap potongan ini kemudian diubah menjadi vektor angka. Selanjutnya, semua vektor dari potongan-potongan tersebut dijumlahkan untuk membentuk satu vektor yang mewakili keseluruhan kata.

Oleh karena itu, unit terkecil bukanlah kata tetapi karakter. Pendekatan ini memungkinkan FastText menghasilkan representasi yang lebih baik untuk kata-kata yang jarang muncul, atau bahkan belum pernah muncul dalam data pelatihan (*out-of-vocabulary*) (Bojanowski et al., 2017).

Penelitian yang dilakukan oleh Darma dalam (Setyaningrum & Nadhif, 2025) melakukan evaluasi kinerja terhadap tiga *word embedding* (*Word2Vec, GloVe,* dan *FastText*), menemukan bahwa kinerja *FastText* lebih unggul, mencapai hasil akurasi sebesar 97,2%. Berdasarkan hasil tersebut, pada penelitian ini menggunakan *Word embedding FastText.* Arsitektur *FastText* ditunjukkan pada Gambar 2.4.



Gambar 2.4 Arsitektur *FastText* (Setyaningrum & Nadhif, 2025)

Gambar 2.4 menggambarkan arsitektur *FastText* dalam proses klasifikasi teks. Proses dimulai dari dokumen yang berisi urutan kata. Setiap kata dalam dokumen diubah menjadi vektor kata melalui proses *embedding*. Dalam gambar, setiap vektor tersebut dilambangkan sebagai X_1, X_2, ..., X_n**,** yang masing-masing merepresentasikan vektor dari kata ke-1 hingga kata ke-n dalam satu dokumen.

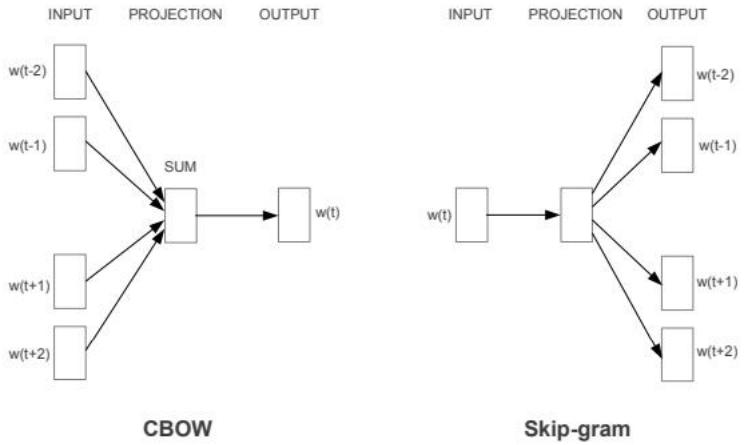
Vektor-vektor ini kemudian dirata-ratakan (averaging) untuk menghasilkan satu vektor representasi yang mewakili keseluruhan dokumen. Representasi ini kemudian diteruskan ke lapisan tersembunyi (*hidden layer*), di mana dilakukan pemrosesan lanjutan oleh jaringan neural. Output dari *hidden layer* masuk ke lapisan klasifikasi**,** yang menggunakan fungsi aktivasi *Softmax* atau *Hierarchical Softmax* untuk menghasilkan probabilitas kelas. Hasil akhirnya, model mengklasifikasikan dokumen ke

salah satu dari kelas keluaran yaitu $(C1, C2, \ldots, Cn)$ berdasarkan hasil prediksi tertinggi.

2.1.8 Word2Vec

Word2Vec merupakan salah satu teknik representasi kata dalam bentuk vektor yang diperkenalkan oleh Mikolov et al. (2013) dari Google. Berbeda dengan pendekatan tradisional yang merepresentasikan kata sebagai indeks diskrit tanpa memperhatikan hubungan antar kata, *Word2Vec* mampu membentuk representasi kata ke dalam vektor kontinu berdimensi tetap berdasarkan konteks penggunaannya dalam kalimat. Pendekatan ini menghasilkan vektor kata yang secara semantik bermakna, di mana kata-kata yang memiliki makna atau fungsi serupa akan memiliki representasi vektor yang berdekatan dalam ruang vektornya.

Word2Vec memiliki dua arsitektur utama, yaitu *Continuous Bag-of-Words* (CBOW) dan Skip-gram. CBOW memprediksi kata target berdasarkan kata-kata konteks di sekitarnya, sedangkan Skip-gram melakukan sebaliknya, yaitu memprediksi kata-kata konteks dari kata target. CBOW cenderung lebih cepat dan efisien untuk dataset besar, sementara Skip-gram lebih unggul dalam menangkap hubungan semantik antara kata-kata yang jarang muncul (Mikolov et al., 2013).



Gambar 2.5 Arsitektur model CBOW dan Skip-gram (Mikolov et al., 2013)

Gambar 2.5 menunjukkan arsitektur model CBOW dan Skip-gram. Arsitektur CBOW bekerja dengan cara memprediksi kata target berdasarkan kata-kata konteks di sekitarnya. Dalam prosesnya, kata-kata

seperti $w(t-2), w(t-1), w(t+1)$, dan $w(t+2)$ yang mengelilingi kata target $w(t)$ dijadikan sebagai input. Masing-masing kata tersebut diubah ke dalam bentuk vektor dan dijumlahkan dalam *projection layer*, lalu vektor hasil penjumlahan digunakan untuk memprediksi kata target $w(t)$ pada *output layer*. Dengan kata lain, CBOW berusaha menemukan kata tengah dari kumpulan kata konteks yang diberikan, dan pendekatan ini dinilai lebih efisien secara komputasi serta cocok untuk data dalam skala besar. Sebaliknya, arsitektur Skip-gram menggunakan satu kata target $w(t)$ sebagai input dan memprediksi kata-kata konteks di sekitarnya, seperti $w(t-2), w(t-1), w(t+1)$, dan $w(t+2)$ sebagai output. Kata target tersebut diproyeksikan ke dalam ruang vektor, kemudian digunakan untuk menghasilkan prediksi terhadap konteks. Skip-gram dinilai lebih efektif dalam menangani kata-kata yang jarang muncul dalam data karena mampu memperkuat relasi antara satu kata dengan berbagai konteks yang mungkin terjadi di sekitarnya(Mikolov et al., 2013).

Keterangan :

$w(t-2), w(t-1), w(t+1)$, $w(t+2)$: kata-kata konteks di sekitar kata target.

$w(t)$: kata target (kata yang diprediksi dalam CBOW atau menjadi input pada Skip-gram)

2.1.9 GloVe

GloVe (*Global Vectors for Word Representation*) merupakan salah satu metode representasi kata berbasis vektor yang dikembangkan oleh Pennington, Socher, dan Manning pada tahun 2014 (Pennington et al., 2014) . Metode ini bertujuan untuk menghasilkan representasi kata dalam bentuk vektor angka yang mampu menangkap makna kata secara semantik. *GloVe* dikembangkan dengan menggabungkan dua pendekatan utama dalam pembelajaran representasi kata, yaitu pendekatan global dan pendekatan lokal. Pendekatan global, seperti La*tent Semantic Analysis* (LSA), menggunakan statistik keseluruhan dari korpus untuk membangun representasi kata, sedangkan pendekatan lokal, seperti *skip-gram* dalam *Word2Vec*, belajar dari konteks kata dalam jendela kalimat yang terbatas. Dengan menggabungkan kedua pendekatan ini, *GloVe* mampu

memanfaatkan kelebihan keduanya secara bersamaan (Pennington et al., 2014).

GloVe bekerja dengan memanfaatkan informasi statistik berupa frekuensi kemunculan bersama (*co-occurrence*) antara kata-kata dalam korpus teks. Inti dari metode ini adalah bahwa makna suatu kata dapat diketahui dengan melihat seberapa sering kata tersebut muncul bersama dengan kata-kata lainnya. Misalnya, kata "es" (*ice*) dan "uap" (*steam*) sering muncul bersama kata "air" (water), namun "es" lebih sering muncul dengan "padat" (solid), sementara "uap" lebih sering muncul dengan "gas". *GloVe* memanfaatkan rasio probabilitas kemunculan bersama ini untuk membedakan makna kata-kata secara lebih akurat. Dengan demikian, *GloVe* tidak hanya memperhatikan frekuensi, tetapi juga pola hubungan antar kata (Pennington et al., 2014).

Secara matematis, *GloVe* memodelkan hubungan antara kata dan konteks menggunakan pendekatan regresi *log-bilinear* berbobot. Tujuannya adalah untuk meminimalkan selisih antara hasil proyeksi vektor kata dan konteks terhadap logaritma dari frekuensi kemunculan bersama. Fungsi objektif utama dari model ini dinyatakan dengan rumus berikut (Pennington et al., 2014):

$$J = \sum_{\{i,j=1\}}^{V} f(X_{ij})\left(w_i{}^T \tilde{w}_j + b_i + \tilde{b}_j - \log\left(x_{\{ij\}}\right)\right)^2$$

Keterangan:

- X_{ij} adalah jumlah kemunculan bersama antara kata ke-iii dan kata ke-j,
- w_i dan $\tilde{w}_j$ adalah vektor representasi untuk kata utama dan kata konteks,
- b_i dan $\tilde{b}_j$ adalah bias,
- $f(X_{ij})$ adalah fungsi pembobot untuk mengurangi pengaruh nilai ekstrem (sangat jarang atau sangat sering).

2.1.10 Python

Python merupakan bahasa pemrograman tingkat tinggi yang yang dibuat oleh Guido Van Rossum dan dirilis pada tahun 1991 (Python Software Foundation, 2023). *Python* juga merupakan bahasa yang sangat populer belakangan ini. Selain itu *Python* juga merupakan bahasa

pemrograman yang multi fungsi contohnya *Python* dapat digunakan untuk *Machine Learning* dan *Deep Learning*. *Python* dipilih sebagai penelitian karena *Python* memiliki penulisan sintaksis yang mudah selain itu *Python* juga memiliki *library* yang lengkap dan memiliki dukungan komunitas yang kuat karena *Python* bersifat *open source*. Untuk menuliskan *source code Python* dapat menggunakan IDE seperti *vs code, sublime text, PyCharm* atau juga dapat menggunakan IDE *online* seperti *Jupyter notebook* dan *google colab*.

Python dirancang untuk memberikan kemudahan yang sangat luar biasa kepada programer baik dari segi efisiensi waktu, maupun kemudahan dalam pengembangan program dan dalam hal kompatibilitas dengan sistem. *Python* bisa digunakan untuk membuat program *standalone* dan pemrograman skrip (*Scripting Programming*). Pada saat ini *Python* telah mencapai versi *Python* 3.12.5 yang dirilis pada 06 Agustus 2024. Beberapa fitur yang dimiliki bahasa pemrograman *Python* adalah sebagai berikut:

1. Memiliki *library* yang luas, dalam distibusi *Python* telah disediakan modul-modul siap pakai untuk berbagai keperluan.
2. Memiliki tata bahasa yang jernih dan mudah dipelajari.
3. Memiliki aturan layout *source code* yang memudahkan pengecekan, pembacaan kembali, dan penulisan ulang *source code* tersebut.
4. Memiliki modular, mudah dikembangkan dengan menciptakan modul-modul baru, dimana modul-modul tersebut dapat dibangun dengan bahasa *Python* maupun C/C++.

Memiliki fasilitas pengumpulan sampah secara auto, seperti halnya pada bahasa pemrograman Java, *Python* memiliki fasilitas pengaturan penggunaan memori komputer sehingga para *programmer* tidak perlu melakukan pengaturan memori komputer secara langsung.

2.1.11 Tensorflow

TensorFlow adalah sebuah kerangka kerja (*framework*) *open-source* yang dikembangkan oleh Google Brain untuk mendukung pemodelan dan pelatihan sistem pembelajaran mesin berskala besar. *Framework* ini dirancang dengan menggunakan pendekatan *graf* komputasi berbasis *dataflow*, di mana setiap operasi matematika direpresentasikan sebagai node, dan data mengalir melalui jalur penghubung (*edges*) dalam bentuk tensor (Abadi et al., 2016) . Pendekatan ini memungkinkan *TensorFlow* untuk menjalankan perhitungan secara efisien pada berbagai perangkat keras, termasuk CPU, GPU, maupun TPU.

Dalam konteks pengembangan model *Deep Learning*, *TensorFlow* menyediakan antarmuka tingkat tinggi (*tf.keras*) yang memudahkan pembuatan arsitektur jaringan saraf seperti *Convolutional Neural Network* (CNN) dan *Bidirectional Long Short-Term Memory* (BiLSTM). CNN biasanya digunakan untuk mengekstraksi fitur spasial dari data seperti gambar atau representasi *embedding* teks, melalui operasi konvolusi dan *pooling*. Sementara itu, BiLSTM merupakan varian dari *Recurrent Neural Network* (RNN) yang dapat menangkap konteks urutan dari dua arah (maju dan mundur), menjadikannya efektif untuk pemrosesan data sekuensial seperti klasifikasi teks (Ramsundar & Zadeh, 2018). (Hope et al., 2017)

2.1.12 Gensim

Gensim merupakan pustaka *open-source* berbasis Python yang dirancang untuk menangani teks dalam skala besar melalui teknik representasi vektor dan pemodelan topik. Sejak diperkenalkan oleh Rehůřek dan Sojka (2010), *Gensim* telah berkembang menjadi salah satu *library* utama dalam pemrosesan bahasa alami (*Natural Language Processing/NLP*), khususnya dalam penerapan *Word embedding* seperti *Word2Vec*, *FastText*, dan Doc2Vec (Kuo, 2023).

Library ini dibuat untuk dapat mengambil topik semantik dari sebuah data secara otomatis. Selain untuk pemodelan topik, toolkit yang bersifat *open source* ini juga dapat digunakan sebagai pemodelan ruang vektor dan dapat diimplementasikan menggunakan *NumPy, SciPy,* dan *Cython* secara opsional untuk meningkatkan kinerja dari *Gensim*. Cara kerja *Gensim* adalah dengan menggunakan algoritma *TextRank* yang sangat sering digunakan, cara kerja ini didasarkan oleh kalimat pada teks yang diberi peringkat menggunakan algoritma *TextRank*. *TextRank* merupakan sebuah algoritma berbasis grafik untuk natural language processing (NLP). Algoritma ini berfungsi untuk membuat ringkasan dari suatu dokumen menjadi peringkat berbasis grafik (Hidayat et al., 2024).

Menurt (Ciaburro & Joshi, 2019) salah satu kekuatan utama *Gensim* adalah kemampuannya menghasilkan representasi vektor kata yang kemudian dapat digunakan sebagai input untuk model klasifikasi teks berbasis *Deep Learning*, seperti *Convolutional Neural Network* (CNN) dan *Bidirectional Long Short-Term Memory* (BiLSTM). *Embedding* hasil pelatihan dari *Gensim* misalnya model *FastText* atau *Word2Vec* dapat diekstrak dan diintegrasikan ke dalam lapisan *embedding* di *framework*

TensorFlow atau *Keras*. Ciaburro dan Joshi (2020) juga menegaskan bahwa integrasi *embedding* dari *Gensim* ke dalam model *Deep Learning* meningkatkan akurasi klasifikasi, terutama pada data teks yang kompleks atau domain khusus.

2.2 Penelitian Terkait

Penelitian ini merupakan studi literatur yang merujuk pada beberapa penelitian terdahulu di antaranya adalah sebagai berikut:

Berdasarkan penelitian yang dilakukan oleh Song & Sun (2020) yang berjudul *"Longer Text Classification Based on BiLSTM-CNN Model"*. Penelitian ini menggunakan *Word2Vec* (Skip-gram) untuk mengubah kata menjadi vektor berdimensi tetap, lalu menerapkan BiLSTM untuk menangkap informasi semantik dua arah, dan CNN untuk ekstraksi fitur lokal. Hasil eksperimen menunjukkan bahwa model BiLSTM-CNN menghasilkan akurasi, *recall*, dan F1-score yang lebih tinggi dibandingkan model CNN dan BiLSTM secara individu, dengan peningkatan performa hingga lebih dari 1% pada beberapa metrik evaluasi, khususnya pada dataset SogouCS. Hal ini membuktikan bahwa penggabungan BiLSTM dan CNN efektif dalam menangani klasifikasi teks panjang yang kompleks.

Penelitian selanjutnya oleh Rahnoui, Mikram, Yousfi, & Barzali (2020) yang berjudul *"A CNN-BiLSTM Model for Document-Level Sentiment Analysis"* mengusulkan kombinasi model CNN dan BiLSTM untuk menganalisis sentimen pada level dokumen panjang, menggunakan *embedding Doc2vec*. CNN digunakan untuk mengekstraksi fitur lokal dari teks, sementara BiLSTM menangkap dependensi semantik dua arah dalam dokumen. Hasil eksperimen pada dataset berisi 2003 artikel koran berbahasa Prancis menunjukkan bahwa model CNN-BiLSTM dengan *Doc2vec* menghasilkan akurasi tertinggi sebesar 90.66%**,** mengungguli model-model lain seperti CNN (88.00%), LSTM (85.87%), BiLSTM (86.40%), dan CNN-LSTM (90.13%). Hasil ini menunjukkan bahwa kombinasi CNN dan BiLSTM, ditambah *embedding* dokumen yang kuat, mampu meningkatkan kinerja klasifikasi sentimen pada teks panjang secara signifikan.

Selain itu, penelitian yang oleh Setyaningrum & Nadhif (2025) dengan judul "Klasifikasi *Cyberbullying* Pada Tweet Bahasa Sunda Dengan Menggunakan *Hybrid Learning* Model" menunjukkan bahwa pendekatan *hybrid* mampu meningkatkan akurasi dalam mendeteksi ujaran *cyberbullying* pada teks berbahasa Sunda. menggabungkan teknik

preprocessing seperti *stemming* dan *FastText word embedding*, serta mengimplementasikan model *Machine Learning (Random Forest, SVM)* dan *Deep Learning* (CNN, BiLSTM, dan CNN-BiLSTM). Hasilnya, model CNN-BiLSTM menunjukkan performa terbaik dengan akurasi sebesar 97,3% dan F1-score sebesar 97%, mengungguli model-model lainnya. Selain itu, waktu pelatihan model *hybrid* ini tergolong cepat, yakni sekitar 30 detik per epoch, yang menandakan efisiensi komputasi yang baik.

Penelitian yang dilakukan oleh Aqila, Sibaroni, & Prasetiyowati (2023) dengan judul "*Word2Vec Architecture in Sentiment Classification of Fuel Price Increase Using CNN-BiLSTM Method*" menunjukkan bahwa pendekatan *hybrid Deep Learning* dengan kombinasi CNN dan BiLSTM mampu mengklasifikasikan sentimen masyarakat terhadap kenaikan harga BBM secara efektif. Penelitian ini menggunakan dua arsitektur *Word2Vec*, yaitu *CBOW* dan *Skip-gram*, dengan berbagai ukuran dimensi vektor. Hasil terbaik diperoleh dari arsitektur *CBOW* dengan dimensi vektor 300, yang menghasilkan akurasi sebesar 87%, *recall* 87%, *precision* 89%, dan F1-score 88%. Kombinasi CNN dan BiLSTM terbukti efektif dalam mengekstraksi fitur dan mempertahankan konteks kalimat, sementara penggunaan *Word2Vec* memungkinkan model memahami hubungan semantik antar kata dalam dataset berbahasa Indonesia.

Terakhir, penelitian oleh Muslikh & Ismail (2024) berjudul *"Multi-label Classification of Indonesian Al-Quran Translation based CNN, BiLSTM, and FastText"* mengkaji klasifikasi multilabel ayat Al-Qur'an terjemahan Indonesia dengan model CNN + BiLSTM dan *FastText*. Data diambil dari tiga surah (An-Nisa', Al-Maidah, Al-An'am) sebanyak 461 ayat, yang dilabeli ke dalam empat kategori: Tauhid, Ibadah, Akhlak, dan Sejarah**.** Hasilnya, model dengan *FastText* menunjukkan peningkatan performa signifikan, mencapai F1-Score tertinggi 73,30% (*embedding* size 200, epoch 100), dibandingkan model tanpa *FastText* yang hanya mencapai 68,70%**.**

Tabel 2.3 Penelitian terkait

Penelitian 1		
Judul	:	*Longer Text Classification Based on BiLSTM-CNN Model* (Song & Sun, 2021)
Metode	:	*Word2Vec* (Skip-gram), BiLSTM-CNN
Hasil	:	Peningkatan >1% akurasi, *recall*, F1 dibanding model individual CNN/BiLSTM

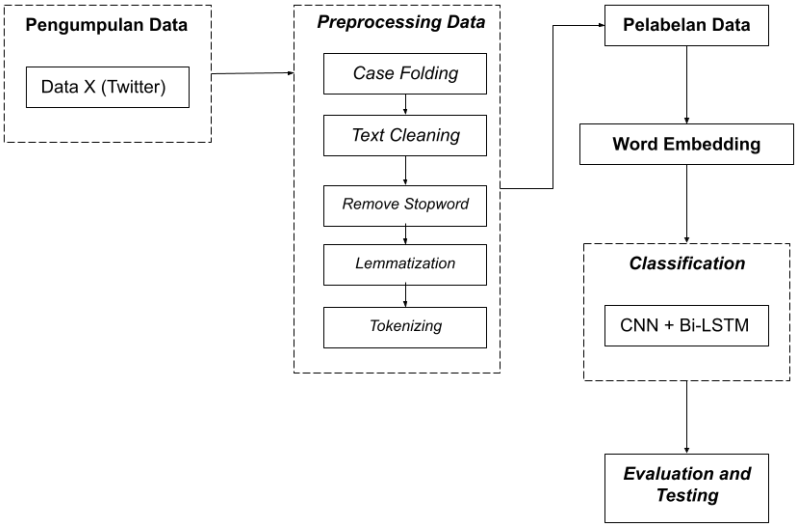
Relevansi	:	Kombinasi BiLSTM dan CNN efektif untuk klasifikasi teks panjang dan kompleks secara dua arah.
Penelitian 2		
Judul	:	*A CNN-BiLSTM Model for Document-Level Sentiment Analysis* (Rhanoui et al., 2019)
Metode	:	Doc2Vec, CNN-BiLSTM
Hasil	:	Akurasi CNN-BiLSTM: 90.66% – lebih tinggi dari model lain seperti CNN, LSTM, BiLSTM
Relevansi	:	CNN dan BiLSTM secara gabungan mampu menganalisis dependensi semantik dua arah secara mendalam pada teks panjang.
Penelitian 3		
Judul	:	Klasifikasi *Cyberbullying* Pada Tweet Bahasa Sunda Dengan Menggunakan Hybrid Learning Model (Setyaningrum & Nadhif, 2025)
Metode	:	*FastText*, CNN-BiLSTM, RF, SVM
Hasil	:	Akurasi CNN-BiLSTM: 97.3%, F1-score: 97%, waktu latih 30 detik/epoch
Relevansi	:	Membuktikan efektivitas CNN-BiLSTM dalam klasifikasi ujaran kebencian lokal berbasis bahasa daerah (*cyberbullying*).
Penelitian 4		
Judul	:	*Word2Vec Architecture in Sentiment Classification of Fuel Price Increase Using CNN-BiLSTM Method* (Aqilla et al., 2023)
Metode	:	*Word2Vec* (CBOW & Skip-gram), CNN-BiLSTM
Hasil	:	CBOW (300 dimensi): akurasi 87%, *recall* 87%, *precision* 89%, F1-score 88%
Relevansi	:	Membuktikan kekuatan CNN-BiLSTM dalam memahami hubungan semantik kata dalam teks Bahasa Indonesia.
Penelitian 5		
Judul	:	Multi-label Classification of Indonesian Al-Quran Translation based CNN, BiLSTM, and *FastText*

		(Muslikh et al., 2024)
Metode	:	CNN + BiLSTM, *FastText*
Hasil	:	F1-score 73.3% dengan *FastText* (*embedding* size 200), lebih baik dari model tanpa *FastText* (68.7%).
Relevansi	:	Relevan untuk menunjukkan adaptabilitas CNN-BiLSTM dalam konteks teks bahasa Indonesia dan multilabel classification.

BAB 3
DESAIN SISTEM

3.1 Tahapan Penelitian

Tahapan penelitian ini disusun secara sistematis untuk menggambarkan alur kerja yang dilakukan mulai dari proses pengumpulan data hingga evaluasi model. Rangkaian tahapan tersebut dapat dilihat pada Gambar 3.1 berikut.



Gambar 3.1 Tahapan penelitian

Tahapan penelitian digambarkan pada Gambar 3.1, yaitu alur proses klasifikasi komentar *cyberbullying* pada media sosial, dimulai dari tahap pengumpulan data hingga evaluasi. Pada tahap pengumpulan data, data diambil dari sosial media X (Twitter), kemudian data mentah tersebut dibersihkan lalu masuk ke tahap pelabelan data, data diberikan label klasifikasi *cyberbullying* dan *non-cyberbullying*. Selanjutnya, pada tahap *Preprocessing data*, data teks melalui proses *preprocessing* yang mencakup *case folding*, *text cleaning, remove stopword, lemmatization,* dan *tokenizing*. Setelah proses ini, representasi kata diubah ke dalam bentuk vektor menggunakan metode *embedding* berbasis *FastText***,** yang mampu menangkap makna kata secara kontekstual termasuk struktur sub-kata. Data

hasil *embedding* kemudian diproses menggunakan model *deep learning hybrid* CNN-BiLSTM, di mana CNN digunakan untuk mengekstraksi fitur lokal dari urutan kata, dan BiLSTM digunakan untuk memahami konteks kalimat dari dua arah (*forward* dan *backward*). Model ini dievaluasi menggunakan metrik performa seperti akurasi, *precision, recall*, dan *F1-score*. Pada tahap Output**,** sistem menghasilkan klasifikasi untuk setiap komentar yang dianalisis dan menampilkannya dalam bentuk visualisasi seperti grafik distribusi kelas dan *confusion matrix*.

3.2 Metodologi Penelitian

Bagian ini menjelaskan secara sistematis tahapan-tahapan yang dilakukan dalam penelitian, yang meliputi proses pengumpulan data, pelabelan data, *preprocessing* data, serta pembangunan dan evaluasi model klasifikasi.

3.2.1 Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan melalui metode *crawling* data dari media sosial "X", yang sebelumnya dikenal sebagai Twitter. Proses pengambilan data dilakukan dengan menggunakan alat bantu bernama *Tweet-Harvest*, yang memungkinkan untuk mengakses dan mengunduh tweet publik secara otomatis dengan bantuan token otentikasi dari akun media sosial "X". Tweet yang dikumpulkan difilter hanya mencakup tweet yang menggunakan Bahasa Indonesia.

Selanjutnya, pencarian tweet difokuskan pada sejumlah kata kunci. Kata kunci tersebut mencakup ungkapan positif seperti semangat, sukses, dan cantik, serta kata-kata negatif yang berpotensi mengandung unsur *cyberbullying* seperti tolol dan norak. Pemilihan kata kunci ini bertujuan untuk memperoleh variasi data yang mencerminkan kedua sisi: ujaran yang bersifat positif maupun negatif. Setelah proses *crawling* selesai dilakukan, diperoleh total sebanyak 2.568 tweet. Kumpulan data inilah yang kemudian digunakan pada tahapan selanjutnya. Gambar 3.2 berikut menyajikan contoh potongan kode program yang digunakan dalam proses *crawling* data tweet berdasarkan kata kunci tertentu.

```
# Crawl Data

filename = 'semangat.csv'
search_keyword = 'semangat lang:id'
limit = 50000

!npx -y tweet-harvest@2.6.1 -o "{filename}" -s "{search_keyword}" --tab "LATEST" -l {limit} --token {twitter_auth_token}
```

Gambar 3.2 *Crawling* data

3.2.2 Preprocessing Data

Data yang telah memiliki label kemudian akan memasuki tahap data *preprocessing* data. *Preprocessing* data akan sangat berpengaruh pada hasil pemrosesan karena dapat mengoptimalkan hasil pemrosesan (Candra & Rozana, 2020). Pada tahap ini, dilakukan beberapa langkah *preprocessing* untuk menyiapkan data teks agar lebih rapi dan siap digunakan dalam melatih model.

Urutan dalam tahapan *preprocessing* teks memiliki peran penting dalam menentukan kualitas data yang akan digunakan dalam pemodelan. Menurut Jurafsky dan Martin (2021), proses seperti *normalization, case folding, remove stopword*, dan *tokenizing* harus dilakukan secara berurutan dan hati-hati agar tidak menyebabkan ketidakkonsistenan dalam hasil token atau kehilangan informasi semantik yang penting. Selain itu, Kwartler (2017) menambahkan bahwa urutan langkah *preprocessing* perlu disesuaikan dengan tujuan analisis dan karakteristik data yang digunakan. Dengan demikian, hasil akhir dari pemrosesan teks dapat tetap akurat, representatif, dan bermakna.

3.2.2.1 Case Folding

Pada tahap ini, dilakukan proses *Case Folding*, yaitu mengubah semua huruf dalam teks menjadi huruf kecil (*lowercase*). Tujuan dari *case folding* adalah untuk menyamakan format teks agar tidak terjadi perbedaan akibat penggunaan huruf kapital. Misalnya, kata "Bahagia" dan "bahagia" akan dianggap sama setelah dilakukan *case folding*. Contoh hasil penerapan proses *case folding* dapat dilihat pada Tabel 3.2, yang menunjukkan perbandingan teks sebelum dan sesudah diubah menjadi huruf kecil secara keseluruhan.

Tabel 3.1 Contoh penerapan *case folding*

Sebelum *Case Folding*	Sesudah *Case Folding*
Yuk Semangat kerja @teman! Cek website kami di http://contoh.com #motivasi2025	yuk semangat kerja @teman! cek website kami di http://contoh.com #motivasi2025
Dasar jelek banget! @orang_ngaco, ngaca Dulu deh! #kampungan123 100% bego!	dasar jelek banget! @orang_ngaco, ngaca dulu deh! #kampungan123 100% bego!

```
def case_folding(teks):
    return teks.lower()

data['"full_text"'] = data['"full_text"'].astype(str).apply(case_folding)
```

Gambar 3.3 Contoh kode program *case folding*

Gambar 3.3 menampilkan contoh potongan kode program untuk proses *case folding*, yang dilakukan dengan menggunakan fungsi `lower()` dari Python. Fungsi ini akan mengubah semua huruf pada teks menjadi huruf kecil. Karena fungsi `str.lower()` sudah tersedia secara langsung dalam Python, maka tahap ini tidak memerlukan library tambahan. Dengan demikian, teks yang awalnya berbentuk campuran huruf kapital dan huruf kecil dapat diubah menjadi huruf kecil seluruhnya sehingga lebih konsisten untuk tahap pengolahan selanjutnya.

3.2.2.2 Text Cleaning

Tahap *Text Cleaning* bertujuan untuk menghapus elemen-elemen yang tidak relevan dalam teks, seperti URL, *mention, hashtag*, angka, dan tanda baca. Hal ini dilakukan untuk memastikan data yang akan dianalisis bersih dari karakter yang tidak diperlukan sehingga hasil analisis dapat lebih akurat dan tidak terganggu oleh elemen teks yang tidak signifikan. Contoh hasil dari proses *text cleaning* ditunjukkan pada Tabel 3.3**.**

Tabel 3.2 Contoh penerapan *text cleaning*

Sebelum *Text Cleaning*	Sesudah *Text Cleaning*
yuk semangat kerja @teman! cek website kami di http://contoh.com #motivasi2025	yuk semangat kerja cek website kami di
dasar jelek banget! @orang_ngaco, ngaca dulu deh! #kampungan123 100% bego!	dasar jelek banget ngaca dulu deh

```python
def bersihkan_teks(teks):
    teks = re.sub(r'http\S+|www.\S+', '', teks)  # Hapus URL
    teks = re.sub(r'@\w+', '', teks)             # Hapus mention
    teks = re.sub(r'#', '', teks)                # Hapus hashtag
    teks = re.sub(r'[^A-Za-z\s]', '', teks)      # Hapus simbol
    teks = teks.strip()                          # Hilangkan spasi berlebih
    return teks

data['"full_text"'] = data['"full_text"'].apply(bersihkan_teks)
```

Gambar 3.4 Contoh kode program *text cleaning*

Gambar 3.4 menunjukkan contoh potongan kode program yang digunakan dalam proses *text cleaning*. Pada kode tersebut, digunakan beberapa *library* Python, yaitu re dan string. *Library* `re` digunakan untuk menangani pola teks menggunakan ekspresi reguler (*regex*). Fungsi `re.sub()` digunakan untuk menghapus URL, *mention, hashtag,* angka, dan karakter lain yang tidak diperlukan. Sementara itu, *library* `string` digunakan untuk menghapus tanda baca melalui fungsi `translate()` dan `str.maketrans()` sehingga teks menjadi lebih bersih dan terstruktur. Setelah semua elemen tidak relevan dihapus, dilakukan juga penghapusan spasi berlebih menggunakan fungsi `re.sub()` agar teks lebih rapi.

3.2.2.3 Remove Stopword

Pada tahap ini, dilakukan proses *Remove Stopword*, yaitu menghapus kata-kata yang dianggap tidak memiliki informasi penting dalam konteks analisis teks, seperti kata "dan", "yang", "dengan", dan sebagainya. *Stopword* merupakan kata-kata umum yang sering muncul tetapi tidak memberikan kontribusi bermakna terhadap analisis atau pemodelan. Contoh penerapan proses *remove stopword* dapat dilihat pada Tabel 3.4, yang memperlihatkan perbedaan teks sebelum dan sesudah penghapusan kata-kata umum yang tidak memiliki kontribusi makna signifikan terhadap analisis.

Tabel 3.3 Contoh penerapan *remove stopword*

Sebelum *Remove Stopword*	Sesudah *Remove Stopword*
yuk semangat kerja cek website kami di	semangat kerja cek website
dasar jelek banget ngaca dulu deh	jelek ngaca

```
stop_words = set(stopwords.words('indonesian'))

def remove_stopwords(tokens):
    return [word for word in tokens if word not in stop_words]

data['tokens'] = data['tokens'].apply(remove_stopwords)
```

Gambar 3.5 Contoh kode program *remove stopword*

Gambar 3.5 menampilkan potongan kode program untuk proses *remove stopword* menggunakan library NLTK *(Natural Language Toolkit)*. NLTK menyediakan daftar *stopword* dalam berbagai bahasa, termasuk bahasa Indonesia. Proses ini dilakukan dengan mengunduh daftar *stopword* menggunakan `nltk.download('stopwords')` terlebih dahulu. Setelah itu, dilakukan pembandingan setiap kata dalam teks dengan daftar *stopword.* Kata-kata yang termasuk dalam daftar *stopword* akan dihapus. Hasilnya, teks menjadi lebih ringkas dan fokus pada kata-kata yang lebih dalam konteks analisis.

3.2.2.4 **Lemmatization**

Tahap *Lemmatization* bertujuan untuk mengubah kata-kata dalam teks menjadi bentuk dasar (*lemma*) agar kata-kata yang memiliki akar yang sama dapat disatukan. Misalnya, kata "bermain", "bermainan", dan "dimainkan" akan diubah menjadi kata dasar "main". Hal ini penting untuk menyederhanakan analisis dan mencegah duplikasi makna dari kata dengan akar yang sama. Hasil penerapan proses *lemmatization* dapat dilihat pada Tabel 3.5**,** yang menunjukkan perubahan kata-kata dalam teks menjadi bentuk dasarnya.

Tabel 3.4 Contoh penerapan *lemmatization*

Sebelum *Lemmatization*	Sesudah *Lemmatization*
Kamu tu udah cantik, baik hati pula.	kamu tu sudah cantik baik hati pula
Dia itu jelek, sok pinter, norak dan bawaannya selalu bikin kesel.	dia itu jelek sok pintar norak bawaan selalu bikin kesal

```
# Lemmatization
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

factory = StemmerFactory()
stemmer = factory.create_stemmer()

words = [stemmer.stem(word) for word in words]
text = " ".join(words)
```

Gambar 3.6 Contoh kode program *lemmatization*

Gambar 3.6 menampilkan potongan kode program untuk melakukan *lemmatization.* Untuk melakukan proses *lemmatization* pada teks berbahasa Indonesia, digunakan *library* Sastrawi. *Library* ini menyediakan metode *stemming* dan *lemmatization* yang sesuai dengan kaidah bahasa Indonesia. Pada implementasinya, digunakan `StemmerFactory()` untuk membuat objek *stemmer*. Kemudian, dilakukan *lemmatization* pada setiap kata dalam teks menggunakan fungsi `stem()` agar kata yang memiliki bentuk turunan dapat diubah ke bentuk dasar.

3.2.2.5 **Tokenizing**

Tahap *Tokenizing* adalah proses memecah teks menjadi unit-unit terkecil yang disebut token. Token dapat berupa kata, frasa, atau simbol yang digunakan sebagai satu kesatuan makna dalam analisis teks. Contoh

hasil dari proses *tokenizing* disajikan pada Tabel 3.6**,** yang memperlihatkan bagaimana teks dipecah menjadi unit-unit terkecil berupa token. Token-token ini nantinya akan digunakan sebagai dasar dalam analisis lebih lanjut, seperti klasifikasi atau ekstraksi fitur.

Tabel 3.5 Contoh penerapan *tokenizing*

Sebelum *Tokenizing*	Sesudah *Tokenizing*
Tinggal sm u kaya nya seru bgt	[Tinggal, sm, u, kaya, nya, seru, bgt]
Muka lo kaya monyet ga pantes nongol di sini	[Muka, lo, kaya, monyet, ga, pantes, nongol, di, sini]

```
def tokenizing(teks):
    return word_tokenize(teks)

data['tokens'] = data['"full_text"'].apply(tokenizing)
```

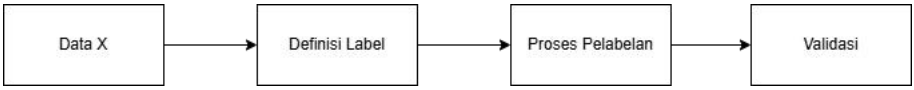
Gambar 3.7 Contoh kode program *tokenizing*

Gambar 3.7 menunjukkan potongan kode program yang digunakan untuk melakukan *tokenizing* pada teks. Fungsi `tokenizing(teks)` dibuat untuk memecah teks menjadi token-token dengan memanfaatkan fungsi `word_tokenize()` dari *library* NLTK. Selanjutnya, fungsi ini diterapkan dengan menyimpan hasilnya ke kolom baru bernama `tokens`.

3.2.3 Pelabelan Data

Proses pelabelan data dilakukan secara manual dengan tujuan untuk mengklasifikasikan setiap tweet ke dalam dua kategori utama, yaitu kelas *cyberbullying* dan *non-cyberbullying.* Pelabelan ini dilakukan dengan mengamati isi dan konteks dari masing-masing tweet, kemudian menetapkan label. Pelabelan manual merupakan metode yang umum digunakan dalam pembuatan dataset untuk keperluan pemrosesan bahasa alami. Menurut Pustejovsky dan Stubbs (2012), pelabelan oleh manusia

diperlukan karena hanya manusia yang dapat memahami konteks, makna, dan nuansa dalam bahasa, terutama pada kasus yang sensitif seperti ujaran kebencian atau *cyberbullying*. Hasil pelabelan manual ini dikenal sebagai *gold standard*, yaitu data yang dianggap paling akurat dan dijadikan acuan dalam pelatihan dan evaluasi model. Agar hasil pelabelan konsisten, proses ini biasanya dilakukan dengan pedoman yang jelas dan dapat melibatkan lebih dari satu pelabel (Pustejovsky & Stubbs, 2012).



Gambar 3.8 Alur Pelabelan Data

Dalam proses pada Gambar 3.8 ini, proses definisi label yaitu tweet yang mengandung unsur penghinaan, pelecehan, atau ujaran kebencian dikategorikan sebagai *cyberbullying* dan sebaliknya tweet yang tidak mengandung unsur tersebut dikategorikan sebagai *non cyberbullying*. Selanjutnya, tahap validasi label dilakukan untuk memastikan konsistensi dan akurasi hasil pelabelan. Validasi ini dilakukan dengan meminta peninjauan dari seorang ahli yang berlatar belakang pendidikan Bahasa Indonesia, guna mengevaluasi apakah klasifikasi tweet telah sesuai dengan makna, konteks, dan kaidah bahasa. Apabila ditemukan ketidaksesuaian atau ambiguitas dalam pemberian label, maka dilakukan revisi berdasarkan masukan dari ahli tersebut.

Contoh hasil dari proses pelabelan manual tersebut dapat dilihat pada Tabel 3.7, yang menunjukkan bagaimana tweet diklasifikasikan ke dalam dua kategori berdasarkan isi dan konteks kalimat.

Tabel 3.6 Contoh pelabelan data

Kalimat	Label
fase paling berbahaya adalah ketika kamu tidak merasakan apapun lagi tidak bahagia juga tidak sedih tapi yang ada hanya rasa hampa dan kosong.	*non-cyberbullying*
lu bukan kampungan lagi, norak senorak noraknya	*cyberbullying*

3.2.4 Word *embedding*

Tahap selanjutnya setelah proses *preprocessing* adalah word *embedding*, yaitu proses mengubah kata-kata dalam teks menjadi

representasi numerik berupa vektor berdimensi tetap. Dalam penelitian ini, digunakan tiga metode *embedding*, yaitu *FastText***,** *Word2Vec*, dan *GloVe*. Ketiga metode ini dipilih karena masing-masing memiliki keunggulan tersendiri dalam menangkap makna semantik dan sintaksis dari kata-kata dalam teks. *FastText* mampu menangkap makna kata dari sub-kata atau karakter n-gram, sehingga lebih baik dalam menangani kata-kata tidak umum atau kata baru. Sementara itu, *Word2Vec* efektif dalam memetakan hubungan semantik antar kata berdasarkan konteks tetangganya, dan *GloVe* memanfaatkan informasi statistik global dari corpus untuk menghasilkan representasi kata yang lebih stabil. Tabel 3.8 memperlihatkan contoh hasil representasi vektor dari beberapa kata setelah melalui proses *embedding* menggunakan ketiga metode tersebut.

Tabel 3.7 Penerapan *FastText embedding*

Kata	Vektor (sebagian nilai)
tidak	[0.027, -0.143, 0.088, ..., 0.091]
usah	[0.112, 0.005, -0.041, ..., -0.033]
ribet	[-0.092, 0.134, 0.065, ..., 0.024]
banget	[0.034, 0.119, -0.076, ..., -0.056]
deh	[0.018, -0.084, 0.096, ..., 0.012]
cek	[-0.052, 0.021, 0.077, ..., -0.098]
saja	[0.043, -0.017, 0.059, ..., 0.073]

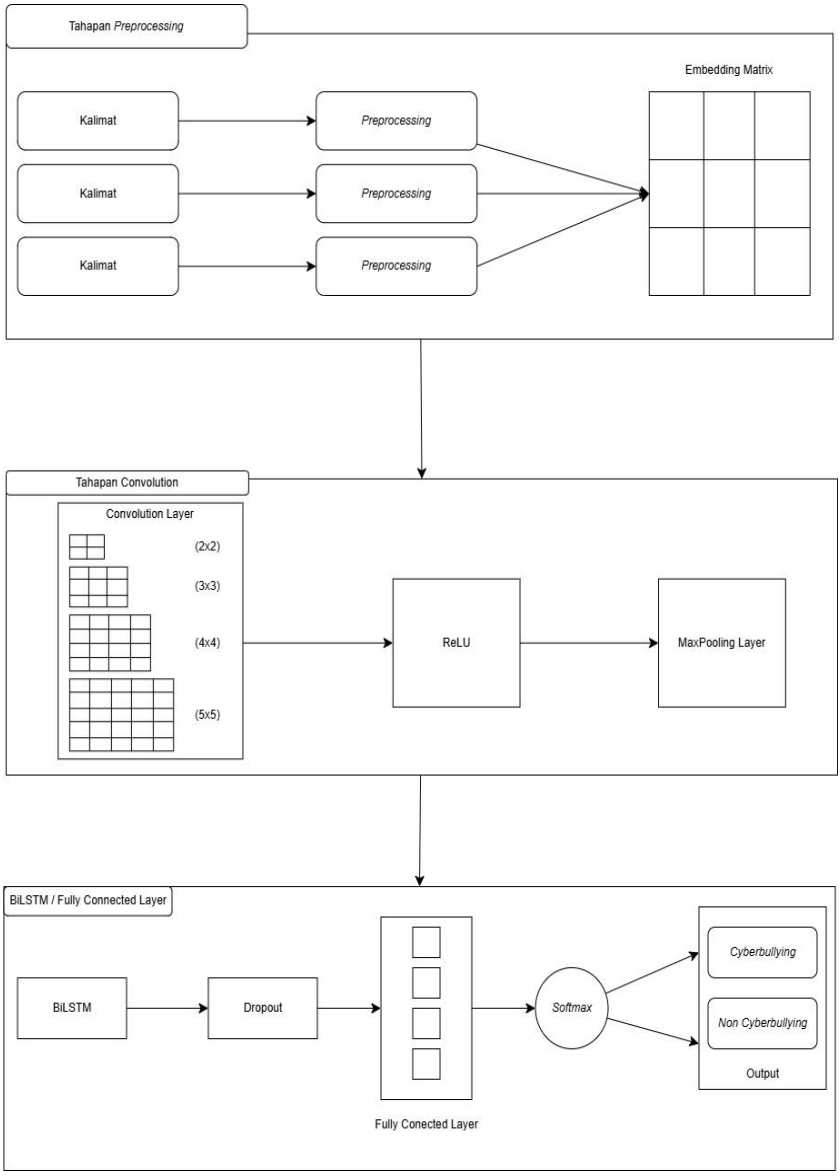
3.2.5 Modelling

Pada tahapan *modelling*, hasil dari tahapan *preprocessing* kemudian ditransformasikan ke dalam bentuk representasi vektorial melalui pembentukan *embedding matrix (FastText)*, yang merepresentasikan kata-kata dalam ruang vektor berdimensi tetap. Setelah data teks dikonversi menjadi representasi numerik, tahap berikutnya adalah ekstraksi fitur menggunakan lapisan konvolusional. Dalam tahap ini, digunakan beberapa kernel dengan ukuran berbeda (2×2, 3×3, 4×4, dan 5×5) untuk mendeteksi berbagai pola lokal seperti frasa pendek atau pasangan kata yang sering muncul bersamaan. Setiap hasil konvolusi diaktifkan menggunakan fungsi aktivasi *Rectified Linear Unit* (ReLU) yang memungkinkan model menangkap fitur non-linear. Untuk mereduksi dimensi dan mempertahankan informasi fitur yang paling penting, hasil dari ReLU kemudian diproses oleh lapisan *max pooling*.

Output dari CNN selanjutnya diteruskan ke lapisan BiLSTM guna menangkap informasi kontekstual dalam urutan kata secara lebih menyeluruh. BiLSTM memproses data dalam dua arah, yaitu *forward* (dari awal ke akhir) dan *backward* (dari akhir ke awal), sehingga dapat memahami hubungan antar kata baik yang muncul sebelum maupun sesudah dalam kalimat. Lapisan ini memungkinkan model untuk menangkap makna yang lebih mendalam terhadap urutan kata dalam kalimat. Untuk menghindari *overfitting* selama pelatihan, ditambahkan lapisan *dropout* sebelum masuk ke tahap klasifikasi.

Tahap akhir dalam arsitektur ini terdiri dari lapisan *fully connected* (*dense layer*) yang bertugas menggabungkan seluruh fitur hasil ekstraksi sebelumnya, dan dilanjutkan dengan fungsi aktivasi *softmax*. Fungsi *softmax* berperan mengubah nilai output menjadi probabilitas yang mengindikasikan kelas akhir dari data teks yang dianalisis. Dalam implementasi ini, model dirancang untuk klasifikasi biner, yakni membedakan apakah suatu teks tergolong dalam kategori *cyberbullying* atau *non-cyberbullying*.

Struktur keseluruhan dari proses ini dapat dilihat pada Gambar 3.9, yang menggambarkan arsitektur gabungan CNN dan BiLSTM yang digunakan dalam penelitian ini.



Gambar 3.9 Arsitektur CNN BiLSTM

3.2.6 Evaluasi

Pada tahap ini model akan dievaluasi menggunakan nilai *accuracy, precision, recall, F1- Score,* dan juga *loss* nya. Jika nilai *accuracy,*

precision, recall, dan *F1- Score* setiap *epoch* semakin naik dan *loss* semakin turun maka model mampu mempelajari data dengan baik. Sebaliknya jika *nilai accuracy, precision, recall, F1- Score*, dan *loss* konstan maka model tidak terlalu baik dalam mempelajari data. Untuk mengukur kinerja model secara menyeluruh, digunakan beberapa metrik evaluasi seperti yang ditunjukkan pada Gambar 3.10, yang berisi contoh kode program untuk melakukan evaluasi terhadap model.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import numpy as np

# Prediksi pada data uji
y_pred = np.argmax(model.predict(X_test), axis=1)
y_true = np.argmax(y_test, axis=1)

# Menghitung metrik evaluasi
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')

# Menampilkan hasil
print(f'Accuracy: {accuracy:.4f}')
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1-Score: {f1:.4f}')

# Menghitung dan menampilkan loss
loss, _ = model.evaluate(X_test, y_test, verbose=0)
print(f'Loss: {loss:.4f}')
```

Gambar 3.10 Evaluasi model

BAB 4
PEMBAHASAN DAN ANALISIS

Bab ini menyajikan hasil eksperimen dan analisis performa CNN–BiLSTM dalam klasifikasi komentar *cyberbullying*, termasuk CNN, LSTM, BiLSTM, dan CNN–LSTM. Setiap model dievaluasi menggunakan beberapa jenis *word embedding*, yaitu FastText, GloVe, dan Word2Vec, dengan variasi dimensi vektor. Hasil eksperimen disajikan dalam bentuk tabel dan grafik untuk mempermudah interpretasi, dengan fokus pada metrik seperti akurasi, *precision, recall, F1-score,* dan AUC.

4.1 Implementasi Pelabelan Data

Pelabelan data dilakukan secara manual dengan mengacu pada definisi bahwa komentar yang mengandung unsur penghinaan, pelecehan, atau ujaran kebencian dikategorikan sebagai *cyberbullying* (label 0), sedangkan komentar yang tidak mengandung unsur tersebut dikategorikan sebagai *non cyberbullying* (label 1).

Proses pelabelan dilakukan setelah tahap praproses data, dengan jumlah data yang dilabeli sebanyak 2543 komentar. Setiap komentar dibaca dan ditafsirkan berdasarkan konteks kalimat dan makna ujaran untuk menentukan label yang paling sesuai. Agar menjaga objektivitas dan validitas pelabelan, seluruh proses penilaian dilakukan oleh pelabel independen yang memiliki latar belakang keilmuan yang relevan, yaitu Oni Suryono, S.Pd, seorang Guru Bahasa Indonesia di SMAN 2 Teluk Kuantan. Berita acara pelabelan dapat dilihat pada Lampiran 1. Tabel 4.1 menunjukkan beberapa contoh hasil data komentar yang telah melalui proses pelabelan.

Tabel 4.1 Implementasi pelabelan data

Komentar	Label
hasil enggak kaget memang inter harus latih ajimumpung tambah skuad muda norak modal adrenalin yagabakal juara ucl malah inter remontada	0
kacaw kalo pola pikir seperti asal bacot lo tong postingan sih paling aktivis cuma ikut demo2 buat bahan posting biar kata keren kkkkknorak	0
chil chil memang kaki gede norak lo enggak pernah lihat bagus kaki gua gede pakai sepatu apa bagus lihat enggak	0

Komentar	Label
bantet sirik lu gituin dong beb	
luhut jokowi sepakat bangun daerah masing masing kaya keluarga masing masing kau yang ndeso luhut norak kau	0
hasil enggak kaget memang inter harus latih ajimumpung tambah skuad muda norak modal adrenalin yagabakal juara ucl malah inter remontada	0
cukup sekian jumpa moga bahagia	1
jangan deh mending cari bahagia dendiri	1
asal kau bahagia dj lampu kakak viwersnya paling banyak	1
kak semangat terus mungkin bukan baik buat kamu moga depan kamu dapat sempat jauh lebih ini amin jangan sedih kak bahagia selalu	1
hari aku menang terus kayak gampang banget nangisssnya hari lihat kucing menang tonton drakor menang lihat orang bahagia malah menang	1

4.2 Implementasi *Preprocessing* Data

Pada tahap ini, dilakukan serangkaian proses *preprocessing* untuk membersihkan dan menyiapkan data teks agar dapat digunakan dalam pelatihan model klasifikasi. Dalam penelitian ini, preprocessing diterapkan secara khusus pada kolom "full_text", yaitu kolom yang berisi komentar atau isi lengkap dari tweet yang dikumpulkan melalui proses *crawling*. Kolom ini dipilih karena memuat informasi utama yang menjadi dasar klasifikasi apakah sebuah tweet termasuk dalam kategori *cyberbullying* atau *non-cyberbullying*. Proses *preprocessing* yang telah diimplementasikan dalam penelitian ini meliputi beberapa tahapan utama, yaitu *case folding***,** *text cleaning***,** *stopword removal***,** *lemmatization***,** dan *tokenizing*. Setiap tahapan dilakukan secara berurutan dan sistematis untuk memastikan kualitas data yang optimal.

4.2.1 Hasil Implementasi Case Folding

Proses *case folding* telah diterapkan pada kolom *full_text*, yaitu kolom yang memuat komentar lengkap dari pengguna media sosial. Untuk memudahkan proses evaluasi, isi kolom tersebut terlebih dahulu disalin ke dalam kolom baru bernama *before_case_folding,* yang berfungsi sebagai data referensi sebelum dilakukan konversi. Selanjutnya, *case folding* diterapkan pada kolom tersebut dengan menggunakan fungsi *apply()* yang

menjalankan fungsi *case_folding* untuk mengubah semua huruf kapital menjadi huruf kecil.

Kode program yang digunakan untuk menjalankan proses ini ditampilkan pada Gambar 4.1. Hasilnya kemudian disimpan dalam kolom baru bernama *after_case_folding.* Untuk memvisualisasikan perubahan yang terjadi, hanya dua kolom tersebut yang ditampilkan melalui perintah *df_display.head(),* yang menampilkan lima baris pertama dari data sebelum dan sesudah case folding.

```
# Salin isi kolom asli ke kolom baru untuk dibandingkan
df['before_case_folding'] = df['"full_text"'].astype(str)

# Terapkan case folding ke kolom baru
df['after_case_folding'] = df['before_case_folding'].apply(case_folding)

df_display = df[['before_case_folding', 'after_case_folding']]
df_display.head()
```

Gambar 4.1 Kode program implementasi *case folding*

Gambar 4.2 menunjukkan hasil akhir dari proses ini. Terlihat bahwa seluruh karakter huruf kapital telah berhasil dikonversi menjadi huruf kecil secara konsisten tanpa mengubah struktur teks.

	before_case_folding	after_case_folding
0	"Hasilnya gak kaget. Emang pasti inter harusny...	"hasilnya gak kaget. emang pasti inter harusny...
1	"@ruddierawk @Duniamistikkk Kacaww kalo pola p...	"@ruddierawk @duniamistikkk kacaww kalo pola p...
2	"@pinkeufa chill chill emang kaki gede napa ??...	"@pinkeufa chill chill emang kaki gede napa ??...
3	"Luhut dan Jokowi bersepakat untuk membangun d...	"luhut dan jokowi bersepakat untuk membangun d...
4	"Kalo Luhut bilang orang-orang yang mengkritik...	"kalo luhut bilang orang-orang yang mengkritik...

Gambar 4.2 Hasil penerapan *case folding*

4.2.2 Hasil Implementasi Text Cleaning

Setelah dilakukan *case folding*, proses selanjutnya adalah *text cleaning*. Hasil dari *text cleaning* ditampilkan pada Gambar 4.3, yang memperlihatkan perbandingan data sebelum dan sesudah proses *cleaning*. Pada kolom *after_case_folding* terlihat bahwa data masih mengandung simbol, mention, serta tanda baca. Setelah melalui tahapan *cleaning*, data

yang ditampilkan pada kolom *after_cleaning* menjadi lebih bersih, seragam, dan siap untuk proses *preprocessing* lanjutan.

	after_case_folding	after_cleaning
0	"hasilnya gak kaget. emang pasti inter harusny...	hasilnya enggak kaget memang pasti inter harus...
1	"@ruddierawk @duniamistikkk kacaww kalo pola p...	kacaw kalo pola pikir anda seperti ini asal ba...
2	"@pinkeufa chill chill emang kaki gede napa ??...	chil chil memang kaki gede kenapa norak lo eng...
3	"luhut dan jokowi bersepakat untuk membangun d...	luhut dan jokowi bersepakat untuk membangun da...
4	"kalo luhut bilang orang-orang yang mengkritik...	kalo luhut bilang orang orang yang mengkritik ...
...	...	...
2450	"betul. kaya aku semangat cari uang demi bawa ...	betul kayak aku semangat cari uang demi bawa p...
2451	"gokiiil top 1 congrats all semangat nonton di...	gokiiil top satu congrats al semangat menonton...
2452	"gemecccccc semangat hari ini bebe "	gemec semangat hari ini bebe
2453	"@chynsukkloverr praktek jadi apa itu? semanga...	praktek jadi apa itu semangat ya
2454	"@dewriel semangat all the best apapun itu"	semangat al the best apapun itu

2455 rows × 2 columns

Gambar 4.3 Hasil penerapan *text cleaning*

4.2.3 Hasil Implementasi Remove Stopword

Pada penelitian ini, proses penghapusan *stopword* dilakukan menggunakan fungsi *remove()* dari pustaka *Sastrawi*, yang merupakan *library* populer untuk pengolahan teks Bahasa Indonesia.

Padaa kolom *after_cleaning* diterapkan fungsi tersebut, dan hasilnya disimpan dalam kolom baru *after_stopwords*. Gambar 4.4 memperlihatkan hasil dari proses ini. Dapat dilihat bahwa kata-kata yang bersifat fungsional dan tidak mengandung makna penting telah berhasil dihapus. Sebagai contoh, kalimat "luhut dan jokowi bersepakat untuk membangun daerahnya" berubah menjadi "luhut jokowi bersepakat membangun daerah", dan kalimat "semangat al the best apapun itu" disederhanakan menjadi "semangat al the best apapun".

	after_cleaning	after_stopwords
0	hasilnya enggak kaget memang pasti inter harus...	hasilnya enggak kaget memang inter harusnya pe...
1	kacaw kalo pola pikir anda seperti ini asal ba...	kacaw kalo pola pikir seperti asal bacot lo to...
2	chil chil memang kaki gede kenapa norak lo eng...	chil chil memang kaki gede norak lo enggak per...
3	luhut dan jokowi bersepakat untuk membangun da...	luhut jokowi bersepakat membangun daerahnya ma...
4	kalo luhut bilang orang orang yang mengkritik ...	kalo luhut bilang orang orang mengkritik kampu...
...	...	...
2450	betul kayak aku semangat cari uang demi bawa p...	betul kayak aku semangat cari uang bawa pulang...
2451	gokiiil top satu congrats al semangat menonton...	gokiiil top satu congrats al semangat menonton tv
2452	gemec semangat hari ini bebe	gemec semangat hari bebe
2453	praktek jadi apa itu semangat ya	praktek jadi apa semangat
2454	semangat al the best apapun itu	semangat al the best apapun

Gambar 4.4 Hasil penerapan *remove stopword*

4.2.4 Hasil Implementasi Lemmatization (Stemming Bahasa Indonesia)

Pada tahap desain sistem di Bab 3, penggunaan *lemmatization* telah dirancang untuk mengubah setiap kata menjadi bentuk dasarnya dengan mempertimbangkan konteks. Namun, dalam tahap implementasi, menyesuaikan pendekatan ini dengan menerapkan stemming menggunakan pustaka Sastrawi**,** mengingat keterbatasan ketersediaan pustaka *lemmatizer* yang mendukung Bahasa Indonesia secara optimal.

Proses *stemming* diterapkan pada kolom *after_stopwords,* dan hasilnya disimpan dalam kolom baru bernama *after_stemming*. Gambar 4.5 menampilkan hasil dari proses ini. Sebagai contoh, kalimat “luhut jokowi bersepakat membangun daerahnya masing-masing” berubah menjadi “luhut jokowi sepakat bangun daerah masing masing”, dan “kalo luhut bilang orang orang mengkritik kampung ini” menjadi “kalo luhut bilang orang orang kritik kampung ini”. Dengan diterapkannya stemming, teks menjadi lebih seragam, sederhana, dan siap digunakan untuk tahapan selanjutnya.

	before_stemming	after_stemming
0	hasilnya enggak kaget memang inter harusnya pe...	hasil enggak kaget memang inter harus latih aj...
1	kacaw kalo pola pikir seperti asal bacot lo to...	kacaw kalo pola pikir seperti asal bacot lo to...
2	chil chil memang kaki gede norak lo enggak per...	chil chil memang kaki gede norak lo enggak per...
3	luhut jokowi bersepakat membangun daerahnya ma...	luhut jokowi sepakat bangun daerah masing masi...
4	kalo luhut bilang orang orang mengkritik kampu...	kalo luhut bilang orang orang kritik kampung i...

Gambar 4.5 Hasil penerapan *stemming*

4.2.5 Hasil Implementasi Tokenizing

Pada tahap ini, melakukan proses tokenisasi terhadap data teks menggunakan kelas *Tokenizer* dari *library Keras*. *Tokenizer* ini dikonfigurasi dengan parameter *oov_token="<OOV>"* untuk mengantisipasi kemunculan kata-kata yang tidak dikenal (Out-of-Vocabulary) saat proses pelatihan dan pengujian model. Seluruh data teks kemudian diproses menggunakan fungsi *fit_on_texts,* yang bertujuan membentuk kamus kata berdasarkan kemunculan kata-kata unik dalam kamus. Setiap kata dalam kamus diberi indeks numerik yang digunakan sebagai representasi. Setelah kamus terbentuk, fungsi *texts_to_sequences* digunakan untuk mengubah setiap teks menjadi urutan angka sesuai indeks kata yang telah ditentukan.

Karena panjang setiap teks tidak seragam, menerapkan proses *padding* menggunakan fungsi *pad_sequences* dengan parameter *padding='post'*, sehingga seluruh urutan kata memiliki panjang yang sama berdasarkan urutan terpanjang yang ditemukan dalam korpus. Hasil dari proses ini adalah matriks input x yang berisi representasi numerik dari seluruh teks, serta array y yang berisi label target. Gambar 4.6 menunjukkan potongan kode implementasi *tokenizing*.

```python
# Tokenizer and padding
tokenizer = Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
max_len = max(len(seq) for seq in sequences)
X = pad_sequences(sequences, maxlen=max_len, padding='post')
y = np.array(labels)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
vocab_size = len(tokenizer.word_index) + 1
```

Gambar 4.6 Kode proses tokenisasi dan padding data teks

Selanjutnya, data dibagi menjadi data latih dan data uji dengan rasio 80:20 menggunakan fungsi *train_test_split*. Pembagian ini dilakukan untuk mengevaluasi performa model secara objektif. Jumlah kosakata yang berhasil diidentifikasi oleh *tokenizer* ditentukan dengan menghitung panjang *word_index,* ditambah satu untuk mengakomodasi token *padding.* Hasil dari *tokenizing* dan *padding* dapat dilihat pada Gambar 4.7.

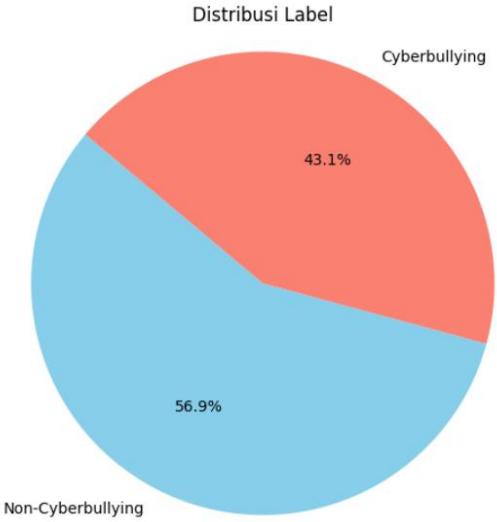
```
=== Hasil Tokenizing ===
Teks Asli    : hasil enggak kaget memang inter harus latih ajimumpung tambah skuad muda norak modal adrenalin yagabakal juara ucl malah inter remontada
Tokenisasi   : [162, 3, 924, 26, 230, 117, 501, 2379, 265, 2380, 355, 6, 611, 2381, 2382, 197, 219, 48, 230, 1161]

Teks Asli    : kacaw kalo pola pikir seperti asal bacot lo tong postingan sih paling aktivis cuma ikut demo2 buat bahan posting biar kata keren kkkkknorak
Tokenisasi   : [2383, 14, 791, 115, 1162, 502, 231, 42, 2384, 1556, 20, 98, 1557, 90, 110, 2385, 18, 407, 1558, 65, 86, 157, 2386]

Teks Asli    : chil chil memang kaki gede norak lo enggak pernah lihat bagus kaki gua gede pakai sepatu apa bagus lihat enggak bantet sirik lu gituin dong beb
Tokenisasi   : [1163, 1163, 26, 503, 247, 6, 42, 3, 81, 27, 158, 503, 50, 247, 61, 792, 23, 158, 27, 3, 2387, 925, 15, 2388, 187, 926]

Teks Asli    : luhut jokowi sepakat bangun daerah masing masing kaya keluarga masing masing kau yang ndeso luhut norak kau
Tokenisasi   : [612, 430, 2389, 431, 1164, 296, 296, 545, 127, 296, 296, 163, 34, 927, 612, 6, 163]

Teks Asli    : kalo luhut bilang orang orang kritik kampung iri urus penting politik luhut saya bilang ndeso norak
Tokenisasi   : [14, 612, 101, 8, 8, 793, 167, 356, 174, 232, 1165, 612, 794, 101, 927, 6]

=== Hasil Setelah Padding ===
[[ 162    3  924   26  230  117  501 2379  265 2380  355    6  611 2381
  2382  197  219   48  230 1161    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0]
 [2383   14  791  115 1162  502  231   42 2384 1556   20   98 1557   90
   110 2385   18  407 1558   65   86  157 2386    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0]
 [1163 1163   26  503  247    6   42    3   81   27  158  503   50  247
    61  792   23  158   27    3 2387  925   15 2388  187  926    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0]
 [ 612  430 2389  431 1164  296  296  545  127  296  296  163   34  927
   612    6  163    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0]
 [  14  612  101    8    8  793  167  356  174  232 1165  612  794  101
   927    6    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0]]
```

Gambar 4.7 Hasil penerapan *tokenizing* dan *padding*

4.3 Exploratory Data Analysis (EDA)

Analisis data eksploratori (EDA) dilakukan untuk memahami karakteristik awal dari data yang digunakan dalam penelitian, termasuk

distribusi kelas dan pola penggunaan kata dalam komentar. Tahap ini bertujuan untuk mengidentifikasi struktur dan kecenderungan data yang dapat memengaruhi proses pemodelan. Beberapa visualisasi digunakan untuk memberikan gambaran awal terkait sebaran data dan karakteristik linguistik yang khas dari masing-masing kategori komentar.



Gambar 4.8 Distribusi pelabelan data

Gambar 4.8 menunjukkan distribusi label pada dataset yang digunakan dalam penelitian ini. Hasil pelabelan manual yang telah dilakukan, diperoleh sebanyak 1096 tweet yang termasuk dalam kategori *cyberbullying* dan 1447 tweet dalam kategori *non-cyberbullying*. Dari total 2543 komentar yang telah melalui proses pelabelan dan validasi, sebanyak 56,9% di antaranya termasuk dalam kategori *non cyberbullying*, sedangkan 43,1% lainnya termasuk dalam kategori *cyberbullying*. Meskipun tidak terlalu timpang, proporsi ini menunjukkan adanya ketidakseimbangan kelas (*class imbalance*) yang dapat memengaruhi kinerja model klasifikasi.

Untuk mengatasi ketidakseimbangan tersebut, diterapkan teknik SMOTE (*Synthetic Minority Oversampling Technique*) pada data latih. Teknik ini digunakan untuk menyeimbangkan jumlah data antara kelas minoritas dan mayoritas dengan cara menghasilkan sampel sintetis pada kelas minoritas berdasarkan karakteristik data yang telah ada. SMOTE hanya diterapkan pada data latih, agar tidak mengganggu proses validasi

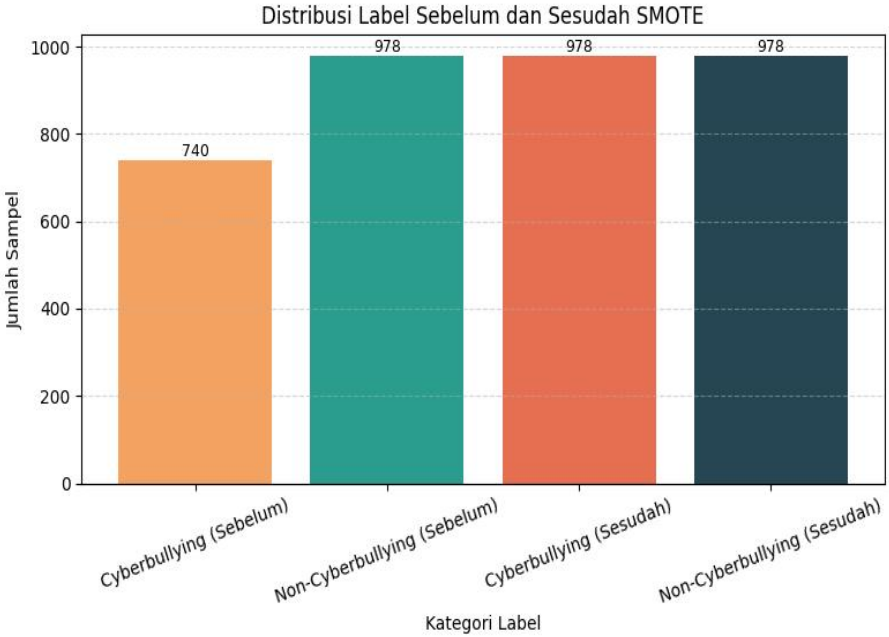
dan evaluasi model secara objektif. Gambar 4.9 menunjukan kode program dalam mengimplementasikan SMOTE.

```
# SMOTE
X_resampled, y_resampled = SMOTE(random_state=42).fit_resample(X_train, y_train)
```

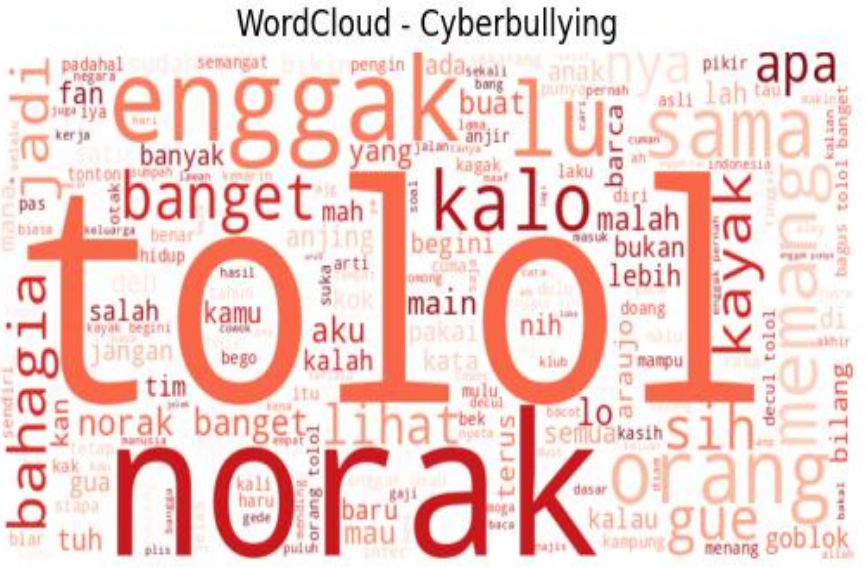
Gambar 4.9 Kode program penerapan SMOTE pada tiap model

Sebelum penerapan SMOTE, jumlah data latih sebanyak 1718 data, yang terdiri dari 978 data *non-cyberbullying* (kelas mayoritas) dan 740 data *cyberbullying* (kelas minoritas). Setelah diterapkan SMOTE, kelas *cyberbullying* diperbanyak dengan menambahkan 238 data sintetis hingga jumlahnya menjadi 978 data, sama dengan kelas mayoritas. Dengan demikian, total data latih setelah proses SMOTE menjadi 1956 data yang seimbang.

Cara kerja SMOTE secara sederhana adalah dengan memilih satu data dari kelas minoritas, lalu mencari beberapa tetangga terdekatnya (menggunakan metode *K-Nearest Neighbors*). Setelah itu, SMOTE membuat data baru dengan mengambil nilai-nilai di antara data tersebut dan tetangganya—proses ini disebut interpolasi. Data baru yang dihasilkan bukan merupakan salinan, melainkan kombinasi dari data asli yang sudah ada, sehingga dapat membantu model belajar lebih baik tanpa membuat data menjadi duplikat. Distribusi data latih setelah proses penyeimbangan ini dapat dilihat pada Gambar 4.10.

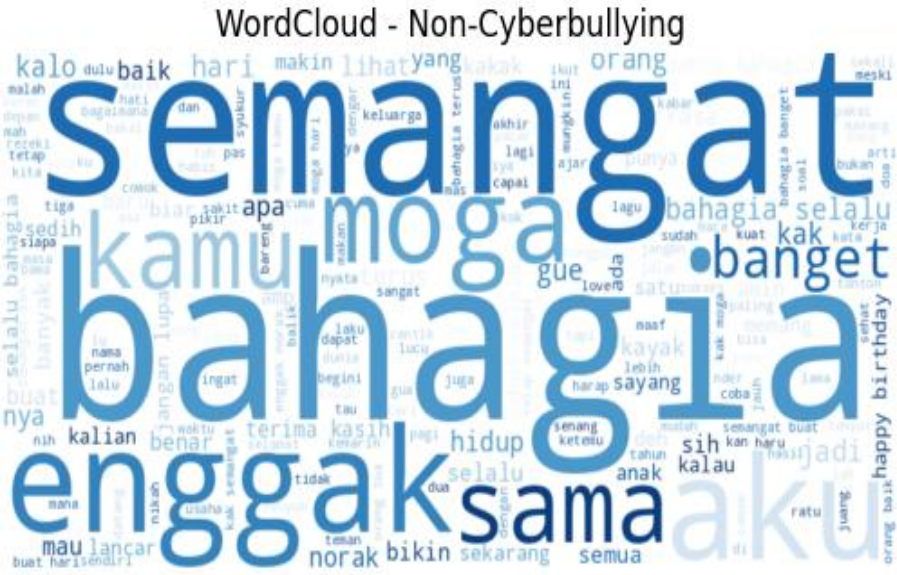


Gambar 4.10 Perbandingan data sebelum dan sesudah penerapan SMOTE



Gambar 4.11 *Word cloud* komentar *cyberbullying*

Gambar 4.11 menunjukkan visualisasi *word cloud* dari komentar yang termasuk dalam kategori *cyberbullying*. *Word cloud* merupakan representasi visual dari kumpulan kata, di mana ukuran setiap kata mencerminkan frekuensi kemunculannya dalam data teks. Semakin besar ukuran suatu kata, semakin sering kata tersebut muncul. Dalam *word cloud* ini, tampak dominasi kata-kata bernada negatif dan ofensif seperti "tolol", "kamu", "norak", "otak", dan "decul". Kata "tolol" menjadi yang paling menonjol, menandakan bahwa kata tersebut paling sering digunakan dalam komentar yang mengandung unsur penghinaan atau pelecehan. Selain itu, terdapat juga kata-kata lain seperti "anak kampung", "goblok", dan "main" yang sering digunakan dalam konteks merendahkan. Hal ini mencerminkan karakteristik utama dari komentar *cyberbullying*, yaitu penggunaan bahasa yang menyerang secara verbal dan dapat berdampak negatif terhadap korban.



Gambar 4.12 *Word cloud* komentar *non cyberbullying*

Gambar 4.12 merupakan *word cloud* dari komentar yang termasuk dalam kategori *non cyberbullying*. Berbeda dari sebelumnya, *word cloud* ini didominasi oleh kata-kata dengan makna netral hingga positif, seperti

“semangat”, “kamu”, “bahagia”, “selalu”, dan “semoga”. Ukuran besar kata-kata tersebut menunjukkan bahwa komentar dalam kategori ini cenderung berisi dukungan, motivasi, ucapan selamat, dan ekspresi positif lainnya. Pola penggunaan kata pada *word cloud* ini memperkuat karakteristik komentar *non cyberbullying* yang tidak mengandung unsur penghinaan atau serangan verbal, melainkan lebih mengedepankan pesan yang membangun atau menyenangkan.

4.4 Implementasi Word Embedding

Dalam penelitian ini, ketiga jenis *embedding* yang digunakan adalah FastText, Word2Vec, dan representasi mirip GloVe. Setiap jenis *embedding* memiliki metode dan sumber data yang berbeda dalam proses pembentukannya.

4.4.1 Implementasi FastText

Model FastText digunakan model Bahasa Indonesia yang telah dilatih oleh Facebook AI menggunakan data dari *Common Crawl* (kumpulan besar teks dari internet). Pada penelitian ini, model *FastText* dengan dimensi 300 digunakan dan dimuat dari berkas *pretrained* cc.id.300.bin (Bahasa Indonesia). Dalam penelitian ini, model FastText tidak hanya digunakan secara langsung, tetapi juga dikonversi ke dalam format teks mirip GloVe dan disimpan sebagai file GloVe_id.txt. Format ini mempermudah proses pembacaan vektor kata dalam sistem yang hanya mendukung format teks. Gambar 4.13 menunjukkan proses konversi vektor kata dari model FastText berbahasa Indonesia ke dalam format teks (.txt) yang menyerupai format GloVe. Hasil konversi ini memungkinkan *embedding* digunakan sebagai alternatif GloVe dalam eksperimen model klasifikasi.

```
ft_model = load_facebook_model("/content/drive/MyDrive/resources/cc.id.300.bin")
with open("glove_id.txt", "w", encoding="utf-8") as f:
    for word in ft_model.wv.index_to_key:
        vec = ft_model.wv[word]
        vec_str = ' '.join(map(str, vec))
        f.write(f"{word} {vec_str}\n")
```

Gambar 4.13 Cuplikan Kode Konversi *FastText* ke Format *GloVe*-style

Untuk memastikan bahwa setiap kata berhasil direpresentasikan dengan baik oleh FastText, Gambar 4.14 menampilkan contoh output dari model FastText berbahasa Indonesia. Setiap kata, seperti *bahagia*, *tidak*, *aku*, dan lainnya direpresentasikan dalam bentuk vektor. Representasi ini

memungkinkan model klasifikasi memahami makna semantik dari kata-kata berdasarkan distribusi konteksnya dalam korpus pelatihan.

```
📌 Training model: FASTTEXT 300D
📥 Loading FastText from: /content/drive/MyDrive/resources/cc.id.300.bin

📌 Contoh kata dari FASTTEXT 300D:
<OOV>: [-0.00119659  0.00295095 -0.00195914  0.00128701 -0.00136976]...
bahagia: [-0.00831283 -0.10030977 -0.02840042  0.06064843 -0.04130146]...
tidak: [ 0.03851964  0.00517239  0.0328753   0.10336006 -0.03426373]...
aku: [ 0.01258465 -0.05965358  0.09049556  0.29484874  0.02158902]...
kamu: [ 0.067098   -0.02873558  0.05053357  0.1828715  -0.01424146]...
semangat: [-0.05048427 -0.039581   -0.02468412  0.00075645 -0.04007722]...
tolol: [ 0.06908239  0.02892014 -0.0007012   0.08941144 -0.01374871]...
norak: [ 0.07936431  0.01482968 -0.07891081  0.01746366  0.01150789]...
sekali: [ 0.01400043 -0.02712647 -0.00084962  0.05114507 -0.04342949]...
orang: [ 0.04398865 -0.03032926 -0.01770458  0.12002504 -0.08117092]...
```

Gambar 4.14 Representasi vektor kata dari *FastText* 300D

4.4.2 Implementasi GloVe

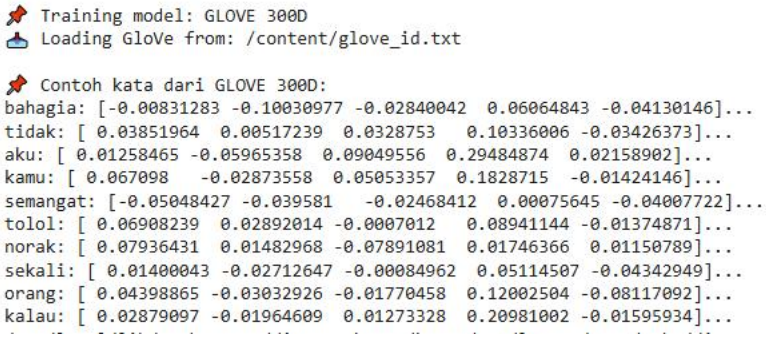
GloVe (*Global Vectors for Word Representation*) merupakan metode *embedding* yang menyusun vektor kata, berdasarkan matriks keterkaitan kata terhadap kata lain dalam konteks global untuk menghasilkan representasi kata. Namun, karena tidak tersedia *pretrained* GloVe khusus untuk Bahasa Indonesia, maka pada penelitian ini digunakan pendekatan alternatif, yaitu dengan mengonversi hasil *embedding FastText* ke format yang menyerupai GloVe.

File GloVe_id.txt yang dihasilkan dari proses sebelumnya berisi kata dan vektor yang dipisahkan oleh spasi, sesuai dengan struktur file GloVe yang umum digunakan. File ini kemudian dibaca ulang secara manual, dan setiap kata beserta vektornya dimasukkan ke dalam *dictionary Python* sebagai *word embedding*. Gambar 4.15 menunjukkan cuplikan kode yang digunakan untuk memuat file *embedding GloVe-style* ke dalam struktur data.

```
with open(path, encoding='utf-8') as f:
    for line in f:
        parts = line.strip().split()
        word = parts[0]
        vector = np.asarray(parts[1:], dtype='float32')
        wv[word] = vector
```

Gambar 4.15 Cuplikan kode pembacaan *embedding GloVe-style*

Gambar 4.16 memperlihatkan representasi vektor kata dari model GloVe berdimensi 300. Tidak seperti FastText, GloVe menghasilkan satu vektor tetap untuk setiap kata, yang berarti bahwa representasi kata tidak berubah meskipun digunakan dalam konteks kalimat yang berbeda. Vektor tersebut dibentuk berdasarkan statistik *co-occurrence* global di seluruh korpus pelatihan.



Gambar 4.16 Representasi vektor kata dari *GloVe* 300D

4.4.3 Implementasi Word2Vec

Berbeda dari FastText dan GloVe, Word2Vec menghasilkan representasi kata berdasarkan konteks lokal menggunakan algoritma CBOW (*Continuous Bag of Words*) atau Skip-Gram. Dalam penelitian ini, Word2Vec dilatih secara mandiri menggunakan korpus Wikipedia Bahasa Indonesia.

Korpus Wikipedia dalam format `.bz2` pertama-tama diekstrak menggunakan modul `gensim.corpora.WikiCorpus`. Proses ini mengubah artikel Wikipedia menjadi baris-baris teks yang bersih. Selanjutnya, teks tersebut diproses menjadi token per kalimat menggunakan fungsi `simple_preprocess` dari Gensim.

Setelah korpus siap, pada penelitian ini proses pelatihan Word2Vec diuji dalam tiga dimensi (50, 100, 300) untuk melihat pengaruh ukuran vektor terhadap hasil klasifikasi. Sedangkan FastText dan GloVe hanya digunakan pada dimensi 300 karena versi *pre-trained* yang umum digunakan dan tersedia adalah yang berdimensi 300, serta dianggap cukup representatif dalam menangkap konteks semantik kata. Masing-masing *embedding* disimpan dalam format *`Word2Vec`*`_id_{dim}.bin`, dan digunakan secara terpisah untuk melatih model yang berbeda. *Embedding*

Word2Vec yang dihasilkan kemudian dimuat dan dikonversi menjadi matriks *embedding* berdasarkan indeks kata dari *tokenizer*. Kode program dapat dilihat pada Gambar 4.17.

```
# Ubah file Wikipedia XML.bz2 menjadi teks bersih
def extract_wiki_text(wiki_bz2_path, output_txt_path):
    wiki = WikiCorpus(wiki_bz2_path, dictionary={})
    with open(output_txt_path, 'w', encoding='utf-8') as output:
        for text in tqdm(wiki.get_texts(), desc="🔲 Extracting wiki"):
            line = ' '.join(text)
            output.write(line + '\n')

extract_wiki_text(wiki_bz2_path, output_txt_path)

# Generator token per kalimat
def read_corpus(file_path):
    with open(file_path, encoding='utf-8') as f:
        for line in f:
            yield simple_preprocess(line)

sentences = list(read_corpus("/content/drive/MyDrive/resources/idwiki_text.txt"))

for dim in [50, 100, 200, 300]:
    model = Word2Vec(sentences=sentences, vector_size=dim, window=5, min_count=5, workers=4)
    model.wv.save_word2vec_format(f"word2vec_id_{dim}.bin", binary=True)
```

Gambar 4.17 Cuplikan kode pembacaan *embedding Word2Vec*

4.5 Implementasi Model

Pada bagian ini dijelaskan proses implementasi masing-masing model *deep learning* yang digunakan dalam penelitian, baik model tunggal maupun kombinasi model, yang dikombinasikan dengan tiga jenis *word embedding*, yaitu FastText, GloVe, dan Word2Vec. Setiap model memulai proses dengan memuat matriks *embedding* yang telah dilatih sebelumnya sesuai dengan jenis dan dimensi *embedding* (dimensi 300 untuk GloVe dan FastText, serta 50, 100, 200, dan 300 untuk Word2Vec). Semua dataset kemudian dibagi menggunakan teknik *stratified split* dengan rasio 70:15:15 untuk data latih, validasi, dan uji, guna menjaga proporsi kelas yang seimbang.

4.5.1 Implementasi Model CNN – BiLSTM

Pengembangan arsitektur model CNN–BiLSTM dimulai menambahkan lapisan Conv1D yang terdiri dari 128 filter dan kernel berukuran 5 untuk mengekstraksi fitur lokal dari representasi vektor teks. Lapisan ini berperan penting dalam mengekstraksi pola lokal dari sekuens teks yang telah direpresentasikan dalam bentuk vektor *embedding*. Untuk

mengurangi kompleksitas data dan mempertahankan fitur penting, hasil konvolusi ini kemudian diproses melalui *MaxPooling1D*. Proses ini dilanjutkan dengan *BatchNormalization* guna menstabilkan distribusi nilai aktivasi, mempercepat konvergensi, dan mengurangi sensitivitas terhadap inisialisasi parameter.

Selanjutnya, output dari proses konvolusi diteruskan ke lapisan BiLSTM yang terdiri dari 96 unit guna mengidentifikasi keterkaitan makna antar kata yang terpisah secara posisi dalam sekuens secara dua arah (*forward* dan *backward*). Untuk mencegah *overfitting*, diterapkan dropout sebesar 0.3 dan regularisasi pada lapisan ini. Setelah itu, *GlobalMaxPooling1D* digunakan untuk merangkum informasi dari seluruh sekuens, yang kemudian dinormalisasi kembali dengan *LayerNormalization*. Proses klasifikasi dilanjutkan melalui tiga lapisan *Dense* berturut-turut dengan jumlah unit 128, 64, dan 32 serta fungsi aktivasi ReLU. Masing-masing lapisan ini juga dilengkapi dengan mekanisme dropout dan regularisasi L2 (kernel_regularizer = l2(0.001)). Terakhir, lapisan *output* menggunakan satu neuron *Dense* dengan aktivasi sigmoid untuk menghasilkan *output* klasifikasi biner. Evaluasi model mencakup metrik Accuracy, AUC, *Precision*, *Recall*, dan F1-score, serta disertai dengan visualisasi seperti grafik kurva pelatihan dan confusion matrix guna memperoleh gambaran menyeluruh terhadap performa model. Arsitektur model CNN–BiLSTM untuk masing-masing jenis *embedding* divisualisasikan pada Gambar 4.18.

```
# ========== Build CNN + BiLSTM Model ==========
inp = Input(shape=(max_len,))
x = Embedding(input_dim=vocab_size, output_dim=dim, weights=[embedding_matrix], trainable=True)(inp)
x = Dropout(0.3)(x)

x = Conv1D(128, 5, activation='relu')(x)
x = MaxPooling1D(pool_size=2)(x)

x = Bidirectional(LSTM(64, return_sequences=True, kernel_regularizer=l2(0.001)))(x)
x = Dropout(0.3)(x)

x = Bidirectional(LSTM(32, kernel_regularizer=l2(0.001)))(x)
x = Dropout(0.3)(x)

x = Dense(64, activation='relu', kernel_regularizer=l2(0.001))(x)
x = Dropout(0.3)(x)

out = Dense(1, activation='sigmoid')(x)

model = Model(inputs=inp, outputs=out)
model.compile(loss='binary_crossentropy', optimizer=Adam(1e-5), metrics=['accuracy', 'AUC'])

# ========== Training ==========
early_stop = EarlyStopping(monitor='val_loss', patience=6, restore_best_weights=True)
history = model.fit(
    X_resampled, y_resampled,
    epochs=50,
    batch_size=64,
    validation_data=(X_val, y_val),
    callbacks=[early_stop],
    verbose=0
)
```

Gambar 4.18 Implementasi arsitektur CNN BiLSTM

4.5.2 Implementasi Model Lainnya

Selain arsitektur utama CNN–BiLSTM yang menjadi fokus dalam penelitian ini, beberapa model *deep learning* lain juga diimplementasikan. Tujuan implementasi model-model ini adalah untuk memperoleh pemahaman yang lebih komprehensif mengenai kinerja arsitektur alternatif dalam tugas klasifikasi komentar bermuatan *cyberbullying* pada media sosial. Setiap model dirancang dengan prinsip arsitektur yang seragam, penerapan regularisasi, dan konfigurasi pelatihan yang setara agar perbandingan performa dapat dilakukan secara adil dan objektif.

Model BiLSTM dibangun menggunakan dua lapisan LSTM berarah ganda (*Bidirectional*). Lapisan pertama terdiri dari 64 unit dan dikonfigurasi untuk mengembalikan seluruh urutan keluaran

(*return_sequences=True*) agar dapat diteruskan ke lapisan BiLSTM kedua yang terdiri dari 32 unit. Setiap lapisan BiLSTM dilengkapi dengan regularisasi L2 (kernel_regularizer = l2(0.001)) dan dropout sebesar 0.3. Setelah melalui lapisan BiLSTM, keluaran jaringan diarahkan ke lapisan Dense sebanyak satu tahap dengan 64 unit dan aktivasi ReLU. Lapisan ini juga menggunakan regularisasi L2 dan dropout tambahan. Terakhir, lapisan output terdiri dari satu neuron Dense dengan fungsi aktivasi *sigmoid*. Seluruh proses pelatihan dikendalikan oleh *EarlyStopping* dengan pemantauan *val_loss* selama 6 *epoch* tanpa perbaikan.

Model LSTM menerapkan struktur berlapis tunggal searah yang juga terdiri dari dua lapisan, masing-masing 64 dan 32 unit. Konfigurasi *return_sequences* digunakan pada lapisan pertama untuk melanjutkan urutan ke lapisan berikutnya. Sama seperti pada BiLSTM, digunakan regularisasi L2 dan dropout 0.3 pada tiap lapisan. Setelah melalui pemrosesan sekuensial, hasilnya diarahkan ke lapisan Dense 64 unit dengan aktivasi ReLU dan regularisasi tambahan, kemudian menuju neuron sigmoid untuk klasifikasi. Konfigurasi pelatihan juga menggunakan *binary_crossentropy* sebagai fungsi loss, Adam optimizer (*learning rate 1e-5*), serta strategi *EarlyStopping* yang sama.

Model CNN dibangun dengan dua lapisan konvolusi satu dimensi (Conv1D) secara berurutan. Lapisan pertama terdiri dari 128 filter yang bertugas mengekstrak pola lokal, dan dilengkapi aktivasi ReLU serta regularisasi L2. Hasil dari Conv1D diringkas melalui MaxPooling1D dengan ukuran pool 2. Selanjutnya, representasi fitur diteruskan ke lapisan Dense 64 unit dengan aktivasi ReLU dan dropout 0.3, kemudian ke lapisan output sigmoid. CNN efektif dalam menangkap n-gram dalam representasi teks, meskipun tidak menangkap konteks urutan jangka panjang.

Model CNN–LSTM menggabungkan kekuatan konvolusi dan sekuensial. Setelah proses *embedding*, data diproses oleh lapisan Conv1D dengan 128 filter untuk mengekstrak fitur lokal, lalu diperkecil dimensinya dengan MaxPooling1D. Hasil ekstraksi ini kemudian dimasukkan ke dalam LSTM 64 unit untuk memahami urutan waktu atau konteks temporal dari fitur tersebut. Model ini dilengkapi dengan regularisasi L2 pada lapisan LSTM, dropout dua kali, dan lapisan Dense 64 unit sebelum ke sigmoid. Berbeda dengan CNN–BiLSTM, model ini hanya memproses urutan secara satu arah.

Seluruh model dikompilasi menggunakan fungsi *loss binary_crossentropy*, *optimizer* Adam dengan *learning rate kecil* (1e-5), serta evaluasi performa dilakukan menggunakan metrik akurasi, AUC, *precision, recall,* dan *F1-score*. Hasil dari pelatihan dan pengujian setiap model divisualisasikan dengan grafik akurasi dan *loss*, *confusion matrix*, serta *classification report*. Tabel 4.2 berikut merangkum struktur arsitektur inti dari setiap model pembanding:

Tabel 4.2 Arsitektur model lainnya

Model	Arsitektur Inti
BiLSTM	BiLSTM (64) → BiLSTM (32) → Dense (64, ReLU) → Sigmoid Output
LSTM	LSTM (64) → LSTM (32) → Dense (64, ReLU) → Sigmoid Output
CNN	Conv1D (128, ReLU) → MaxPooling1D → Dense (64, ReLU) → Sigmoid Output
CNN–LSTM	Conv1D (128, ReLU) → MaxPooling1D → LSTM (64) → Dense (64, ReLU) → Sigmoid Output

4.6 Evaluasi dan Analisis Performa Model

Bagian ini menyajikan hasil evaluasi dan analisis performa dari berbagai model deep learning yang telah digunakan untuk tugas klasifikasi *cyberbullying* pada media sosial. Evaluasi dilakukan berdasarkan beberapa metrik, yaitu akurasi**,** *precision***,** *recall***,** dan *F1-score***,** dengan mempertimbangkan variasi penggunaan metode *word embedding*. Seluruh model diuji menggunakan arsitektur yang konsisten serta parameter pelatihan yang setara agar hasil evaluasi adil dan dapat dibandingkan secara objektif.

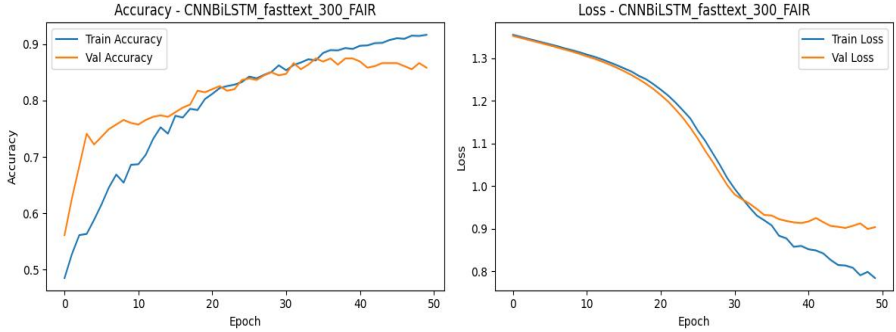
4.6.1 Evaluasi dan Analisis Performa Model CNN – BiLSTM

Bagian ini membahas evaluasi performa model CNN – BiLSTM dengan tiga jenis *word embedding* yang berbeda. Tujuannya adalah untuk

melihat bagaimana kombinasi arsitektur ini merespons variasi representasi kata dalam tugas klasifikasi *cyberbullying*.

4.6.1.1 CNN – BiLSTM Word *Embedding FastText*

Visualisasi akurasi dan *loss* diperlukan untuk menilai apakah model CNN–BiLSTM telah belajar secara optimal dan konsisten. Gambar 4.22 berikut memperlihatkan grafik hasil pelatihan selama 50 epoch menggunakan *embedding* FastText berdimensi 300.

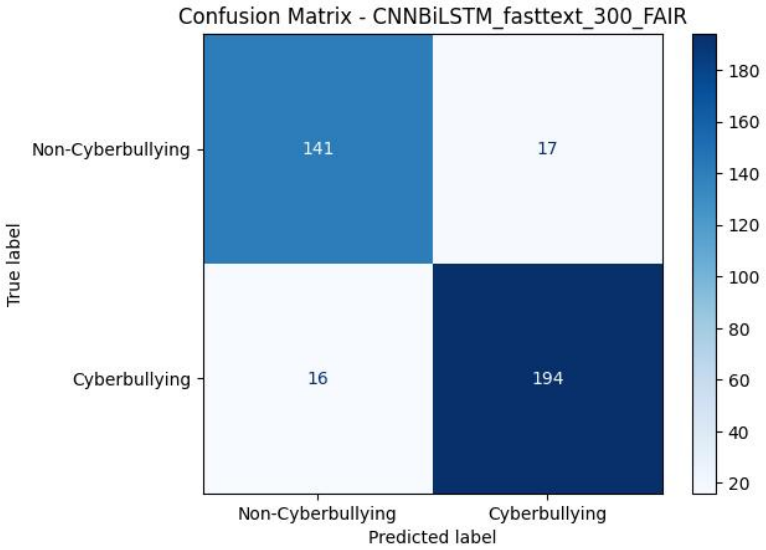


Gambar 4.19 Grafik akurasi dan *loss* CNN BiLSTM FastText

Gambar 4.19 menampilkan grafik akurasi dan *loss* selama proses pelatihan. Pada grafik akurasi, terlihat bahwa model berhasil mencapai akurasi validasi di atas 90% pada *epoch* ke-35 dan terus meningkat secara bertahap hingga akhir pelatihan. Namun, perlu diperhatikan bahwa mulai sekitar *epoch* ke-30, terdapat pola divergensi antara kurva akurasi pelatihan dan validasi, yang mulai membentuk pola terbuka. Fenomena ini juga terlihat pada grafik *loss*, di mana penurunan *loss* pada data pelatihan terjadi lebih tajam dibandingkan dengan *loss* pada data validasi yang mulai mendatar, sehingga menciptakan celah yang semakin lebar.

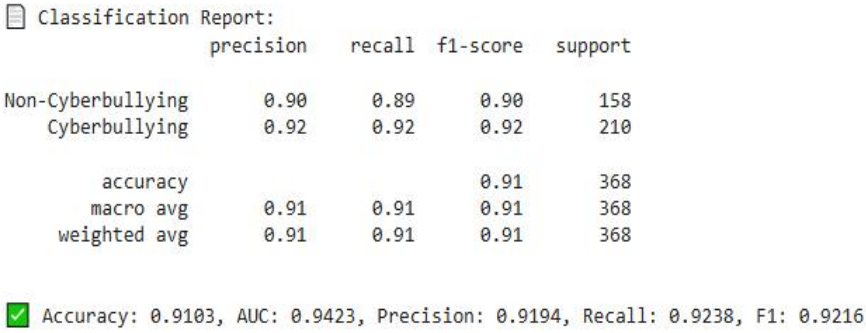
Pola ini menunjukkan indikasi awal dari *overfitting* ringan, yaitu kondisi di mana model semakin baik dalam mengingat data pelatihan tetapi mulai kehilangan kemampuan untuk menggeneralisasi terhadap data baru. Meskipun demikian, *overfitting* ini belum mencapai tahap kritis karena nilai akurasi dan *loss* pada data validasi masih cukup stabil, tanpa menunjukkan fluktuasi ekstrem atau penurunan kinerja yang drastis. Model menunjukkan kemampuan dalam mempelajari pola data secara optimal dan menghasilkan performa validasi yang memuaskan. Oleh karena itu, kondisi

ini dapat dikategorikan sebagai *good fit*, dan model dinilai telah mencapai performa yang cukup ideal dalam konteks klasifikasi *cyberbullying*.



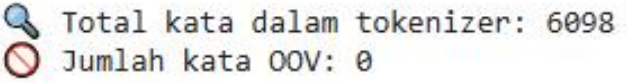
Gambar 4.20 Confusion Matrix CNN BiLSTM FastText

Berdasarkan Gambar 4.20, Evaluasi performa model secara kuantitatif yang menunjukkan keseimbangan dalam prediksi kedua kelas. Dari total 368 data uji, model berhasil mengklasifikasikan 141 dari 158 komentar *non-cyberbullying* secara benar, dan 194 dari 210 komentar *cyberbullying* dengan tepat. Jumlah *false positive* dan *false negative* yang relatif kecil dan seimbang (masing-masing 17 dan 16) menunjukkan bahwa model tidak bias terhadap salah satu kelas dan mampu mengenali komentar bermuatan *cyberbullying* dan *non-cyberbullying* dengan baik. Hal ini juga tercermin dari nilai metrik evaluasi lainnya seperti *precision*, *recall*, dan *F1-score*.



Gambar 4.21 *Classification report* CNN BiLSTM FastText

Berdasarkan *classification repor*t pada Gambar 4.21, model menghasilkan *precision* sebesar 0.9194, *recall* sebesar 0.9238, dan *F1-score* sebesar 0.9216 secara keseluruhan. Nilai-nilai tersebut mencerminkan kemampuan model dalam menjaga keseimbangan antara kesalahan positif dan negatif, yang sangat penting dalam konteks deteksi *cyberbullying*. Selain itu, nilai AUC (*Area Under Curve*) mencapai 0.9423, yang mengindikasikan bahwa model memiliki kapabilitas diskriminatif yang tinggi dalam membedakan antara komentar yang mengandung unsur *cyberbullying* dan yang tidak.

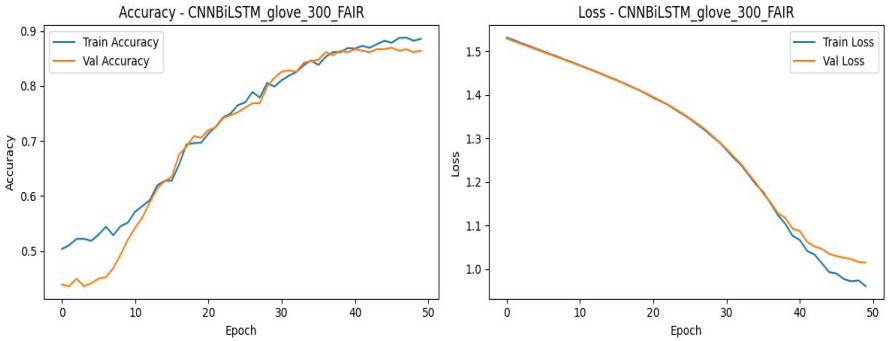


Gambar 4.22 Jumlah kata OOV CNN BiLSTM

Gambar 4.22 menunjukkan jumlah kata *Out-of-Vocabulary* (OOV) pada model CNN–BiLSTM dengan *embedding* FastText. Keberhasilan model ini juga dipengaruhi oleh kualitas representasi kata yang digunakan. Dengan jumlah kata dalam tokenizer sebanyak 6.098 dan tidak adanya kata OOV (*Out-of-Vocabulary*), FastText terbukti mampu mengatasi tantangan dalam menangani kata tidak baku, slang, atau variasi morfologis yang sering muncul dalam komentar di media sosial.

4.6.1.2 CNN – BiLSTM Word *Embedding GloVe*

Gambar 4.23 menyajikan grafik akurasi dan loss dari model CNN–BiLSTM yang menggunakan *embedding* GloVe berdimensi 300 dalam 50 epoch.



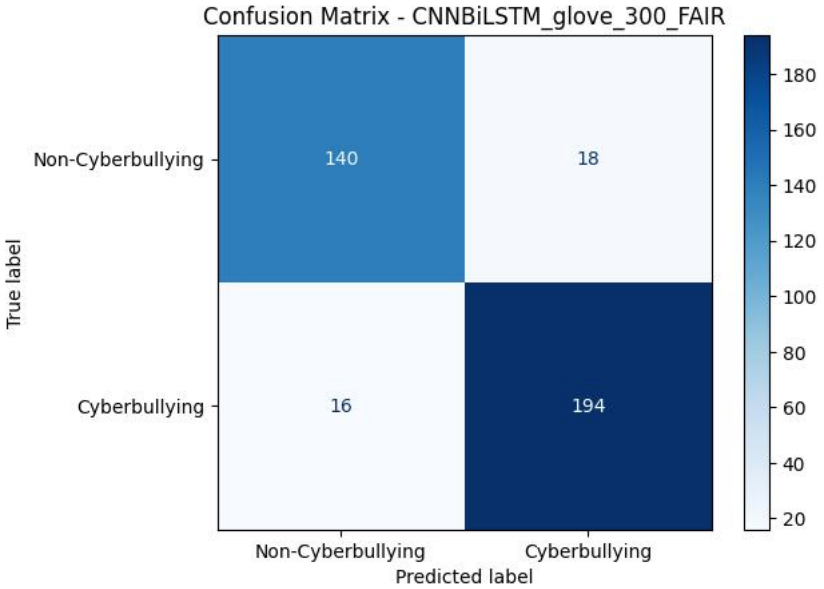
Gambar 4.23 Grafik Akurasi dan Loss Model CNN–BiLSTM dengan GloVe

Dari grafik tersebut, terlihat bahwa akurasi pada data pelatihan dan validasi meningkat secara konsisten seiring bertambahnya *epoch*. Tidak ditemukan lonjakan ekstrem ataupun penurunan signifikan, yang menunjukkan bahwa proses pelatihan berjalan stabil (*good fit*). Kurva akurasi validasi mengikuti pola kurva pelatihan dengan sangat rapat, terutama setelah memasuki *epoch* ke-20. Sementara itu, grafik *loss* menunjukkan penurunan yang konsisten dan berkelanjutan, baik pada data pelatihan maupun validasi. Kedua kurva *loss* cenderung mendekati satu sama lain dengan jarak yang tidak signifikan.

Meskipun akurasi pelatihan terus meningkat, akurasi pada data validasi mulai melambat dan stagnan sekitar *epoch* ke-35. Seiring bertambahnya *epoch*, jarak antara kurva akurasi pelatihan dan validasi semakin melebar, membentuk pola yang terbuka. Pola ini sering diasosiasikan dengan gejala *overfitting*, yaitu kondisi di mana model terlalu menyesuaikan diri dengan data pelatihan sehingga mengorbankan kemampuan untuk menggeneralisasi pada data baru.

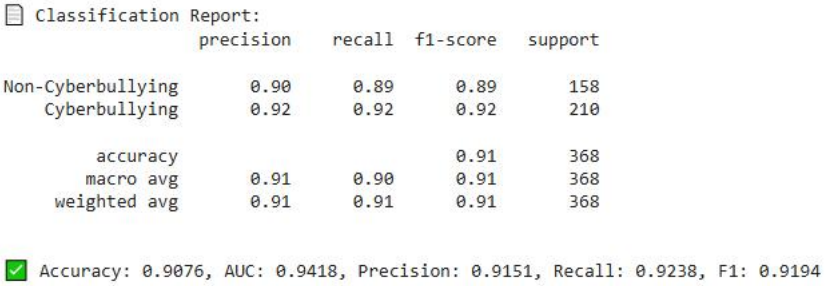
Fenomena serupa juga terlihat pada grafik *loss*. Kurva *loss* pelatihan menunjukkan penurunan yang stabil, menandakan bahwa model semakin baik dalam meminimalkan kesalahan prediksi. Namun, kurva *loss* validasi cenderung datar dan bahkan sedikit meningkat pada fase akhir

pelatihan. Ketidakseimbangan ini menunjukkan bahwa model mulai kehilangan kemampuannya untuk menggeneralisasi pada data validasi.



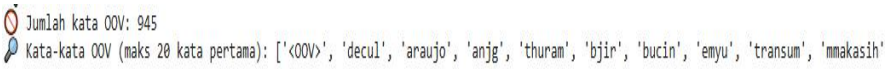
Gambar 4.24 Confusion Matrix Model CNN–BiLSTM GloVe

Jika dilihat dari Gambar 4.24, hasil prediksi model terhadap data uji menunjukkan performa yang seimbang dalam mengklasifikasikan dua kelas. Model berhasil mengenali 140 dari 158 komentar *non-cyberbullying* dengan benar, serta 194 dari 210 komentar *cyberbullying*. Hanya terdapat sedikit kesalahan klasifikasi, yaitu 18 *false positive* dan 16 *false negative*. Rasio kesalahan yang rendah ini menunjukkan bahwa model tidak hanya akurat, tetapi juga adil dalam mengenali masing-masing kelas, tanpa kecenderungan bias terhadap salah satunya.



Gambar 4.25 *Classification Report* Model CNN–BiLSTM GloVe

Laporan klasifikasi yang ditampilkan pada Gambar 4.25 memperkuat hasil evaluasi sebelumnya. Model mencatatkan *precision* sebesar 0.9151, *recall* sebesar 0.9238, dan *F1-score* sebesar 0.9194. Ketiga metrik tersebut mencerminkan kinerja model yang tangguh dan konsisten, baik dalam mengidentifikasi komentar yang mengandung unsur *cyberbullying* maupun yang tidak. Selain itu, nilai AUC sebesar 0.9418 juga memperlihatkan bahwa model memiliki kemampuan klasifikasi yang sangat baik dan dapat diandalkan untuk memisahkan dua kelas target dengan akurasi tinggi.



Gambar 4.26 Jumlah Kata OOV pada *Embedding* GloVe

Berbeda dengan *embedding* FastText, model berbasis GloVe masih menghadapi tantangan dalam menangani kata-kata tidak baku. Seperti terlihat pada Gambar 4.26, terdapat 945 kata dari total 6.098 dalam tokenizer yang tidak dikenali atau termasuk dalam kategori *Out-of-Vocabulary* (OOV). Sebagian besar kata ini merupakan bentuk slang, singkatan, atau istilah khas yang sering muncul dalam konteks media sosial, seperti "anjg", "bucin", "emyu", dan lainnya.

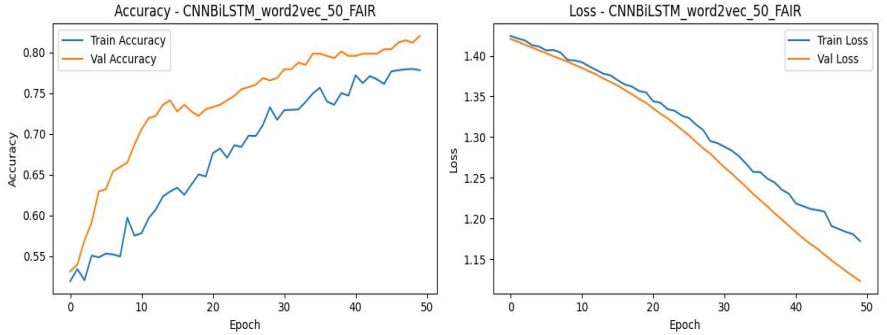
4.6.1.3 CNN – BiLSTM Word *Embedding Word2Vec*

Untuk mengamati pengaruh ukuran dimensi *embedding* terhadap performa model, CNN–BiLSTM dilatih menggunakan Word2Vec dengan empat konfigurasi dimensi berbeda, yaitu 50, 100, 200, dan 300. Pengujian variasi dimensi hanya dilakukan pada Word2Vec karena *embedding* ini dapat disesuaikan dengan mudah dalam berbagai konfigurasi dimensi.

Sementara itu, FastText dan GloVe digunakan dalam bentuk *pre-trained* dengan dimensi tetap (300), sehingga tidak memungkinkan untuk diuji dalam berbagai ukuran tanpa pelatihan ulang. Oleh karena itu, hanya Word2Vec yang dieksplorasi lebih dalam dari sisi variasi dimensi *embedding*.

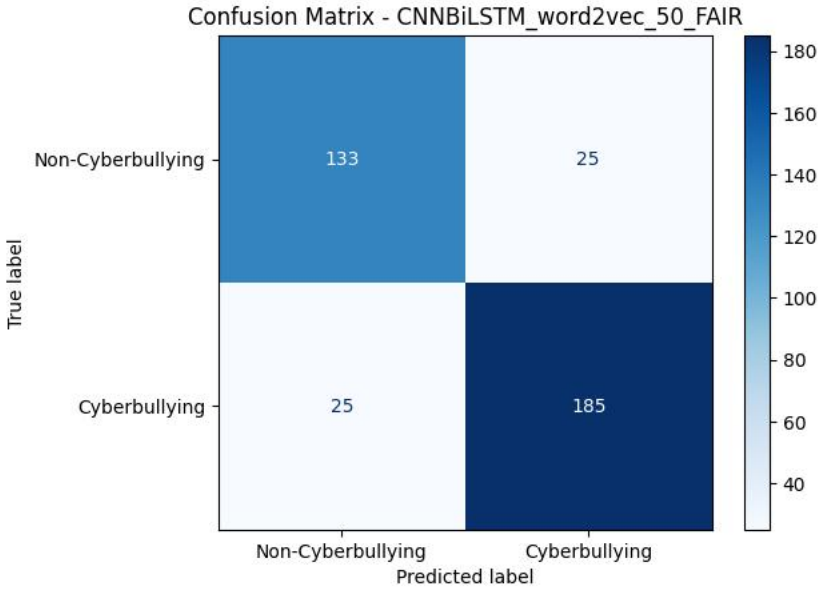
4.6.1.3.1 CNN – BiLSTM Word2Vec Dim 50

Eksperimen dengan Word2Vec berdimensi 50 pada arsitektur CNN–BiLSTM dilakukan untuk mengevaluasi kemampuan representasi kata berdimensi rendah dalam mendukung klasifikasi. Gambar 4.30 menunjukkan bagaimana model belajar dan menyesuaikan bobotnya selama 50 *epoch* pelatihan.



Gambar 4.27 Grafik Akurasi dan Loss Model CNN–BiLSTM Word2Vec 50D

Grafik akurasi dan *loss* pada Gambar 4.27 memperlihatkan *embedding* Word2Vec digunakan dengan dimensi 50. Salah satu pola menarik yang terlihat adalah bahwa akurasi pada data validasi cenderung lebih tinggi dibanding data pelatihan, terutama setelah *epoch* ke-15. Fenomena ini menunjukkan bahwa meskipun model terus belajar, kompleksitas representasi kata dari Word2Vec dimensi rendah belum cukup kuat untuk merepresentasikan konteks kalimat secara optimal pada data pelatihan. Ini dapat mengarah pada gejala *underfitting* ringan, yang menandakan bahwa model belum sepenuhnya menyerap pola dari data pelatihan secara efektif. Selain itu, perbedaan antara kurva train *loss* dan val *loss* tetap terjaga dalam rentang yang cukup rapat dan cenderung menurun hingga akhir pelatihan. Namun, meskipun secara grafis terlihat sehat, nilai akurasi validasi yang stagnan di bawah 0.80 menegaskan keterbatasan model dalam menangkap makna mendalam dari teks berbasis Word2Vec 50D.



Gambar 4.28 Confusion matrix model CNN–BiLSTM Word2Vec 50D

Beralih ke *confusion matrix* pada Gambar 4.28, distribusi prediksi menunjukkan bahwa model mampu mengenali kedua kelas dengan tingkat akurasi yang cukup berimbang. Meskipun demikian, jumlah *false negative* dan *false positive* yang sama-sama berjumlah 25 menunjukkan bahwa model belum mampu mengatasi ambiguitas semantik dengan presisi tinggi. Ini bisa disebabkan oleh lemahnya kapasitas representasional Word2Vec 50D dalam menangkap nuansa emosional atau konotatif dari kata-kata yang sering muncul dalam ujaran *cyberbullying*, seperti sarkasme, ejekan halus, atau plesetan.

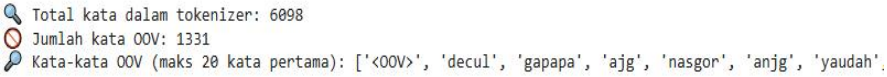
Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.84	0.84	0.84	158
Cyberbullying	0.88	0.88	0.88	210
accuracy			0.86	368
macro avg	0.86	0.86	0.86	368
weighted avg	0.86	0.86	0.86	368

Accuracy: 0.8641, AUC: 0.887, Precision: 0.881, Recall: 0.881, F1: 0.881

Gambar 4.29 *Classification Report* Model CNN–BiLSTM Word2Vec 50D

Gambar 4.29 menampilkan hasil evaluasi metrik performa model. Dengan *precision* dan *recall* masing-masing sebesar 0.84 dan 0.88, terlihat bahwa model sedikit lebih unggul dalam mengenali komentar bermuatan *cyberbullying* dibanding komentar netral atau *non-cyberbullying*. Walau ada sedikit kesalahan dalam mengenali komentar netral, hal ini bisa dimaklumi karena sistem yang lebih sensitif terhadap komentar *cyberbullying* lebih dibutuhkan dalam konteks pencegahan. Dengan *F1-score* 0.86, model menunjukkan performa yang cukup baik dan konsisten.

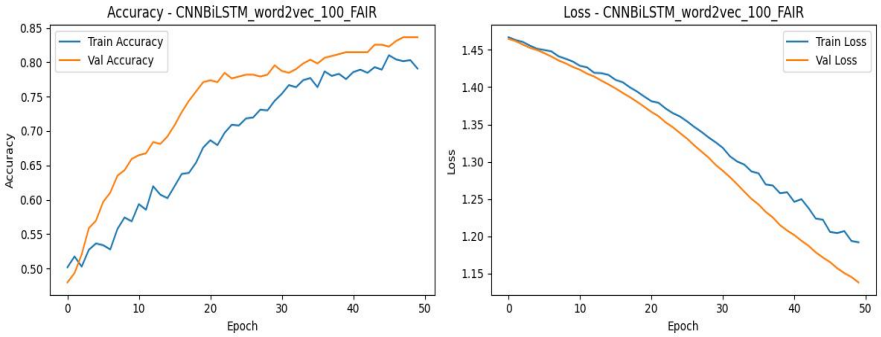


Gambar 4.30 Jumlah Kata OOV pada *Embedding* Word2Vec 50D

Dari sisi kualitas representasi kata, Gambar 4.230 menunjukkan bahwa Word2Vec dengan dimensi 50 menghasilkan 1331 kata OOV dari total 6.098 token dalam korpus. Ini berarti lebih dari 21% kata tidak dapat dikenali oleh model *embedding* dan hanya direpresentasikan sebagai token <OOV>. Kata-kata seperti “yaelah”, “anjg”, “gapapa”, dan “nasgor” merupakan contoh dari kosakata khas media sosial Indonesia yang tidak tercakup dalam korpus pelatihan Word2Vec. Ketidakhadiran kata-kata ini dalam *embedding* berdampak langsung terhadap menurunnya kemampuan model dalam menangkap intensi sebenarnya dari sebuah komentar. Akibatnya, informasi kontekstual yang penting justru hilang sejak tahap awal *embedding*, sehingga jaringan CNN–BiLSTM tidak bisa mengolah makna secara utuh.

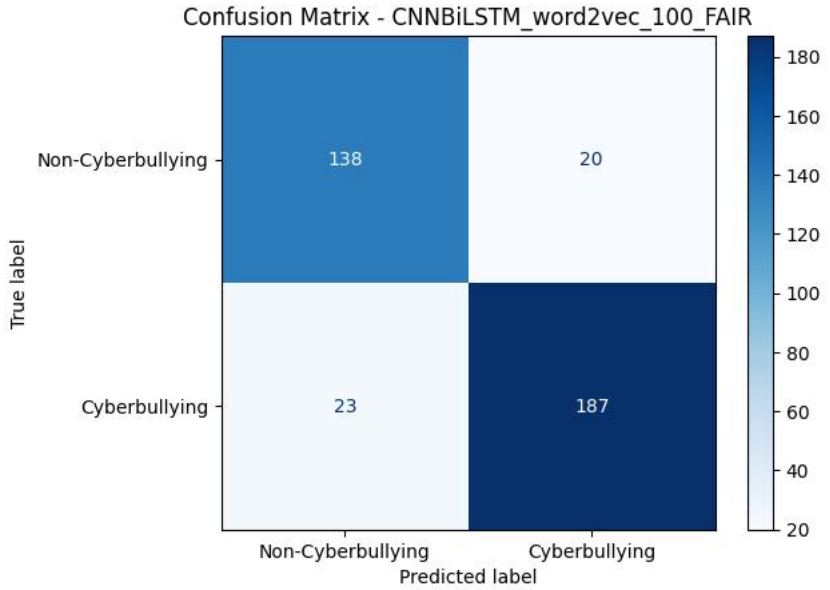
4.6.1.3.2 CNN – BiLSTM *Word2Vec* Dim 100

Eksperimen dengan Word2Vec berdimensi 100 pada arsitektur CNN–BiLSTM dilakukan untuk mengevaluasi kemampuan representasi kata.



Gambar 4.31 Grafik Akurasi dan Loss Model CNN–BiLSTM Word2Vec 100D

Gambar 4.31 menyajikan performa model CNN–BiLSTM dengan *embedding* Word2Vec berdimensi 100 selama proses pelatihan sebanyak 50 *epoch*. Dibandingkan konfigurasi berdimensi 50, model ini menunjukkan peningkatan kemampuan dalam mempelajari pola data. Berdasarkan grafik pelatihan, terlihat bahwa kurva *train accuracy* dan *validation accuracy* membentuk pola terbuka, di mana *validation accuracy* lebih tinggi daripada *train accuracy* seiring bertambahnya *epoch*. Pola serupa juga tampak pada grafik *loss*, di mana *validation loss* lebih rendah dibandingkan *train loss*. Hal ini mengindikasikan bahwa model justru lebih baik dalam memprediksi data validasi daripada data latih, yang bisa disebabkan oleh adanya regularisasi seperti *dropout* atau L2, distribusi data validasi yang lebih mudah dipelajari, atau penggunaan class weight yang seimbang.



Gambar 4.32 Confusion Matrix Model CNN–BiLSTM Word2Vec 100D

Evaluasi performa model secara kuantitatif ditunjukkan melalui *confusion matrix* pada Gambar 4.32. Dari total 368 data uji, model berhasil mengklasifikasikan 138 dari 158 komentar *non-cyberbullying* dengan benar, serta 187 dari 210 komentar *cyberbullying*. Jumlah kesalahan klasifikasi relatif seimbang, yaitu 20 *false positive* dan 23 *false negative*. Distribusi ini mengindikasikan bahwa model memiliki kemampuan klasifikasi yang cukup proporsional dalam mengenali kedua kelas target tanpa kecenderungan bias terhadap salah satu kelas.

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.86	0.87	0.87	158
Cyberbullying	0.90	0.89	0.90	210
accuracy			0.88	368
macro avg	0.88	0.88	0.88	368
weighted avg	0.88	0.88	0.88	368

Accuracy: 0.8832, AUC: 0.8932, Precision: 0.9034, Recall: 0.8905, F1: 0.8969

Gambar 4.33 Classification Report Model CNN–BiLSTM Word2Vec 100D

Sementara itu, Gambar 4.33 memperlihatkan laporan klasifikasi lengkap dari model. Dengan nilai akurasi sebesar 88,32%, *precision* sebesar 0.9034, *recall* sebesar 0.8905, dan *F1-score* sebesar 0.8969, model menunjukkan kinerja yang tinggi dan seimbang dalam mengenali komentar bermuatan *cyberbullying* maupun yang tidak. Selain itu, nilai AUC sebesar 0.8932 mengindikasikan kemampuan model dalam membedakan dua kelas secara efektif.

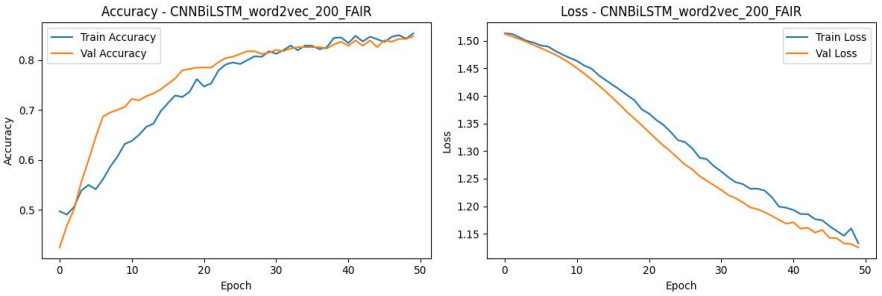
```
🔍 Total kata dalam tokenizer: 6098
🚫 Jumlah kata OOV: 1331
🔑 Kata-kata OOV (maks 20 kata pertama): ['<OOV>', 'decul', 'gapapa', 'ajg', 'nasgor', 'anjg', 'yaudah', 'bjir',
```

Gambar 4.34 Jumlah Kata OOV Word2Vec 100D

Gambar 4.34 menunjukkan bahwa jumlah kata OOV ini konsisten dengan model Word2Vec berdimensi lainnya, seperti 50D, 200D, dan 300D. Hal ini terjadi karena keempat model *embedding* tersebut dilatih menggunakan korpus yang sama, sehingga daftar kosakata yang dihasilkan pun identik. Perbedaan dimensi vektor tidak memengaruhi jumlah kosakata yang tersedia, melainkan hanya memengaruhi kedalaman atau kompleksitas representasi semantiknya. Oleh karena itu, selama kosakata yang digunakan tidak berubah, jumlah OOV akan tetap sama, dan hal ini tetap menjadi kendala dalam memahami nuansa konteks khas media sosial secara menyeluruh.

4.6.1.3.3 CNN – BiLSTM *Word2Vec* Dim 200

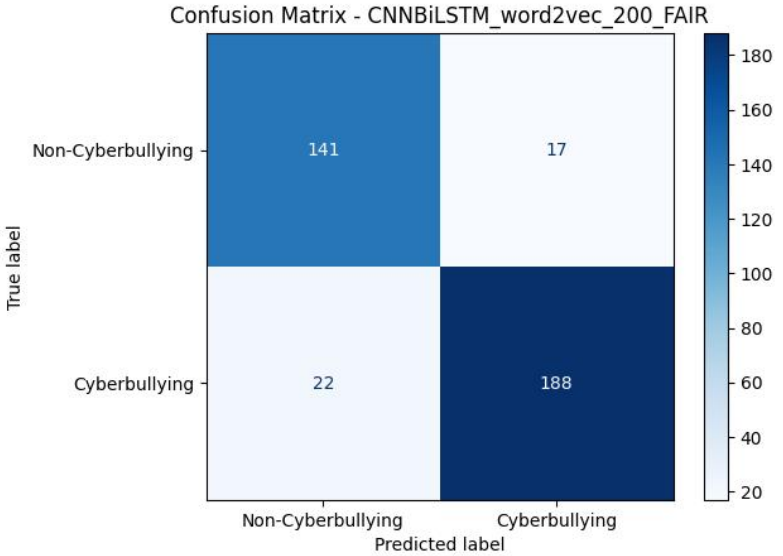
Dimensi Word2Vec ditingkatkan dari 100 menjadi 200 pada arsitektur CNN–BiLSTM untuk menguji pengaruh peningkatan kapasitas representasi terhadap kinerja model.



Gambar 4.35 Grafik Akurasi dan Loss Model CNN–BiLSTM Word2Vec 200D

Berdasarkan grafik pada Gambar 4.35, terlihat bahwa performa model CNN-BiLSTM dengan *embedding* Word2Vec 200 dimensi mengalami peningkatan yang stabil baik pada akurasi maupun penurunan nilai *loss* selama proses pelatihan selama 50 *epoch*. Grafik akurasi menunjukkan bahwa akurasi data validasi terus meningkat dan cenderung sejajar dengan akurasi data latih. Hal ini mengindikasikan bahwa model belajar secara konsisten dari data tanpa mengalami *overfitting*, karena tidak terdapat jarak signifikan antara kedua kurva. Bahkan, dalam beberapa *epoch* terakhir, akurasi validasi sedikit lebih tinggi daripada akurasi pelatihan, yang mengisyaratkan bahwa model memiliki generalisasi yang baik terhadap data yang tidak terlihat selama pelatihan.

Sementara itu, nilai *loss* baik pada data latih maupun validasi terus menurun secara bertahap tanpa lonjakan yang tidak wajar. Penurunan *loss* yang konsisten ini menunjukkan bahwa proses pembelajaran berlangsung secara efektif, dan model tidak mengalami masalah seperti *underfitting* ataupun *overfitting*. Performa yang stabil pada kedua grafik ini menunjukkan bahwa kombinasi arsitektur CNN-BiLSTM dengan *embedding* Word2Vec 200 berfungsi dengan baik dalam mengenali pola dalam data teks untuk tugas klasifikasi yang diberikan.



Gambar 4.36 Confusion Matrix Model CNN–BiLSTM Word2Vec 200D

Gambar 4.36 menampilkan *confusion matrix* hasil prediksi model pada data uji. Terlihat bahwa model berhasil mengklasifikasikan 141 dari 158 komentar *non-cyberbullying*, serta 188 dari 210 komentar *cyberbullying* dengan benar. Hanya terdapat 17 *false positive* dan 22 *false negative* angka kesalahan klasifikasi ini lebih rendah dibandingkan dengan dimensi 50 maupun 100, menandakan bahwa peningkatan dimensi *embedding* memang berdampak langsung pada penurunan jumlah kesalahan prediksi.

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.87	0.89	0.88	158
Cyberbullying	0.92	0.90	0.91	210
accuracy			0.89	368
macro avg	0.89	0.89	0.89	368
weighted avg	0.89	0.89	0.89	368

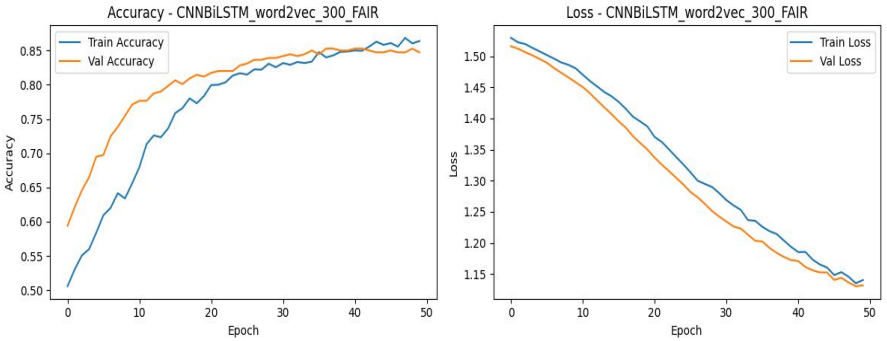
✅ Accuracy: 0.894, AUC: 0.9133, Precision: 0.9171, Recall: 0.8952, F1: 0.906

Gambar 4.37 Classification Report Model CNN–BiLSTM Word2Vec 200D

Gambar 4.37 menyajikan classification report yang memperkuat performa model secara kuantitatif. Dengan akurasi sebesar 89.4%, *precision* 91.71%, *recall* 89.52%, dan F1-score 90.6%, terlihat bahwa model tidak hanya unggul dalam satu aspek metrik, tetapi konsisten tinggi di semua metrik utama. Nilai AUC sebesar 0.9133 juga mencerminkan kemampuan diskriminatif yang baik. Ini menunjukkan bahwa representasi Word2Vec 200D mampu memberikan kejelasan kontekstual yang lebih baik kepada model dalam membedakan ujaran bernuansa agresif secara halus dengan komentar biasa. Sedangkan satistik OOV yang tetap konsisten dengan eksperimen Word2Vec lainnya, yaitu sebanyak 1.331 kata dari 6.098 token tidak dikenali oleh *embedding*. Sebagaimana telah dijelaskan sebelumnya, hal ini disebabkan karena *embedding* Word2Vec digunakan dalam versi *pre-trained*, di mana kosakata bersifat tetap dan tidak mencakup kata-kata tidak baku atau slang lokal khas media sosial Indonesia. Dengan demikian, meningkatkan dimensi tidak akan mengurangi jumlah OOV, karena hal ini lebih berkaitan dengan isi korpus pelatihan Word2Vec daripada ukuran vektornya.

4.6.1.3.4 CNN – BiLSTM *Word2Vec* Dim 300

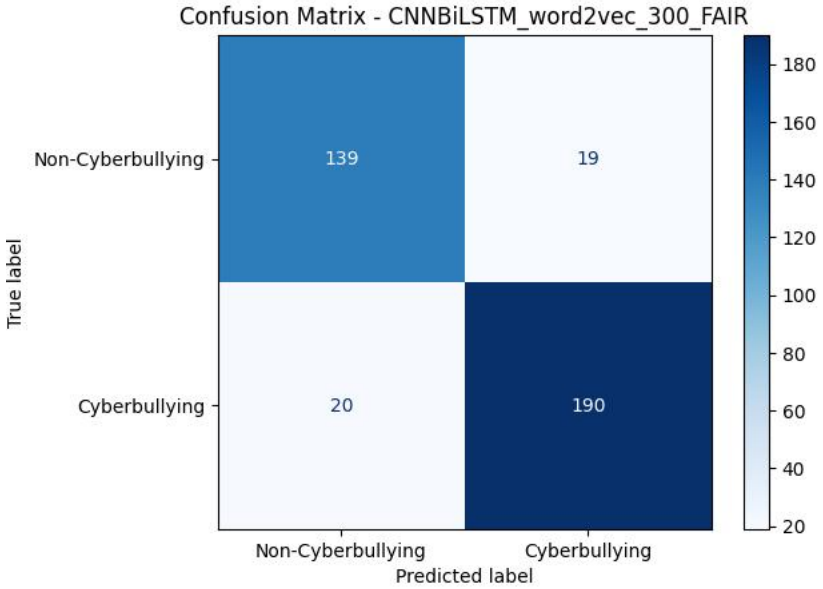
Model CNN–BiLSTM yang dikombinasikan dengan Word2Vec berdimensi 300 menunjukkan performa klasifikasi terbaik dari seluruh eksperimen CNN BiLSTM menggunakan Word2Vec yang telah dilakukan.



Gambar 4.38 Grafik Akurasi dan Loss Model CNN–BiLSTM Word2Vec 300D

Berdasarkan grafik pada Gambar 4.38, terlihat bahwa performa model CNN-BiLSTM dengan *embedding* Word2Vec 300 dimensi menunjukkan pembelajaran yang stabil selama proses pelatihan. Grafik akurasi memperlihatkan bahwa akurasi pada data latih meningkat secara bertahap, sementara akurasi pada data validasi juga mengalami kenaikan meskipun dengan laju yang lebih lambat. Meskipun terdapat sedikit jarak antara kedua kurva, perbedaan tersebut masih dalam batas yang wajar dan tidak menunjukkan adanya *overfitting* yang parah. Hal ini mengindikasikan bahwa model mampu belajar dengan cukup baik dari data latih dan menunjukkan performa yang cukup baik pada data validasi.

Sementara itu, grafik *loss* menunjukkan penurunan nilai *loss* pada data latih yang konsisten, disertai dengan penurunan loss pada data validasi yang relatif stabil. Tidak terdapat lonjakan drastis atau peningkatan tiba-tiba pada nilai loss validasi, yang menandakan bahwa proses pelatihan berlangsung dengan efektif dan model tidak mengalami kendala serius seperti underfitting maupun overfitting. Secara keseluruhan, grafik akurasi dan loss ini menggambarkan bahwa kombinasi arsitektur CNN-BiLSTM dengan *embedding* Word2Vec 300 dimensi memiliki kemampuan generalisasi yang cukup baik dalam mengenali pola teks untuk tugas klasifikasi yang diberikan.



Gambar 4.39 Confusion Matrix Model CNN–BiLSTM Word2Vec 300D

Evaluasi melalui confusion matrix pada Gambar 4.39 mengungkapkan bahwa model mampu mengidentifikasi sebagian besar teks *cyberbullying* dengan baik, disertai dengan penurunan jumlah kesalahan klasifikasi yang signifikan. Secara khusus, tingkat kesalahan pada kelas yang lebih penting secara sosial—yakni *cyberbullying*—berhasil ditekan dengan efektif. Tidak hanya itu, capaian nilai AUC sebesar 0.9222 menandai kemampuan model untuk memisahkan dua kelas dengan presisi tinggi, menandakan bahwa model tidak hanya tepat secara numerik tetapi juga andal dalam menangani ambiguitas bahasa alami.

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.87	0.88	0.88	158
Cyberbullying	0.91	0.90	0.91	210
accuracy			0.89	368
macro avg	0.89	0.89	0.89	368
weighted avg	0.89	0.89	0.89	368

Accuracy: 0.894, AUC: 0.9222, Precision: 0.9091, Recall: 0.9048, F1: 0.9069

Gambar 4.40 Confusion Matrix Model CNN–BiLSTM Word2Vec 300D

Pencapaian metrik lainnya ditunjukkan pada Gambar 4.40, seperti *precision* sebesar 91%, *recall* 90%, dan F1-score mendekati 91%, memperkuat posisi model ini sebagai representasi terbaik dalam eksperimen. Konsistensi di seluruh metrik memperlihatkan bahwa performa tidak hanya ditopang oleh satu aspek evaluasi saja, melainkan merupakan hasil dari keseimbangan antara kemampuan mengenali *cyberbullying* dan menghindari kesalahan prediksi pada kelas *non-cyberbullying*. Meski demikian, seperti telah disampaikan sebelumnya, jumlah kata OOV tetap bertahan pada angka 1331 dari total 6.098 token. Ini kembali menegaskan bahwa peningkatan dimensi tidak serta-merta memperluas kosakata yang dikenal *embedding*, karena keterbatasan ini bersumber dari isi korpus pelatihan Word2Vec pre-trained, bukan dari kapasitas dimensinya.

Tabel 4.3 di bawah ini menyajikan rekapitulasi hasil eksperimen model CNN–BiLSTM dengan penerapan tiga jenis *word embedding*.

Tabel 4.3 Perbandingan CNN BiLSTM dengan penerapan tiga word *embedding*

Model	**Dim**	**Validation Accuracy**	**AUC**	***Precision***	***Recall***	**F1-Score**	**Training Time (detik)**	**Jumlah kata OOV**
CNN BiLSTM FastText	**300**	**0.9130**	**0.9410**	**0.9238**	**0.9238**	**0.9238**	55	**0**
CNN BiLSTM Word2Vec	300	0.9049	0.9248	0.9187	0.9143	0.9165	62	1331
CNN BiLSTM GloVe	300	0.9022	0.9377	0.9223	0.9048	0.9135	67	945
CNN BiLSTM Word2Vec	100	0.8750	0.9064	0.8868	0.8952	0.8910	**50**	1331

Model	Dim	Validation Accuracy	AUC	*Precision*	*Recall*	F1-Score	Training Time (detik)	Jumlah kata OOV
CNN BiLSTM Word2Vec	200	0.8750	0.9033	0.9184	0.8571	0.8867	52	1331
CNN BiLSTM Word2Vec	50	0.8424	0.8860	0.8918	0.8238	0.8564	**50**	1331

Hasil eksperimen yang disajikan dalam Tabel 4.3 memperlihatkan bahwa kombinasi model CNN–BiLSTM dengan *word embedding* FastText berdimensi 300 menunjukkan performa terbaik secara konsisten di seluruh metrik evaluasi. Model ini mencapai nilai *validation accuracy* sebesar 0.9130 dan *F1-Score* sebesar 0.9238, menjadikannya sebagai konfigurasi yang paling unggul dalam mendeteksi ujaran *cyberbullying*. Keunggulan FastText dapat dijelaskan oleh kemampuannya dalam menangani kata-kata *out-of-vocabulary* (OOV) dan kata tidak baku melalui pendekatan *subword information*, yang menjadikannya sangat relevan pada korpus yang mengandung bahasa informal seperti komentar media sosial berbahasa Indonesia. Meskipun grafik awal model FastText sempat menunjukkan indikasi *overfitting*—ditandai dengan jarak (gap) antara kurva akurasi dan *loss* pada data latih dan validasi—namun secara keseluruhan, hasil metrik evaluasi menunjukkan bahwa model tetap mampu melakukan generalisasi dengan baik. Hal ini mengindikasikan bahwa *overfitting* yang terjadi masih terkendali dan tidak signifikan, sehingga tidak menurunkan performa model pada data validasi secara drastis.

Dalam perbandingan dengan Word2Vec dan GloVe pada dimensi yang sama (300), FastText menunjukkan keunggulan yang tipis namun signifikan. Word2Vec menempati posisi kedua dengan akurasi validasi sebesar 0.9049 dan F1-score 0.9165, sedangkan GloVe mengikuti dengan akurasi 0.9022 dan F1-score 0.9135. Kinerja Word2Vec menunjukkan keseimbangan yang baik antara *precision* (0.9187) dan *recall* (0.9143). Di sisi lain, GloVe menunjukkan kecenderungan yang sedikit lebih tinggi pada *precision* dibandingkan *recall*, yang mengindikasikan bahwa model ini

lebih berhati-hati dalam mengklasifikasikan data sebagai *cyberbullying*. Dari perspektif *Area Under Curve* (AUC), FastText tetap unggul dengan nilai 0.9410, yang menegaskan bahwa model ini memiliki kemampuan prediktif yang tinggi di seluruh rentang threshold klasifikasi.

Analisis lebih lanjut terhadap Word2Vec dengan berbagai dimensi menunjukkan pola peningkatan performa seiring dengan bertambahnya ukuran vektor. Word2Vec dengan dimensi 50 menghasilkan *F1-Score* terendah (0.8564) dan mengalami peningkatan bertahap pada dimensi 100 (0.8910) dan 200 (0.8867), hingga mencapai nilai tertinggi pada dimensi 300 (0.9165). Hal ini mencerminkan bahwa vektor dengan dimensi 300 mampu merepresentasikan makna semantik kata dengan lebih kompleks dan kaya, sehingga menghasilkan representasi input yang lebih informatif untuk model CNN–BiLSTM. Namun, peningkatan dari dimensi 200 ke 300 menunjukkan bahwa tidak semua metrik meningkat secara linier; misalnya, *precision* pada dimensi 200 (0.9184) lebih tinggi dibandingkan dengan dimensi 300 (0.9187), sementara *recall* pada dimensi 200 justru lebih rendah (0.8571), yang berdampak pada *F1-Score* secara keseluruhan.

Analisis *trade-off* antara *precision* dan *recall* menjadi penting, terutama pada Word2Vec dengan dimensi 200. Meskipun *precision*-nya sangat tinggi, *recall* yang lebih rendah menunjukkan bahwa model ini cenderung menghindari kesalahan *false positive* dengan mengorbankan klasifikasi kasus *cyberbullying* yang sebenarnya ada. Sebaliknya, FastText 300 dan Word2Vec 300 menunjukkan keseimbangan yang ideal antara *precision* dan *recall* (keduanya 0.9238 dan 0.9143), menjadikannya pilihan yang lebih stabil untuk diterapkan, khususnya dalam konteks moderasi konten daring yang memerlukan kepekaan terhadap kesalahan klasifikasi baik positif maupun negatif.

Lebih lanjut, keunggulan FastText dapat dijelaskan dari perspektif lingkungan pelatihan *embedding*. Model ini dibangun di atas korpus multibahasa yang besar, yang mencakup struktur morfologis dan karakter sub-kata, sehingga lebih tangguh dalam memahami kata-kata tidak baku, serapan, atau slang yang umum dalam komentar media sosial berbahasa Indonesia. Sementara itu, *embedding* Word2Vec dan GloVe umumnya dilatih pada korpus bahasa Inggris formal (seperti Google News atau Wikipedia), yang memiliki keterbatasan dalam menangkap nuansa kata dalam bahasa Indonesia yang tidak baku, kecuali jika dilakukan pelatihan ulang dengan korpus lokal. Oleh karena itu, tanpa adanya proses *fine-*

tuning pada *embedding* tersebut, keunggulan FastText menjadi sangat wajar.

Dari sisi waktu pelatihan model (*running time*) atau durasi pelatihan model mengalami perbedaan yang cukup mencolok antara masing-masing jenis dan dimensi word *embedding* yang digunakan. Model CNN-BiLSTM dengan FastText berukuran 300 dimensi berhasil menunjukkan performa evaluasi terbaik sekaligus membutuhkan waktu pelatihan paling singkat, yaitu 0.55 atau 55 detik, jika dibandingkan dengan model lain yang menggunakan Word2Vec maupun GloVe berdimensi sama. Di sisi lain, model yang memanfaatkan GloVe dengan dimensi 300 justru memerlukan waktu pelatihan terlama, yakni 67 detik , walaupun tidak menghasilkan performa lebih unggul dari FastText. Selain itu, peningkatan dimensi *embedding* pada Word2Vec (dari 50D ke 100D hingga 300D) tampak berdampak pada peningkatan durasi pelatihan, meskipun tidak selalu diikuti dengan peningkatan performa model secara sebanding. Temuan ini mengindikasikan bahwa efisiensi pelatihan tidak hanya dipengaruhi oleh ukuran dimensi *embedding*, tetapi juga oleh struktur dan karakteristik representasi kata dari masing-masing metode *embedding*. Oleh karena itu, pemilihan jenis *embedding* perlu mempertimbangkan keseimbangan antara efektivitas performa dan efisiensi waktu pelatihan.

Dari segi jumlah kata yang tidak dikenali (OOV), terlihat bahwa hal ini memiliki dampak signifikan terhadap kinerja model. Model yang menggunakan *embedding* FastText menunjukkan keunggulan yang jelas dengan menghasilkan jumlah kata OOV sebesar 0, berkat kemampuannya dalam menangani kata-kata yang tidak secara eksplisit terdapat dalam kamus *embedding* melalui pemrosesan sub-kata. Sebaliknya, Word2Vec dan GloVe menunjukkan hasil yang berbeda, di mana model dengan *embedding* Word2Vec pada semua dimensi (50, 100, 200, 300) mencatatkan jumlah OOV tertinggi, yaitu 1331 kata, sementara GloVe memiliki 945 kata OOV. Tingginya jumlah OOV ini menunjukkan bahwa banyak kata dalam korpus pelatihan atau pengujian tidak dikenali oleh *embedding*, sehingga tidak memiliki representasi vektor, yang dapat berdampak negatif pada pemahaman konteks oleh model.

4.6.2 Evaluasi dan Analisis Performa Model Lainnya

Subbab ini menyajikan evaluasi dan analisis performa dari beberapa model pembanding yang telah diterapkan, yaitu BiLSTM, CNN, LSTM, dan CNN–LSTM. Setiap model juga diuji menggunakan 3 jenis

word embedding yang sama, namun dalam bagian ini hanya ditampilkan hasil terbaik dari masing-masing arsitektur, yaitu model dengan kombinasi *embedding* yang menghasilkan nilai *F1-score* tertinggi sesuai dengan evaluasi model CNN-BiLSTM sebelumnya.

4.6.2.1 BiLSTM

Tabel 4.4 menyajikan ringkasan hasil eksperimen model BiLSTM menggunakan tiga jenis *word embedding* yang berbeda, yakni FastText, GloVe, dan Word2Vec dengan berbagai dimensi vektor.

Tabel 4.4 Hasil eksperimen BiLSTM dengan penerapan *word embedding*

Model	Dim	Validation Accuracy	AUC	*Precision*	*Recall*	F1-Score	Training Time (detik)	Jumlah kata OOV
BiLSTM FastText	**300**	**0.9130**	0.9276	**0.9113**	0.8810	0.8959	55	**0**
BiLSTM GloVe	300	0.8995	**0.9307**	0.9078	**0.8905**	**0.8990**	55	954
BiLSTM Word2Vec	300	0.8859	0.9152	0.9010	0.8667	0.8835	55	1331
BiLSTM Word2Vec	100	0.8696	0.9022	0.8844	0.8381	0.8606	53	1331
BiLSTM Word2Vec	200	0.8668	0.8975	0.8900	0.8476	0.8683	54	1331
BiLSTM Word2Vec	50	0.8451	0.8700	0.8866	0.8190	0.8515	**51**	1331

Berdasarkan hasil evaluasi terhadap model BiLSTM dengan berbagai jenis dan dimensi *embedding*, terlihat bahwa performa antar model bervariasi tergantung pada metrik yang digunakan. FastText dengan dimensi 300 menunjukkan keunggulan pada metrik akurasi (0.9130), *precision* (0.9113), dan F1-score (0.8959), serta memiliki waktu pelatihan yang efisien, yaitu sekitar 55 detik. Namun, model dengan GloVe 300D justru memiliki nilai AUC tertinggi (0.9307), yang berarti model ini paling mampu membedakan antara kelas *cyberbullying* dan *non cyberbullying*

secara menyeluruh. Selain itu, *recall* tertinggi juga diperoleh oleh GloVe (0.8905), yang menandakan bahwa model ini lebih baik dalam menangkap kasus-kasus *cyberbullying* meskipun itu berarti ada potensi sedikit lebih banyak *false positive*.

Model BiLSTM dengan GloVe 300D juga menunjukkan performa yang kompetitif dengan *F1-score* tertinggi sebesar 0.8990, meskipun akurasi validasi (0.8995) sedikit lebih rendah dibanding FastText. GloVe yang berbasis statistik global *co-occurrence* memberikan representasi kata yang stabil dan andal, terutama pada kata-kata umum yang sering muncul dalam korpus besar. Keunggulan *F1-score* ini kemungkinan besar disumbang oleh *recall* yang tinggi (0.8905), menunjukkan bahwa GloVe cukup efektif dalam mengidentifikasi *cyberbullying*, meskipun kemampuannya dalam menghadapi kata OOV masih kalah dibanding FastText.

Sementara itu, eksperimen dengan Word2Vec menunjukkan performa yang cenderung lebih rendah dibanding FastText dan GloVe, namun tetap memperlihatkan tren yang konsisten. Pada dimensi 300, Word2Vec mencatat *F1-score* sebesar 0.8835, namun dengan akurasi dan AUC yang lebih rendah. Hal ini menunjukkan bahwa meskipun model mampu mendeteksi pola-pola lokal melalui konteks sempit (CBOW atau Skip-gram), Word2Vec tidak memiliki fleksibilitas dalam menangani kata tidak dikenal, karena tidak mengandalkan komponen subword atau statistik global. Pada dimensi yang lebih rendah seperti 100, 50, dan 200, performa model semakin menurun. Misalnya, pada dimensi 50, *F1-score* turun hingga 0.8515 dan *recall* menjadi hanya 0.8190.

Secara umum, dimensi *embedding* yang lebih tinggi berpotensi memberikan representasi semantik yang lebih kaya. Namun, hasil eksperimen menunjukkan bahwa peningkatan dimensi tidak selalu berbanding lurus dengan performa model. Misalnya, model dengan Word2Vec berdimensi 200 justru mencatat *F1-score* lebih tinggi daripada versi 100D, menunjukkan bahwa dimensi yang lebih besar dapat membantu hingga titik tertentu, tetapi dampaknya juga dipengaruhi oleh konteks data, arsitektur model, dan distribusi fitur dalam *embedding*. Visualisasi hasil model BiLSTM berupa grafik akurasi dan *loss* selama proses pelatihan, *confusion matrix*, serta *classification report* dapat dilihat secara lengkap pada Lampiran 2 hingga Lampiran 4.

Dari segi *training time*, seluruh model BiLSTM dengan *embedding* FastText, GloVe, dan Word2Vec berdimensi 300 menunjukkan durasi pelatihan yang serupa, yakni 0.55 menit (55 detik), sehingga tidak terdapat perbedaan signifikan dalam efisiensi waktu pelatihan pada dimensi tersebut. Sementara itu, Word2Vec dengan dimensi lebih rendah (200D, 100D, dan 50D) menunjukkan efisiensi waktu yang sedikit lebih baik dengan durasi pelatihan masing-masing 0.54, 0.53, dan 0.51 menit, yang mencerminkan beban komputasi lebih ringan. Secara keseluruhan, semua model tergolong efisien dari segi waktu pelatihan, sehingga pemilihan *embedding* sebaiknya lebih mempertimbangkan performa klasifikasi dibandingkan hanya berfokus pada lamanya pelatihan.

Seperti halnya pada model sebelumnya yaitu CNN-BiLSTM, jumlah kata yang tidak dikenali (OOV) pada model-model lain juga menunjukkan pola yang konsisten. *Embedding* FastText kembali menunjukkan keunggulannya dengan jumlah OOV sebesar 0, sementara GloVe mencatatkan 945 kata OOV, dan Word2Vec memiliki jumlah OOV tertinggi, yaitu 1.331 kata. Perbedaan ini menegaskan bahwa karakteristik masing-masing *embedding* terhadap kata-kata yang tidak terdapat dalam korpus tetap sama meskipun digunakan pada model yang berbeda.

4.6.2.2 CNN

Tabel 4.5 menyajikan ringkasan hasil eksperimen model CNN menggunakan tiga jenis *word embedding* yang berbeda, yakni FastText, GloVe, dan Word2Vec dengan berbagai dimensi vektor.

Tabel 4.5 Hasil eksperimen CNN dengan penerapan *word embedding*

Model	Dim	Validation Accuracy	AUC	*Precision*	*Recall*	F1-Score	Training Time (detik)	Jumlah kata OOV
CNN FastText	**300**	**0.9076**	**0.9260**	**0.9154**	**0.8762**	**0.8954**	35	**0**
CNN GloVe	300	0.8696	0.9213	0.9128	0.8476	0.8790	31	945
CNN Word2Vec	200	0.8071	0.8626	0.8594	0.7857	0.8209	30	1331
CNN	300	0.8043	0.8589	0.8743	0.7286	0.7948	34	1331

Model	Dim	Validation Accuracy	AUC	*Precision*	*Recall*	F1-Score	Training Time (detik)	Jumlah kata OOV
Word2Vec								
CNN Word2Vec	50	0.7283	0.7713	0.7148	0.8714	0.7854	29	1331
CNN Word2Vec	100	0.7418	0.8220	0.7696	0.7952	0.7822	29	1331

Berdasarkan hasil eksperimen pada Tabel 4.5 terhadap model CNN menggunakan tiga jenis *word embedding* yaitu FastText, GloVe, dan Word2Vec dapat disimpulkan bahwa *embedding* FastText dengan dimensi 300 menunjukkan performa terbaik secara keseluruhan. Model ini menghasilkan nilai akurasi tertinggi sebesar 0.9076, AUC sebesar 0.9260, *precision* sebesar 0.9154, *recall* sebesar 0.8762, dan F1-score sebesar 0.8954. Kinerja ini secara konsisten melampaui semua kombinasi model *embedding* lainnya, termasuk GloVe dan Word2Vec, baik dari sisi akurasi maupun keseimbangan metrik evaluasi. Selain itu, waktu pelatihan model FastText 300D juga relatif efisien, yaitu sekitar 0.35 menit (31 detik).

Model CNN dengan GloVe 300D berada pada posisi kedua dengan *F1-score* sebesar 0.8790. Sementara itu, model CNN dengan Word2Vec, meskipun menggunakan berbagai dimensi (50D, 100D, 200D, 300D), secara umum memiliki performa yang lebih rendah dibandingkan FastText dan GloVe. Model dengan Word2Vec 300D misalnya, hanya menghasilkan *F1-score* sebesar 0.7948, bahkan lebih rendah dari versi Word2Vec 200D yang mencapai 0.8327.

Hal yang patut dicermati adalah bahwa pada Word2Vec, dimensi 200 justru mengungguli 300 dalam hampir seluruh metrik, termasuk *F1-score* dan akurasi. Meskipun secara teori *embedding* berdimensi lebih tinggi seperti 300D seharusnya mampu menangkap makna semantik dengan lebih baik, fenomena penurunan performa ini dapat dijelaskan melalui kemungkinan *overfitting*. Peningkatan jumlah parameter pada dimensi tinggi dapat menyebabkan model terlalu menyesuaikan diri dengan data pelatihan, sehingga kehilangan kemampuan generalisasi pada data validasi. Temuan ini sejalan dengan penelitian Mikolov et al. (2013) dan

Zhang et al. (2020), yang menekankan pentingnya keseimbangan antara ukuran dimensi *embedding* dengan kecocokan konteks pelatihan terhadap tugas spesifik seperti klasifikasi teks.

Dari sisi efisiensi, Word2Vec dengan dimensi 50D dan 100D memiliki waktu pelatihan tercepat, masing-masing hanya 0.29 menit (29 detik). Meskipun cepat, namun performanya tidak sebanding dengan model FastText. Oleh karena itu, meskipun waktu pelatihan merupakan aspek penting, kompromi antara performa dan efisiensi perlu dipertimbangkan, terutama ketika akurasi klasifikasi menjadi prioritas utama.

4.6.2.3 LSTM

Tabel 4.6 menyajikan ringkasan hasil eksperimen model LSTM menggunakan tiga jenis *word embedding* yang berbeda, yakni FastText, GloVe, dan Word2Vec dengan berbagai dimensi vektor.

Tabel 4.6 Hasil eksperimen LSTM dengan penerapan *word embedding*

Model	Dim	Validation Accuracy	AUC	*Precision*	*Recall*	F1-Score	Training Time (detik)	Jumlah kata OOV
LSTM FastText	**300**	**0.9130**	**0.9250**	0.9101	0.8190	0.8622	38	**0**
LSTM GloVe	300	0.9022	0.9242	0.9137	**0.8571**	**0.8845**	38	945
LSTM Word2Vec	300	0.8750	0.9191	0.9124	0.8429	0.8762	38	1331
LSTM Word2Vec	200	0.8777	0.9092	**0.9157**	0.7762	0.8402	38	1331
LSTM Word2Vec	100	0.8016	0.7776	0.7585	0.8524	0.8027	45	1331
LSTM Word2Vec	50	0.7446	0.7585	0.7723	0.8238	0.7972	36	1331

Evaluasi terhadap model LSTM yang dikombinasikan dengan *embedding* FastText, GloVe, dan Word2Vec menunjukkan adanya variasi performa yang cukup mencolok pada sejumlah metrik evaluasi, yakni

akurasi, AUC, *precision*, *recall*, F1-score, durasi pelatihan, serta jumlah kata yang tidak dikenali (Out-of-Vocabulary/OOV). Dari segi akurasi, *embedding* FastText dengan dimensi 300 menunjukkan kinerja tertinggi dengan nilai sebesar 0.9130. Di posisi berikutnya, GloVe memperoleh skor 0.9022, diikuti oleh Word2Vec berdimensi 300 dengan nilai 0.8750. Namun demikian, performa Word2Vec cenderung menurun seiring dengan berkurangnya dimensi vektornya. Hal ini terlihat pada model berdimensi 200, 100, dan 50 yang masing-masing mencatatkan akurasi sebesar 0.8777, 0.8016, dan 0.7446. Temuan ini mengindikasikan bahwa pada Word2Vec, peningkatan dimensi *embedding* dapat memperbaiki akurasi model, meskipun tetap belum mampu mengungguli FastText maupun GloVe.

Untuk metrik AUC *(Area Under Curve)*, nilai tertinggi diraih oleh FastText sebesar 0.9250, diikuti secara sangat dekat oleh GloVe dengan skor 0.9242, dan Word2Vec 300 dengan nilai 0.9191. Perbedaan AUC di antara ketiga *embedding* pada dimensi 300 tergolong kecil, yang mengisyaratkan bahwa ketiganya memiliki kapabilitas yang hampir setara dalam membedakan kelas target, meskipun FastText sedikit lebih unggul dalam hal ini. Beralih ke metrik *precision*, model dengan Word2Vec berdimensi 200 menunjukkan capaian tertinggi sebesar 0.9157. Disusul oleh Word2Vec 300 (0.9124) dan GloVe (0.9137), sementara FastText mencatat *precision* sedikit lebih rendah, yaitu 0.9101. Meskipun selisihnya tidak signifikan, hasil ini menunjukkan bahwa Word2Vec cenderung memberikan rasio prediksi positif yang benar lebih tinggi pada skenario tertentu dibandingkan *embedding* lainnya.

Namun demikian, pada aspek *recall*, FastText (0.8190) justru mencatatkan nilai yang lebih rendah dibandingkan GloVe (0.8571) dan Word2Vec 300 (0.8429). Bahkan, Word2Vec berdimensi lebih rendah seperti 100 dan 50 pun memperoleh *recall* lebih tinggi, masing-masing sebesar 0.8524 dan 0.8238. Artinya, meskipun FastText memiliki akurasi tinggi, kemampuannya dalam mengidentifikasi sampel positif secara menyeluruh masih kalah dibandingkan *embedding* lain. Sementara itu, pada metrik *F1-score*—yang merupakan rata-rata harmonis dari *precision* dan *recall*—GloVe menunjukkan kinerja paling optimal dengan nilai 0.8845, diikuti oleh Word2Vec 300 (0.8762) dan FastText (0.8622). Fakta ini mengindikasikan bahwa GloVe berhasil menjaga keseimbangan antara *precision* dan *recall* dalam konteks arsitektur LSTM, meskipun FastText unggul dalam akurasi dan AUC. Dengan demikian, GloVe dapat dianggap sebagai pilihan *embedding* yang lebih stabil secara keseluruhan.

Dari segi efisiensi pelatihan, seluruh model menunjukkan durasi pelatihan yang relatif setara, berkisar antara 36 detik hingga 45 detik. Ini menunjukkan bahwa perbedaan jenis dan dimensi *embedding* tidak terlalu berdampak terhadap waktu pelatihan dalam arsitektur LSTM. Namun, Word2Vec berdimensi 100 justru membutuhkan waktu pelatihan terlama (45 detik), meskipun performanya berada di posisi terbawah. Hal ini menimbulkan pertimbangan efisiensi yang perlu diperhatikan dalam pemilihan *embedding*.

Terakhir, dalam hal jumlah kata yang tidak dikenali (OOV), *embedding* FastText mencatatkan keunggulan absolut dengan jumlah OOV sebanyak 0. Hal ini dimungkinkan karena pendekatan berbasis sub-kata yang diadopsi oleh FastText, memungkinkan model mengenali kata-kata yang tidak ada secara eksplisit dalam kamus *embedding*. Sebaliknya, GloVe dan Word2Vec mencatat jumlah OOV masing-masing sebesar 945 dan 1331 kata. Menariknya, meskipun Word2Vec memiliki jumlah OOV tertinggi, model ini tetap mampu mencapai nilai *precision* dan *recall* yang kompetitif pada dimensi tertentu. Kendati demikian, jumlah OOV yang besar tetap menjadi faktor yang dapat mengurangi efektivitas pemahaman konteks semantik dalam pemrosesan teks.

4.6.2.4 CNN – LSTM

Tabel 4.7 menyajikan ringkasan hasil eksperimen model CNN menggunakan tiga jenis *word embedding* yang berbeda, yakni FastText, GloVe, dan Word2Vec dengan berbagai dimensi vektor.

Tabel 4.7 Hasil eksperimen CNN - LSTM dengan penerapan *word embedding*

Model	Dim	Validation Accuracy	AUC	*Precision*	*Recall*	F1-Score	Training Time (detik)	Jumlah kata OOV
CNN LSTM FastText	**300**	**0.9076**	**0.9312**	0.9038	**0.8952**	**0.8995**	33	**0**
CNN LSTM GloVe	300	0.8886	0.9308	**0.9219**	0.8429	0.8806	32	945
CNN LSTM Word2Vec	300	0.8804	0.8860	0.9037	0.8048	0.8514	34	1331

Model	Dim	Validation Accuracy	AUC	*Precision*	*Recall*	F1-Score	Training Time (detik)	Jumlah kata OOV
CNN LSTM Word2Vec	100	0.8424	0.8393	0.8947	0.8095	0.8500	**29**	1331
CNN LSTM Word2Vec	200	0.8696	0.8784	0.8889	0.8000	0.8421	30	1331
CNN LSTM Word2Vec	50	0.8397	0.8353	0.8827	0.7524	0.8123	**29**	1331

Secara keseluruhan sari hasil yang diperoleh pada Tabel 4.7, model CNN–LSTM dengan *embedding* FastText 300D menunjukkan kinerja paling seimbang di antara seluruh varian yang diuji. Model ini memperoleh akurasi validasi tertinggi sebesar 0.9076 dan juga mencatat F1-score tertinggi sebesar 0.8995, yang menandakan keseimbangan yang baik antara *precision* dan *recall*. Namun, keunggulan FastText tidak berlaku mutlak pada seluruh metrik. Misalnya, meskipun AUC-nya (0.9312) merupakan yang tertinggi, selisihnya sangat kecil jika dibandingkan dengan GloVe 300D (0.9308), sehingga dari sudut pandang statistik, perbedaan ini kemungkinan tidak signifikan.

Pada sisi *precision***,** model dengan *embedding* GloVe 300D mencatat nilai tertinggi (0.9219). Ini menunjukkan bahwa GloVe memiliki kecenderungan lebih baik dalam meminimalkan kesalahan klasifikasi positif palsu (false positive). Akan tetapi, model ini mencatat *recall* yang lebih rendah (0.8429) dibanding FastText (0.8952), mengindikasikan bahwa GloVe lebih konservatif dalam mendeteksi kasus *cyberbullying* dan berpotensi melewatkan sejumlah kasus aktual (false negative). Kendati demikian, F1-score GloVe (0.8806) tetap kompetitif, menandakan bahwa performanya cukup solid dalam konteks klasifikasi yang seimbang.

Sementara itu, *embedding* Word2Vec 300D menunjukkan performa yang secara umum lebih rendah dibandingkan FastText dan GloVe. Dengan akurasi 0.8804, *recall* 0.8048, dan F1-score 0.8514, model ini terlihat lebih berhati-hati dalam mengidentifikasi kelas positif, yang tercermin dari *precision*-nya yang masih tinggi (0.9037), namun dengan *trade-off* terhadap *recall* yang lebih rendah. Strategi ini dapat berdampak pada pengurangan

deteksi kasus cyberbullying, yang dalam konteks aplikasi nyata bisa menjadi kelemahan.

Jika dibandingkan berdasarkan variasi dimensi *embedding* Word2Vec, terlihat penurunan performa seiring penurunan dimensi vektor. Model dengan Word2Vec 200D dan 100D masing-masing memiliki F1-score 0.8421 dan 0.8500, yang tidak berbeda jauh dari versi 300D (0.8514). Akan tetapi, saat dimensi diturunkan menjadi 50D, F1-score turun lebih signifikan ke angka 0.8123, dengan *recall* paling rendah (0.7524) di antara semua varian. Hal ini mengindikasikan bahwa *embedding* berdimensi rendah kurang mampu menangkap informasi semantik secara mendalam, sehingga berdampak negatif terhadap kinerja model.

Dari sisi waktu pelatihan, semua model menunjukkan durasi yang relatif serupa, berada dalam kisaran 29–34 detik, dan tidak menunjukkan adanya dominasi tertentu dalam hal efisiensi pelatihan. Bahkan, model dengan *embedding* berdimensi rendah seperti Word2Vec 50D dan 100D tidak memperlihatkan keunggulan yang berarti dari sisi waktu, karena perbedaannya sangat kecil dan tidak sebanding dengan penurunan performa yang terjadi.

Dalam hal jumlah kata OOV (*out-of-vocabulary*), model dengan *embedding* FastText kembali mencatat angka terbaik, yaitu 0 OOV, berkat pendekatannya yang memanfaatkan sub-kata. Sebaliknya, GloVe dan Word2Vec mencatat masing-masing 945 dan 1331 kata OOV, yang menunjukkan adanya keterbatasan dalam representasi kata-kata yang tidak dikenal dalam data pelatihan. Walaupun demikian, GloVe tampaknya mampu mempertahankan performa yang cukup baik meskipun memiliki jumlah OOV yang tidak sedikit, sementara Word2Vec tampak lebih terpengaruh oleh keterbatasan ini, terutama dalam metrik *recall* dan F1-score.

4.6.3 Perbandingan Akhir dan Pemilihan Model Terbaik

Tabel 4.8 menyajikan ringkasan hasil eksperimen semua model yaitu: CNN-BiLSTM, BiLSTM, LSTM, CNN, CNN-LSTM menggunakan tiga jenis *word embedding*, yakni FastText, GloVe, dan Word2Vec dengan berbagai dimensi vektor dan merangkum metrik performa utama, yaitu *validation accuracy*, *AUC (Area Under the Curve)*, *precision*, *recall*, dan *F1-score* untuk setiap konfigurasi model.

Dalam konteks klasifikasi komentar yang mengandung *cyberbullying*, pemilihan model terbaik tidak hanya bergantung pada akurasi validasi. Masalah ketidakseimbangan kelas yang umum dalam data sosial media menyebabkan akurasi menjadi kurang representatif. Oleh karena itu, *F1-score* dipilih sebagai metrik utama dalam evaluasi karena mencerminkan keseimbangan antara *precision* dan *recall*, yang krusial dalam menghindari baik *false positive* maupun *false negative*. Selain itu, metrik AUC digunakan untuk mengukur kemampuan model dalam membedakan antara dua kelas secara keseluruhan, tanpa bergantung pada ambang klasifikasi tertentu.

Tabel 4.8 Hasil eksperimen semua model dengan penerapan *word embedding*

Model	*Embedding*	Dim	Validation Accuracy	AUC	*Precision*	*Recall*	F1-Score	Training Time (detik)	Jumlah kata OOV
CNN BiLSTM	**FastText**	**300**	**0.9130**	**0.9410**	**0.9238**	**0.9238**	**0.9238**	55	**0**
CNN BiLSTM	Word2Vec	300	0.9049	0.9248	0.9187	0.9143	0.9165	62	1331
CNN BiLSTM	GloVe	300	0.9022	0.9377	0.9223	0.9048	0.9135	67	945
CNN LSTM	FastText	300	0.9076	0.9312	0.9038	0.8952	0.8995	33	**0**
BiLSTM	GloVe	300	0.8995	0.9307	0.9078	0.8905	0.8990	55	945
BiLSTM	FastText	300	**0.9130**	0.9276	0.9113	0.8810	0.8959	55	**0**
CNN	FastText	300	0.9076	0.9260	0.9154	0.8762	0.8954	35	**0**
CNN BiLSTM	Word2Vec	100	0.8750	0.9064	0.8868	0.8952	0.8910	50	1331
CNN BiLSTM	Word2Vec	200	0.8750	0.9033	0.9184	0.8571	0.8867	52	1331
LSTM	GloVe	300	0.9022	0.9242	0.9137	0.8571	0.8845	38	945
BiLSTM	Word2Vec	300	0.8859	0.9152	0.9010	0.8667	0.8835	55	1331
CNN LSTM	GloVe	300	0.8886	0.9308	0.9219	0.8429	0.8806	32	945
CNN	GloVe	300	0.8696	0.9213	0.9128	0.8476	0.8790	31	945

Model	*Embedding*	Dim	Validation Accuracy	AUC	*Precision*	*Recall*	F1-Score	Training Time (detik)	Jumlah kata OOV
LSTM	Word2Vec	300	0.8750	0.9191	0.9124	0.8429	0.8762	38	1331
BiLSTM	Word2Vec	200	0.8668	0.8975	0.8900	0.8476	0.8683	54	1331
LSTM	FastText	300	**0.9130**	0.9250	0.9101	0.8190	0.8622	38	**0**
BiLSTM	Word2Vec	100	0.8696	0.9022	0.8844	0.8381	0.8606	53	1331
CNN BiLSTM	Word2Vec	50	0.8424	0.8860	0.8918	0.8238	0.8564	50	1331
BiLSTM	Word2Vec	50	0.8451	0.8700	0.8866	0.8190	0.8515	51	1331
CNN LSTM	Word2Vec	300	0.8804	0.8860	0.9037	0.8048	0.8514	34	1331
CNN LSTM	Word2Vec	100	0.8424	0.8393	0.8947	0.8095	0.8500	**29**	1331
CNN LSTM	Word2Vec	200	0.8696	0.8784	0.8889	0.8000	0.8421	30	1331
LSTM	Word2Vec	200	0.8777	0.9092	0.9157	0.7762	0.8402	38	1331
CNN	Word2Vec	200	0.8071	0.8626	0.8594	0.7857	0.8209	30	1331
CNN LSTM	Word2Vec	50	0.8397	0.8353	0.8827	0.7524	0.8123	**29**	1331
LSTM	Word2Vec	100	0.8016	0.7776	0.7585	0.8524	0.8027	45	1331
LSTM	Word2Vec	50	0.7446	0.7585	0.7723	0.8238	0.7972	36	1331
CNN	Word2Vec	300	0.8043	0.8589	0.8743	0.7286	0.7948	34	1331
CNN	Word2Vec	50	0.7283	0.7713	0.7148	0.8714	0.7854	**29**	1331
CNN	Word2Vec	100	0.7418	0.8220	0.7696	0.7952	0.7822	**29**	1331

Berdasarkan data pada Tabel 4.8, nilai *Validation Accuracy* tertinggi diraih oleh tiga model sekaligus, yaitu CNN–BiLSTM FastText, BiLSTM FastText, dan LSTM FastText, yang masing-masing mencatatkan skor sebesar 0.9130. Ketiganya menggunakan *embedding* FastText berdimensi 300, yang secara desain unggul dalam mengatasi kata-kata tidak umum berkat pendekatannya yang berbasis sub-kata (*subword*). Keunggulan ini memungkinkan model mengenali kata-kata baru, bentuk

tidak baku, atau kesalahan ejaan yang tidak ditemukan dalam korpus—suatu kemampuan penting dalam konteks deteksi *cyberbullying* yang kerap melibatkan bahasa informal. Di sisi lain, model dengan akurasi paling rendah adalah CNN Word2Vec 50D dengan nilai 0.7283. Rendahnya akurasi ini dapat ditelusuri ke dua faktor utama: dimensi *embedding* yang terlalu kecil (50D), yang membatasi kemampuan representasi makna kata, serta jumlah OOV yang tinggi (1331), sehingga banyak informasi penting tidak diwakili dalam proses pelatihan.

Selanjutnya, nilai *AUC (Area Under Curve)* tertinggi sebesar 0.9410 juga diraih oleh CNN–BiLSTM FastText, mempertegas kapabilitas model ini dalam membedakan antara teks *cyberbullying* dan *non cyberbullying*. Nilai AUC yang tinggi mencerminkan bahwa model mampu menjaga keseimbangan optimal antara sensitivitas dan spesifisitas, bahkan ketika ambang batas klasifikasi berubah. Sebaliknya, model dengan nilai AUC terendah adalah LSTM Word2Vec 50D (0.7585), yang menunjukkan kelemahan dalam membedakan dua kelas tersebut. Hal ini diduga disebabkan oleh rendahnya kualitas representasi semantik dari *embedding* berdimensi kecil serta keterbatasan arsitektur LSTM tunggal dalam menangkap konteks dan pola spasial sekuensial, dibandingkan dengan model gabungan seperti CNN–BiLSTM.

Dari segi *precision*, CNN–BiLSTM FastText kembali menjadi yang tertinggi dengan skor 0.9238. Ini berarti model tersebut mampu mengurangi jumlah *false positive*, atau dengan kata lain, tidak mudah salah menandai teks yang bukan *cyberbullying* sebagai *cyberbullying*. Dalam sistem moderasi konten, ketepatan ini sangat penting untuk mencegah terjadinya penindakan berlebihan terhadap teks netral. Sebaliknya, CNN Word2Vec 50D mencatat *precision* terendah sebesar 0.7148, yang menunjukkan kecenderungan model dalam membuat kesalahan klasifikasi positif palsu. Hal ini besar kemungkinan disebabkan oleh *embedding* yang kurang kuat dan arsitektur CNN murni yang tidak memiliki kemampuan menangkap konteks jangka panjang sebagaimana BiLSTM.

Sementara itu, r*ecall* tertinggi juga diraih oleh CNN–BiLSTM FastText, dengan skor 0.9238, menandakan kemampuannya yang sangat baik dalam mendeteksi sebanyak mungkin teks yang memang mengandung unsur *cyberbullying* (minim *false negative*). Ini menjadi faktor krusial untuk sistem yang bertujuan melindungi pengguna dari potensi risiko. Di sisi lain, CNN Word2Vec 300D memperoleh nilai *recall* terendah (0.7286),

meskipun menggunakan *embedding* berdimensi besar. Hal ini mengindikasikan bahwa struktur CNN murni yang digunakan tidak cukup efektif dalam menangkap urutan konteks, yang sangat diperlukan dalam interpretasi ujaran *cyberbullying*.

Dari aspek *F1-Score* yang merupakan rata-rata dari *precision* dan *recall*, model CNN–BiLSTM FastText kembali mencatatkan performa tertinggi dengan nilai 0.9238. Ini menunjukkan keseimbangan luar biasa antara ketepatan dalam melakukan klasifikasi kasus *cyberbullying* dan kemampuan menangkap sebanyak mungkin kasus tersebut. Sebaliknya, nilai F1 terendah ditemukan pada CNN Word2Vec 100D (0.7822), yang menempatkannya di posisi lemah antara *embedding* dengan kapasitas terbatas dan struktur CNN yang kurang mampu memahami konteks jangka panjang. Meskipun dimensinya sedikit lebih besar dari 50D, performa tetap tidak optimal karena keterbatasan arsitektural.

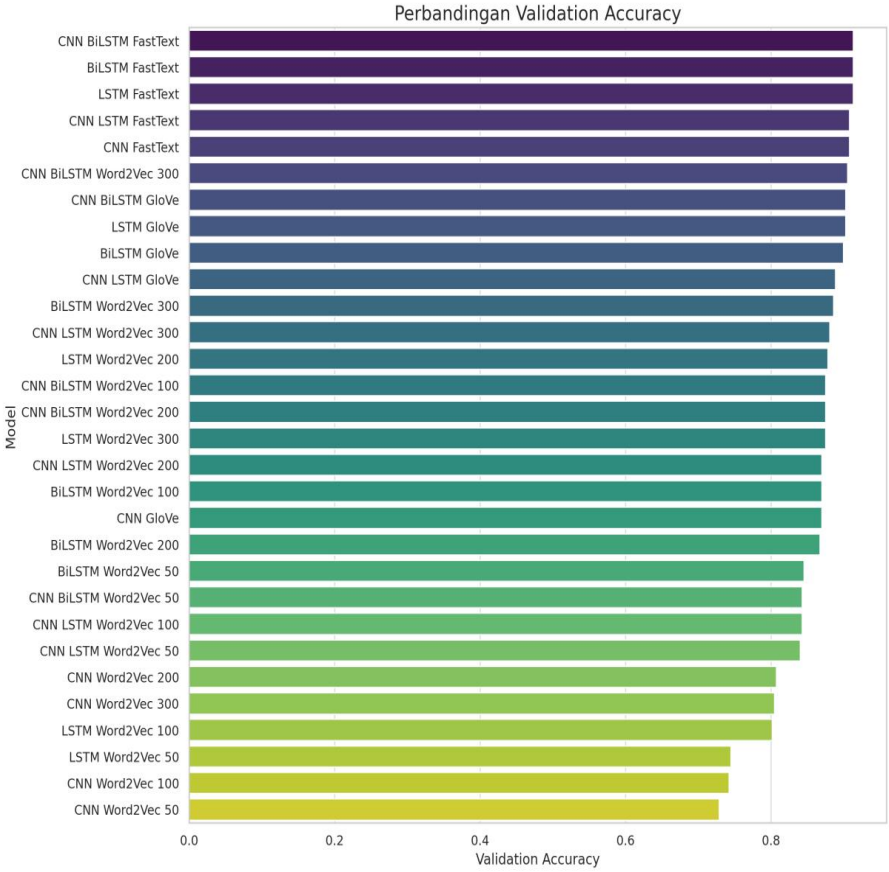
Dalam hal waktu pelatihan *(training time)*, model yang paling lambat adalah CNN–BiLSTM GloVe, dengan durasi 67 detik. Hal ini dapat dipahami karena kombinasi kompleksitas arsitektur dua arah (BiLSTM) dengan CNN serta embedding GloVe yang padat, membutuhkan waktu lebih lama untuk menyelesaikan proses pelatihan. Sebaliknya, waktu pelatihan tercepat (29 detik) diperoleh oleh sejumlah model CNN dan CNN–LSTM berbasis Word2Vec dengan dimensi 50D dan 100D. Kecepatan ini disebabkan oleh kecilnya ukuran embedding dan struktur jaringan yang relatif ringan. Namun, perlu dicatat bahwa kecepatan ini tidak berbanding lurus dengan kualitas performa, karena model-model tersebut cenderung mencatat hasil evaluasi yang rendah.

Terakhir, untuk parameter jumlah kata yan tidak di kenali (OOV), semua model yang menggunakan FastText mencatat nilai OOV 0 (tidak ada), menjadikannya unggul dalam aspek ini. FastText, yang bekerja dengan pendekatan sub-kata, mampu mengenali bentuk kata baru atau tidak standar yang lazim muncul dalam bahasa informal seperti di media sosial. Sebaliknya, seluruh model berbasis Word2Vec mencatat OOV tertinggi, yakni 1331 kata. Jumlah ini sangat berdampak pada akurasi representasi teks, karena kata-kata yang tidak dikenal tidak dapat diproses dengan tepat. GloVe sendiri mencatat OOV sebanyak 945, lebih rendah dibandingkan Word2Vec, dan performanya terlihat lebih stabil, kemungkinan karena cakupan korpus pelatihannya yang lebih luas atau lebih seimbang.

Secara keseluruhan, model CNN–BiLSTM dengan embedding FastText 300D terbukti menjadi pilihan terbaik dalam penelitian ini, dengan performa unggul di hampir seluruh metrik evaluasi, mulai dari akurasi, *AUC, precision, recall*, hingga *F1-score*. Keunggulan FastText dalam menangani OOV juga menjadi faktor krusial dalam meningkatkan keandalan model, terutama dalam menghadapi data teks informal yang kerap muncul dalam kasus *cyberbullying*. Sementara GloVe menawarkan *precision* yang tinggi dan performa stabil, serta Word2Vec cenderung tertinggal di sebagian besar aspek, pilihan model dan *embedding* terbaik pada akhirnya harus disesuaikan dengan prioritas sistem: apakah ingin memaksimalkan deteksi kasus *cyberbullying* (*recall*), menghindari kesalahan tuduhan (*precision*), atau mencapai keseimbangan keduanya secara efisien seperti yang ditunjukkan oleh FastText.

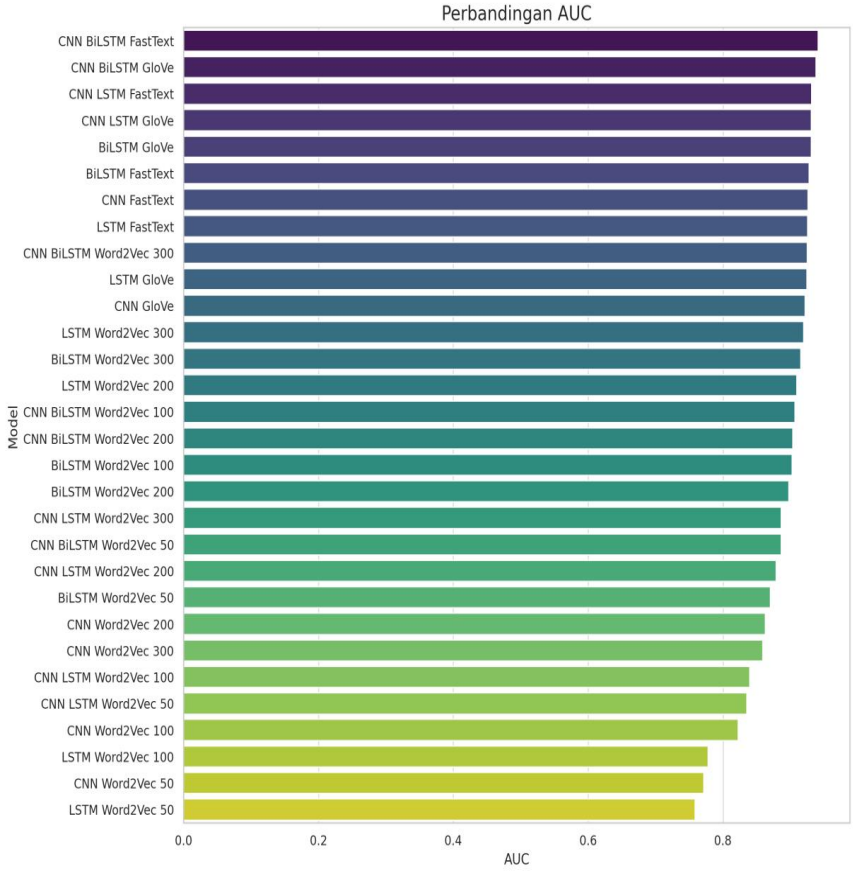
Secara teoritis, keunggulan model CNN–BiLSTM dengan FastText dapat dijelaskan melalui kelebihan arsitektur dan teknik *embedding* yang digunakan. Arsitektur CNN–BiLSTM menggabungkan kekuatan *Convolutional Neural Network* (CNN) dalam mengekstraksi fitur spasial (misalnya frasa kasar atau ujaran kebencian), dengan kekuatan *Bidirectional Long Short-Term Memory* (BiLSTM) yang efektif dalam memahami konteks sekuensial dua arah dari teks (Graves & Schmidhuber, 2005; Zhang et al., 2015). Kombinasi ini memungkinkan model untuk memahami makna lokal dan global dalam satu kesatuan, yang sangat krusial dalam analisis komentar sosial media yang sering bersifat tidak eksplisit dan kontekstual.

Untuk memberikan pemahaman yang lebih komprehensif terhadap performa masing-masing model, Gambar 4.40 menyajikan visualisasi perbandingan nilai *validation accuracy* dari seluruh model yang telah dievaluasi. Visualisasi ini bertujuan untuk memperjelas perbedaan performa antar model dan mempermudah analisis terhadap model dengan akurasi validasi terbaik.



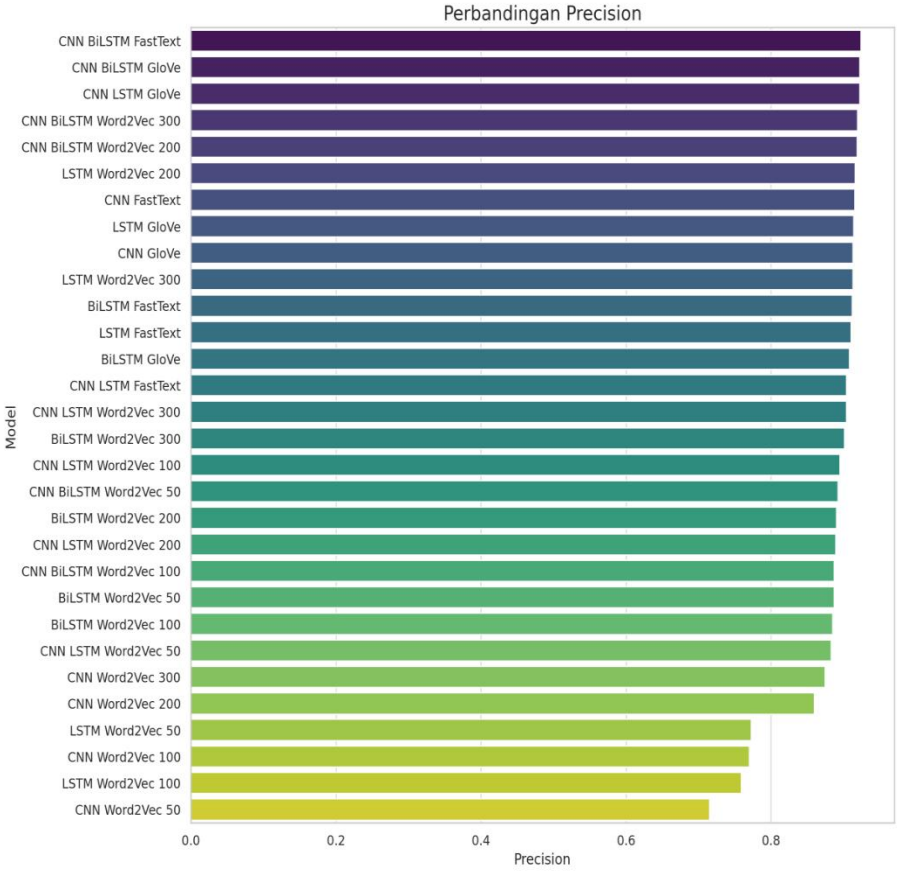
Gambar 4.41 Grafik perbandingan *validation accuracy* tiap model

Gambar 4.41 menyajikan grafik perbandingan nilai AUC dari setiap model. Metrik AUC digunakan untuk mengevaluasi kemampuan model dalam membedakan antara kelas *cyberbullying* dan *non-cyberbullying* secara menyeluruh, tanpa terpengaruh oleh threshold tertentu. Semakin tinggi nilai AUC, semakin baik kemampuan klasifikasi model terhadap kedua kelas tersebut.



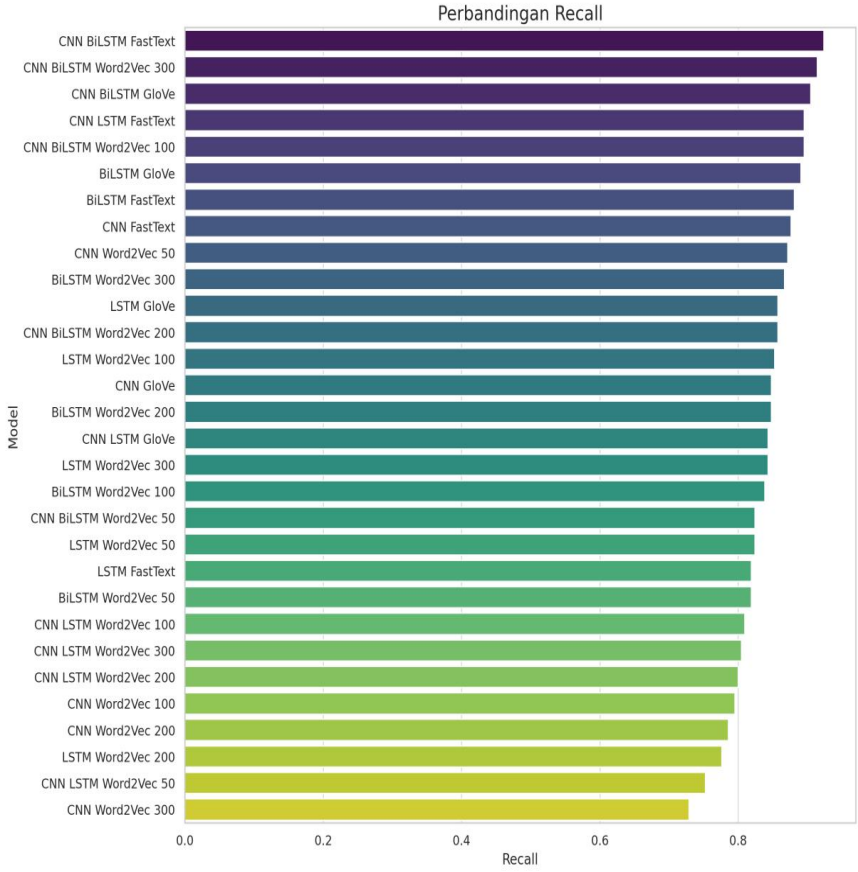
Gambar 4.42 Grafik perbandingan AUC tiap model

Selanjutnya, Gambar 4.42 menampilkan grafik perbandingan nilai *precision* dari semua model. *Precision* menunjukkan proporsi komentar yang benar-benar merupakan *cyberbullying* dari seluruh komentar yang diprediksi sebagai *cyberbullying* oleh model. Nilai *precision* yang tinggi mengindikasikan rendahnya jumlah *false positive*.



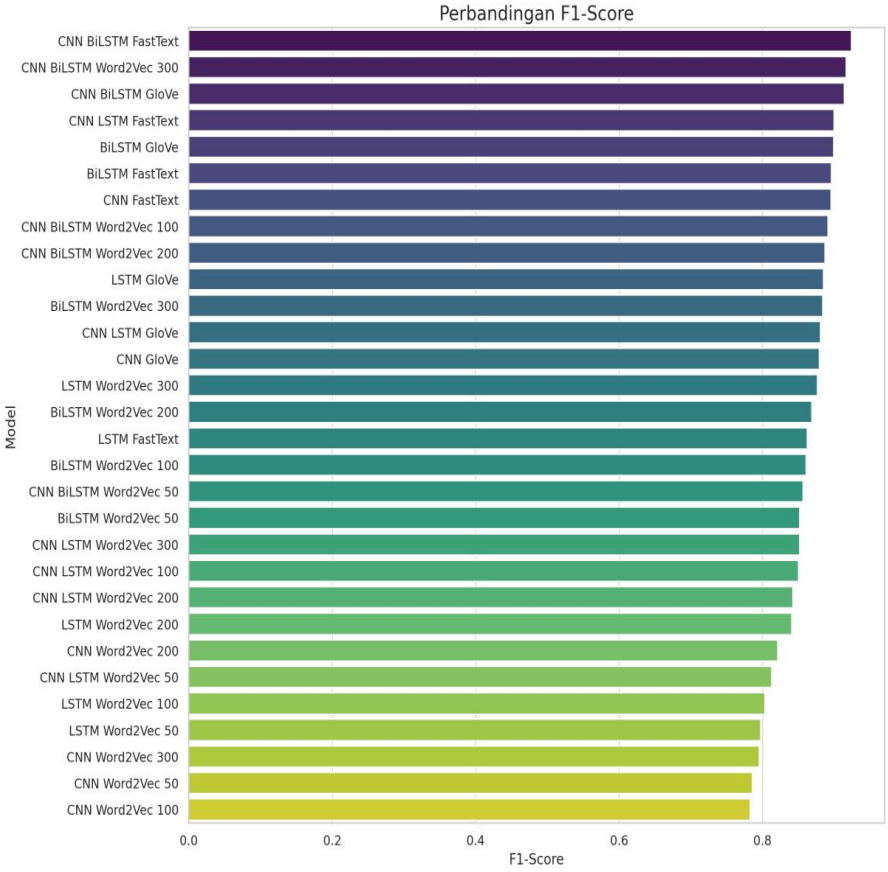
Gambar 4.43 Grafik perbandingan *precision* tiap model

Gambar 4.43 memperlihatkan nilai *recall* dari setiap model. *Recall* mengukur seberapa banyak komentar *cyberbullying* yang berhasil dikenali dengan benar oleh model dari seluruh komentar yang memang tergolong dalam kelas *cyberbullying*. Nilai *recall* yang tinggi menunjukkan kemampuan model dalam meminimalkan kesalahan klasifikasi pada komentar yang seharusnya terdeteksi.



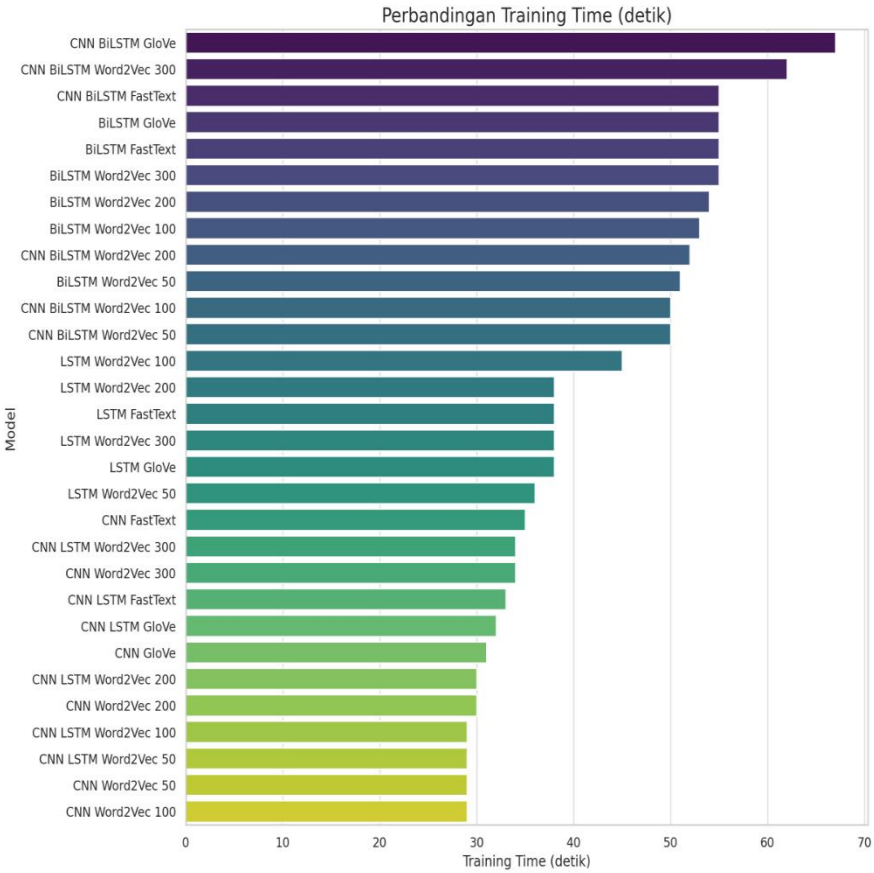
Gambar 4.44 Grafik perbandingan *recall* tiap model

Selanjutnya, Gambar 4.44 menggambarkan nilai *F1-score* dari masing-masing model, yaitu metrik harmonis antara *precision* dan *recall*. *F1-score* sangat berguna dalam kasus klasifikasi tidak seimbang karena memberikan penilaian yang mempertimbangkan kedua metrik tersebut secara seimbang.



Gambar 4.45 Grafik perbandingan *F1-Score* tiap model

Terakhir, Gambar 4.45 menampilkan grafik perbandingan waktu pelatihan dari setiap model, yang diukur dalam satuan detik. Informasi ini penting dalam mempertimbangkan efisiensi dan kelayakan model untuk diimplementasikan secara nyata, terutama ketika model akan digunakan dalam sistem yang membutuhkan respons cepat.



Gambar 4.46 Grafik perbandingan *training time* tiap model

Terakhir, untuk menilai efektivitas representasi teks yang digunakan, Gambar 4.46 menyajikan perbandingan jumlah kata *out-of-vocabulary* (OOV) pada masing-masing model. Semakin sedikit kata OOV yang ditemukan, maka semakin baik representasi embedding dalam mencakup kosakata pada data pelatihan dan pengujian, yang secara tidak langsung dapat memengaruhi performa klasifikasi.

Tabel 4.9 Rekapitulasi rata-rata performa berdasarkan jenis *word embedding*

Word *Embedding*	**Validation Accuracy**	**AUC**	***Precision***	***Recall***	**F1-Score**
FastText	**0.911**	**0.930**	**0.913**	**0.879**	**0.895**
GloVe	0.892	0.929	0.916	0.869	0.891
Word2Vec	0.839	0.866	0.866	0.825	0.843

Selain perbandingan berdasarkan arsitektur model, penting juga untuk mengevaluasi kontribusi masing-masing jenis *word embedding* terhadap performa keseluruhan. Sebagaimana ditampilkan pada Gambar 4.39 dan dirangkum pada Tabel 4.9, *embedding* FastText menunjukkan performa terbaik secara konsisten pada seluruh metrik utama, dengan nilai *validation accuracy* sebesar 0.911, *AUC* 0.930, dan *F1-score* tertinggi 0.895. Hal ini kembali menegaskan keunggulan FastText yang telah dijelaskan sebelumnya, khususnya dalam menangani kata-kata tidak umum dan variasi morfologis melalui pendekatan berbasis sub-kata (*subword*).

Di sisi lain, GloVe menempati urutan kedua dengan nilai *F1-score* sebesar 0.891, hanya sedikit di bawah FastText. Meski tidak memiliki mekanisme untuk menangani kata OOV seperti FastText, GloVe mampu memodelkan hubungan semantik global antarkata dengan cukup baik melalui pendekatan *co-occurrence matrix*. Hal ini menjadikannya tetap kompetitif, terutama dalam kalimat yang strukturnya lebih rapi atau formal.

Sebaliknya, Word2Vec menunjukkan performa paling rendah di antara ketiga *embedding*, dengan *F1-score* sebesar 0.843 dan *AUC* hanya 0.866. Meskipun Word2Vec efektif dalam merepresentasikan kata berdasarkan konteks lokal, kelemahannya dalam menangani kata-kata baru serta ketiadaan informasi karakter menyebabkan penurunan akurasi terutama pada domain sosial media yang sangat bervariasi. Penurunan performa Word2Vec ini juga telah tergambarkan pada evaluasi masing-masing model sebelumnya.

Analisis ini semakin memperkuat temuan bahwa pilihan *embedding* memainkan peran yang sangat krusial dalam meningkatkan efektivitas model klasifikasi teks. FastText dapat dianggap sebagai *embedding* yang paling adaptif dan representatif untuk tugas klasifikasi *cyberbullying* dalam konteks bahasa Indonesia informal yang digunakan di media sosial.

4.7 Perbandingan Hasil Penelitian dengan Penelitian Terdahulu

Penelitian ini menghasilkan model CNN–BiLSTM yang menggunakan *embedding* FastText dengan dimensi 300, dan model ini menunjukkan kinerja terbaik dalam klasifikasi *cyberbullying*. Model tersebut mencatat akurasi sebesar 91.30%, AUC 0.9410, serta nilai *precision* dan *recall* masing-masing 0.9238, dan F1-score sebesar 0.9238. Untuk mengevaluasi keunggulan model ini secara lebih menyeluruh, peneliti membandingkan hasilnya dengan beberapa penelitian terdahulu pada Sub Bab 2.2 Penelitian Terkait.

Penelitian pertama oleh Aqilla et al., (2023) menggunakan CNN–BiLSTM dengan Word2Vec CBOW berdimensi 300, menghasilkan akurasi 87% dan F1-score 88%. Nilai ini lebih rendah dibandingkan dengan model yang dikembangkan dalam penelitian ini. Perbedaan kinerja ini mungkin disebabkan oleh pendekatan *embedding* yang berbeda: Word2Vec yang berbasis kata utuh cenderung menghasilkan banyak kata yang tidak terdaftar (OOV) dan kurang mampu menangani variasi kata, sedangkan FastText yang berbasis *subword* dapat mengenali bentuk kata tidak baku yang sering muncul dalam teks *cyberbullying*.

Berikutnya, penelitian oleh Muslikh et al., (2024) yang menggunakan model CNN–BiLSTM dengan FastText berukuran vektor 200 pada data terjemahan Al-Qur'an menghasilkan F1-score sebesar 73.3%, yang lebih tinggi dibandingkan model tanpa FastText yang hanya mencapai 68.7%. Meskipun konteks dan data yang digunakan sangat berbeda, hasil ini memperkuat efektivitas FastText dalam bahasa Indonesia. Dalam penelitian ini, penggunaan FastText dengan dimensi 300 memberikan kinerja yang jauh lebih baik, dengan F1-score 92.38%, yang kemungkinan besar dipengaruhi oleh konteks media sosial dan optimasi *hyperparameter* model.

Pada penelitian selanjutnya yang dilakukan oleh Rhanoui et al., (2019). yang berjudul "*A CNN-BiLSTM Model for Document-Level Sentiment Analysis*" melaporkan bahwa model CNN–BiLSTM dengan *embedding* Doc2Vec mencapai akurasi 90.66%. Meskipun angka ini cukup

tinggi, model yang diusulkan dalam penelitian ini berhasil melampaui akurasi tersebut dengan nilai 91.30%. Hal ini menunjukkan bahwa penggunaan FastText, yang berbasis *subword*, lebih efektif dalam menangani teks informal seperti komentar di media sosial dibandingkan dengan Doc2Vec yang berbasis representasi dokumen.

Dalam penelitian oleh Setyaningrum & Nadhif (2025) yang fokus pada klasifikasi *cyberbullying* dalam bahasa Sunda menggunakan CNN–BiLSTM dan FastText, mereka melaporkan akurasi yang sangat tinggi, yaitu 97.3%, dan F1-score 97%. Meskipun angka ini lebih tinggi daripada hasil penelitian ini, perbedaan konteks bahasa (lokal vs nasional) dan kemungkinan *overfitting* akibat kompleksitas data yang lebih rendah harus dipertimbangkan. Selain itu, durasi pelatihan model mereka hanya 30 detik per *epoch*, yang dapat menunjukkan bahwa model tersebut mungkin terlalu sederhana atau kurang kompleks. Penelitian ini mengambil pendekatan yang lebih menyeluruh dengan evaluasi pada berbagai embedding, dimensi, dan metrik yang ketat, serta menggunakan *threshold* ROC yang dioptimalkan.

Terakhir, yang dilakukan oleh Song & Sun, (2021)., mereka menggunakan model BiLSTM–CNN dengan *embedding* Word2Vec. Hasilnya menunjukkan peningkatan akurasi, *recall*, dan *F1-score* lebih dari 1% dibandingkan dengan model individual. Meskipun tidak menyebutkan angka spesifik, peningkatan ini menunjukkan bahwa kombinasi BiLSTM dan CNN memberikan keunggulan dibandingkan arsitektur tunggal. Namun, dalam penelitian ini, model CNN–BiLSTM dengan FastText menunjukkan peningkatan yang lebih signifikan dan terukur, terutama dengan *F1-score* yang melebihi 92%, yang memperkuat efektivitas arsitektur *hybrid* dalam konteks data *cyberbullying*.

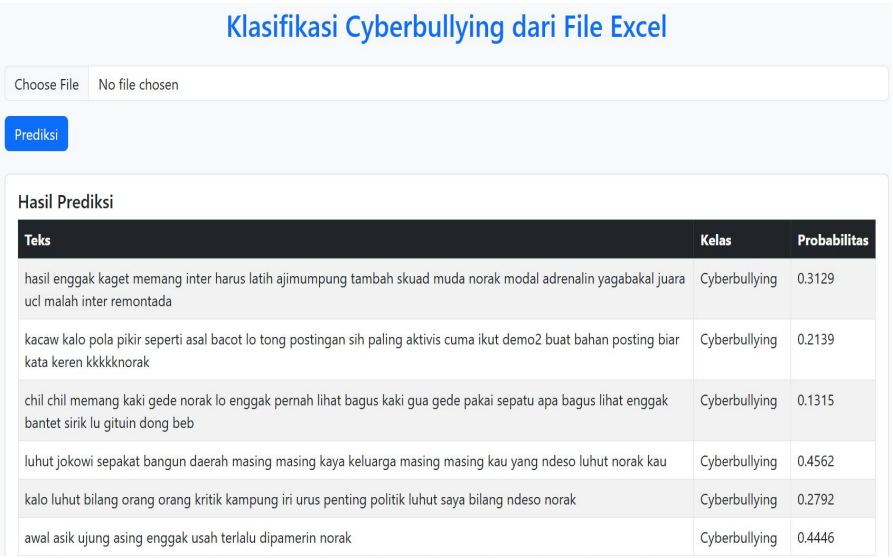
Secara keseluruhan, hasil penelitian ini menunjukkan kinerja yang konsisten unggul atau setidaknya sebanding dengan penelitian-penelitian sebelumnya, terutama dalam hal keseimbangan antara precision dan recall, kemampuan menangani OOV, serta akurasi dalam konteks bahasa informal. Ini menunjukkan bahwa kombinasi arsitektur CNN–BiLSTM dan embedding FastText berdimensi 300 merupakan pendekatan yang sangat efektif untuk tugas klasifikasi *cyberbullying* dalam konteks media sosial berbahasa Indonesia.

4.8 Implementasi Aplikasi Web Sederhana Klasifikasi Cyberbullying

Gambar 4.47 merupakan tampilan antarmuka dari aplikasi web sederhana yang dikembangkan sebagai implementasi dari hasil penelitian. Aplikasi ini dirancang untuk menunjukkan bagaimana model klasifikasi *cyberbullying* dapat diterapkan dalam konteks penggunaan nyata. Meskipun pengembangan aplikasi web ini bukanlah fokus utama dalam penelitian, keberadaannya memberikan nilai tambah dengan memperlihatkan integrasi model yang telah dilatih ke dalam sistem dengan antarmuka pengguna yang ramah.

Aplikasi ini dibangun menggunakan *framework* Flask, dan dijalankan melalui perintah `python app.py`. Aplikasi dirancang untuk memproses input berupa file Excel (.xlsx) yang berisi komentar teks. Setiap komentar diproses menggunakan *tokenizer* yang telah disimpan dalam file `tokenizer.pkl` dari tahap pelatihan, lalu diubah menjadi urutan vektor dan dipadatkan (*padding*) hingga panjang maksimum sekuens yang ditentukan, yaitu 55 token. Setelah itu, komentar-komentar tersebut diklasifikasikan menggunakan model CNN–BiLSTM dengan embedding FastText berdimensi 300, yang disimpan dalam file `cnn_bilstm_fasttext.h5`.

Antarmuka aplikasi ini dirancang secara sederhana dan responsif. Pengguna dapat mengunggah data, melihat hasil klasifikasi setiap komentar dalam bentuk label (*cyberbullying* atau *non-cyberbullying*), serta mengamati nilai probabilitas dari hasil prediksi. Meskipun bukan komponen utama dari penelitian, aplikasi web ini memberikan kontribusi tambahan dengan menyajikan solusi yang praktis, mudah digunakan, dan memiliki potensi untuk dikembangkan lebih lanjut di masa depan.



Gambar 4.47 Tampilan web sederhana klasifikasi *cyberbullying*

BAB 5
PENUTUP

Setelah melakukan pembahasan yang mendalam dalam BAB 4, sejumlah kesimpulan penting dapat diambil, dan berbagai saran yang konstruktif dapat diusulkan. BAB 4 telah menyajikan analisis yang komprehensif mengenai topik yang dibahas, memungkinkan untuk merumuskan beberapa temuan utama dan rekomendasi. Berdasarkan evaluasi dan pemahaman yang diperoleh dari bab tersebut, dapat disimpulkan berbagai aspek yang krusial serta memberikan saran yang relevan dan aplikatif untuk pengembangan selanjutnya.

5.1. Kesimpulan

Berdasarkan hasil penelitian dan eksperimen yang telah dilakukan terhadap tugas klasifikasi komentar *cyberbullying* dengan pendekatan deep learning, dapat diambil beberapa kesimpulan sebagai berikut:

1. Model CNN–BiLSTM terbukti menjadi arsitektur terbaik dalam menangani klasifikasi komentar *cyberbullying* dibandingkan model-model lain seperti CNN, LSTM, dan BiLSTM murni. Kombinasi CNN dan BiLSTM mampu memanfaatkan keunggulan dalam menangkap pola lokal dan konteks sekuensial dua arah secara efektif, yang sangat penting dalam menganalisis teks sosial media yang cenderung tidak terstruktur.
2. Penggunaan *word embedding* FastText menghasilkan performa paling unggul dibandingkan GloVe dan Word2Vec. FastText mampu menangani kata-kata baru, variasi ejaan, dan morfologi bahasa Indonesia informal dengan lebih baik melalui pendekatan *subword embedding*. Hal ini berdampak signifikan terhadap peningkatan nilai *F1-score* dan AUC pada hampir semua model.
3. Model terbaik dalam penelitian ini adalah CNN–BiLSTM dengan *embedding* FastText berdimensi 300, yang menghasilkan *F1-score* sebesar 0.9238**,** AUC 0.9410, *precision* dan *recall* masing-masing 0.9238. Model ini menunjukkan keseimbangan yang sangat baik antara kemampuan klasifikasi (*recall*) dan keakuratan klasifikasi (*precision*), menjadikannya solusi yang andal untuk pendeteksian komentar *cyberbullying.*
4. *Embedding* berdimensi 300 secara umum memberikan performa yang lebih baik dibandingkan *embedding* berdimensi 50, 100, atau 200. Dimensi yang lebih tinggi memungkinkan pemodelan

hubungan semantik kata yang lebih kompleks, selama diimbangi dengan mekanisme regularisasi dan jumlah data yang memadai.

5.2. Saran

Berdasarkan hasil penelitian dan analisis yang telah dilakukan, berikut adalah beberapa saran yang dapat dijadikan masukan untuk pengembangan penelitian lanjutan:

1. Melakukan *fine-tuning* lebih lanjut pada model CNN-BiLSTM dapat membantu meningkatkan akurasi serta mengatasi masalah *overfitting* ringan yang terjadi.
2. Dengan memperluas korpus data dan melibatkan lebih banyak sumber komentar dari berbagai platform media sosial, model akan memiliki kemampuan yang lebih baik dalam menggeneralisasi variasi bahasa dan ekspresi yang lebih luas.
3. Mengimplementasikan teknik augmentasi data berbasis teks atau menyesuaikan ambang batas prediksi berdasarkan analisis *ROC Curve* dapat membantu menyeimbangkan antara *recall* dan *precision*.

DAFTAR PUSTAKA

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. http://arxiv.org/abs/1603.04467

Andika, A. J., Kristian, Y., & Setiawan, E. I. (2023). Deteksi Komentar Cyberbullying Pada YouTube Dengan Metode Convolutional Neural Network – Long Short-Term Memory Network (CNN-LSTM). *Teknika*, *12*(3), 183–188. https://doi.org/10.34148/teknika.v12i3.677

Aqilla, L. N., Sibaroni, Y., & Prasetiyowati, S. S. (2023). Word2vec Architecture in Sentiment Classification of Fuel Price Increase Using CNN-BiLSTM Method. *Sinkron*, *8*(3), 1654–1664. https://doi.org/10.33395/sinkron.v8i3.12639

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. https://doi.org/10.1162/tacl_a_00051

Ciaburro, G., & Joshi, P. (2019). *Python Machine Learning Cookbook: Over 100 Recipes to Progress from Smart Data Analytics to Deep Learning Using Real-World Datasets* (2nd ed.). Packt Publishing.

DataReportal. (2024). *Digital 2024: Indonesia*. https://datareportal.com/reports/digital-2024-indonesia

Dwipayana, N. L. A. M., Setiyono, S., & Pakpahan, H. (2020). Cyberbullying Di Media Sosial. *Bhirawa Law Journal*, *1*(2), 63–70. https://doi.org/10.26905/blj.v1i2.5483

Fauziah, I. N. N., Saputri, S. A., & Herlambang, Y. T. (2024). Teknologi Informasi: Dampak Media Sosial pada Perubahan Sosial Masyarakat. *Indo-MathEdu Intellectuals Journal*, *5*(1), 757–766. https://doi.org/10.54373/imeij.v5i1.645

Graves, A., Fernández, S., & Schmidhuber, J. (2005). Bidirectional LSTM networks for improved phoneme classification and recognition.

Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), *3697 LNCS*, 799–804. https://doi.org/10.1007/11550907_126

Harinsa Putri, L., & Ina Savira, S. (2022). *Dampak Psikologis Pada Remaja Yang Mengalami Cyberbullying Psychological impact on teenagers who experience cyberbullying*. *10*(01), 309–323.

Hidayat, M. R., Firjatulloh Fano, N., Margaretha, R. D., Rasendriya, Z. A., & Rakhmawati, N. A. (2024). Klasifikasi Topik Pembahasan Mahasiswa ITS dalam Bermedia Sosial Menggunakan Latent Dirichlet Allocation. In *Journal Information Engineering and Educational Technology* (Vol. 08).

Hope, T., Resheff, Y. S., & Lieder, I. (2017). *Learning TensorFlow: A Guide to Building Deep Learning Systems*. O'Reilly Media. https://books.google.co.id/books/about/Learning_TensorFlow.html?id=YFPwswEACAAJ&redir_esc=y

Informatika, K. K. dan. (2024). *Indonesia masuk 5 besar pengguna X di dunia*. https://www.kominfo.go.id/content/detail/63743/indonesia-masuk-5-besar-pengguna-x-di-dunia/0/berita

Ira, Jusak, & Gerald, M. (2021). *TA : Analisis Perbandingan Metode LSTM dan BiLSTM untuk Klasifikasi Sinyal Jantung Phonocardiogram*. https://repository.dinamika.ac.id/id/eprint/5962/

Jurafsky, Daniel S. and Martin, J. H. (2025). *Speech and Language Processing: An Introduction to NLP, Computational Linguistics, and Speech Recognition with Language Modelstle* (3rd ed.). Stanford University. https://doi.org/https://doi.org/10.1162/coli.B09-001

Khan, J., Ahmad, K., Jagatheesaperumal, S. K., & Sohn, K. (2025). Textual variations in social media text processing applications: challenges, solutions, and trends. *Artificial Intelligence Review*, *58*(3), 89. https://doi.org/10.1007/s10462-024-11071-z

Kim, Y. (2014). Convolutional neural networks for sentence classification. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1746–1751. https://doi.org/10.3115/v1/d14-1181

Kuo, C. (2023). *The Handbook of NLP with Gensim*. Packt Publishing.

Lu, N., Wu, G., Zhang, Z., Zheng, Y., Ren, Y., & Choo, K. K. R. (2020). Cyberbullying detection in social media text based on character-level convolutional neural network with shortcuts. *Concurrency and Computation: Practice and Experience*, *32*(23). https://doi.org/10.1002/cpe.5627

Martha, A. E. (2024). Perundungan Siber (Cyberbullying) Melalui Media Sosial Instagram dalam Teori the Space Transition of Cybercrimes. *Jurnal Hukum Ius Quia Iustum*, *31*(1), 199–218. https://doi.org/10.20885/iustum.vol31.iss1.art9

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 1–12.

Muslikh, A. R., Akbar, I., Setiadi, D. R. I. M., & Islam, H. M. M. (2024). Multi-label Classification of Indonesian Al-Quran Translation based CNN, BiLSTM, and FastText. *Techno.Com: Jurnal Teknologi Informasi*, *23*(1), 37–50. https://doi.org/10.62411/tc.v23i1.9925

Nugraha Manoppo, T., & Hatta Fudholi, D. (2021). Deteksi Cyberbullying berdasarkan Unsur Perbuatan Pidana yang Dilanggar dengan Naive Bayes dan Support Vector Machine. In *Jurnal Sains Komputer & Informatika (J-SAKTI* (Vol. 5, Issue 1).

Pawit Widiyantoro, Paradise Paradise, & Yogo Dwi Prasetyo. (2025). Deteksi Cyberbullying pada Pemain Sepak Bola di Platform Media Sosial "X" Menggunakan Metode Long Short-Term Memory (LSTM). *Repeater : Publikasi Teknik Informatika Dan Jaringan*, *3*(1), 201–217. https://doi.org/10.62951/repeater.v3i1.382

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, *19*(5), 1532–1543. https://doi.org/10.3115/v1/D14-1162

Pratiwi, B. P., Handayani, A. S., & Sarjana, S. (2021). PENGUKURAN KINERJA SISTEM KUALITAS UDARA DENGAN TEKNOLOGI WSN MENGGUNAKAN CONFUSION MATRIX. *Jurnal Informatika Upgris*, *6*(2), 66–76. https://doi.org/10.26877/jiu.v6i2.6552

Purwarianti, A., & Crisdayanti, I. A. P. A. (2019). Improving Bi-LSTM Performance for Indonesian Sentiment Analysis Using Paragraph Vector. *Proceedings - 2019 International Conference on Advanced Informatics: Concepts, Theory, and Applications, ICAICTA 2019*. https://doi.org/10.1109/ICAICTA.2019.8904199

Python Software Foundation. (2023). *Python documentation*. https://docs.python.org/3/

Ramsundar, B., & Zadeh, R. B. (2018). *TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning*. O'Reilly Media.

Rhanoui, M., Mikram, M., Yousfi, S., & Barzali, S. (2019). A CNN-BiLSTM Model for Document-Level Sentiment Analysis. *Machine Learning and Knowledge Extraction*, *1*(3), 832–847. https://doi.org/10.3390/make1030048

Setyaningrum, A. P., & Nadhif, M. F. (2025). Klasifikasi Cyberbullying Pada Tweet Bahasa Sunda Dengan Menggunakan Hybrid Learning Model. *Jurnal Rekayasa Hijau*, *9*(1), 58–69. https://doi.org/10.26760/jrh.v9i1.58-69

Silitonga, P. (2023). Pengaruh Positif dan Negatif Media Sosial terhadap Perkembangan Psikologis dan Perilaku Remaja yang Tidak ternbiasa dengan Teknologi Sosial Media di Indonesia. *Jurnal Pendidikan Sosial Dan Humaniora*, *2*(4), 13077–13089. https://publisherqu.com/index.php/pediaqu

Song, D., & Sun, B. (2021). Longer Text Classification Based on BiLSTM-CNN Model. *International Journal of Engineering Research And Management (IJERM)*, *08*(09), 1–3. https://www.ijerm.com/download_data/IJERM0809001.pdf

Sukma, B. P., Puspitasari, D. A., Afiyani, S. A., Okitasari, I., Palupi, D., Kusumawardani, F., Khatimah, H., & Prayoga, R. A. (2024). Pola tuturan perundungan siber (cyberbullying) di kalangan pelajar Indonesia. *Bahasa Dan Seni: Jurnal Bahasa, Sastra, Seni, Dan Pengajarannya*, *49*(2). https://doi.org/10.17977/um015v49i22021p205

Whittaker, E., & Kowalski, R. M. (2015). Cyberbullying Via Social Media. *Journal of School Violence*, *14*(1), 11–29.

https://doi.org/10.1080/15388220.2014.949377

Widhiyasana, Y., Semiawan, T., Gibran, I., Mudzakir, A., & Noor, M. R. (2021). Penerapan Convolutional Long Short-Term Memory untuk Klasifikasi Teks Berita Bahasa Indonesia (Convolutional Long Short-Term Memory Implementation for Indonesian News Classification). In *Jurnal Nasional Teknik Elektro dan Teknologi Informasi* | (Vol. 10, Issue 4).

Willard, N. (2005). Educator ' s Guide to Cyberbullying and Cyberthreats. *Online*, 1–16. https://cdn.ymaws.com/www.safestates.org/resource/resmgr/imported/educatorsguide.pdf

Lampiran 1 Repositori Kode Program Penelitian

Kode QR repositori GitHub klasifikasi *cyberbullying* dengan model CNN–BiLSTM yang dikembangkan dalam penelitian ini.

Tautan Repositori GitHub:

https://github.com/sarifauziaelza/cnn-bilstm-cyberbullying

Lampiran 2 Berita Acara Labeling Data

BERITA ACARA PELABELAN DATA

Pada hari ini, Rabu, tanggal 25 / Juni / 2025 telah selesai dilaksanakan proses pelabelan data komentar dalam rangka penelitian yang berjudul:

"**Klasifikasi Cyberbullying pada Media Sosial Menggunakan Model Hybrid CNN–BiLSTM**"

Pelabelan dilakukan terhadap data komentar yang diperoleh dari media sosial dengan tujuan untuk mengklasifikasikan komentar ke dalam dua kategori utama berdasarkan isi dan konteks ujaran. Adapun kriteria pelabelan yang digunakan dalam penelitian ini adalah sebagai berikut:

- Label 0: Komentar mengandung unsur cyberbullying, seperti hinaan, pelecehan, provokasi, atau ujaran kebencian yang menyerang individu atau kelompok.
- Label 1: Komentar tidak mengandung unsur cyberbullying dan bersifat netral, positif, informatif, atau tidak menyerang pihak tertentu.

Proses pelabelan dilakukan secara manual oleh pihak yang telah ditunjuk sebagai pelabel, dengan mempertimbangkan konteks kalimat, nada ujaran, serta makna implisit yang mungkin terkandung dalam komentar. Jumlah total data yang diberi label pada tahap ini adalah sebanyak 2.453 komentar.

Data yang telah diberi label ini kemudian digunakan sebagai dataset utama dalam proses pelatihan dan pengujian model klasifikasi dalam penelitian ini.

Demikian berita acara ini dibuat dengan sebenarnya untuk digunakan sebagaimana mestinya.

Pelabel

Nama/	: ONI SURYONO, S.Pd / 196908222007011005
Jabatan/Instansi	: Guru Bahasa Indonesia / SMAN 2 Teluk Kuantan
Tanggal Pelabelan	: 16 Juni - 25 Juni 2025

Teluk Kuantan, 27 Juni 2025
Tanda Tangan

ONI SURYONO, S.Pd
NIP. 196908222007011005

Lampiran 3 Implementasi model lainnya

1. Implementasi arsitektur BiLSTM

```
# Bi-LSTM Model
inp = Input(shape=(max_len,))
x = Embedding(input_dim=vocab_size, output_dim=dim, weights=[embedding_matrix], input_length=max_len, trainable=True)(inp)
x = Bidirectional(LSTM(64, return_sequences=True, kernel_regularizer=l2(0.001)))(x)
x = Dropout(0.3)(x)
x = Bidirectional(LSTM(32, kernel_regularizer=l2(0.001)))(x)
x = Dropout(0.3)(x)
x = Dense(64, activation='relu', kernel_regularizer=l2(0.001))(x)
x = Dropout(0.3)(x)
out = Dense(1, activation='sigmoid')(x)

model = Model(inputs=inp, outputs=out)
model.compile(loss='binary_crossentropy', optimizer=Adam(1e-5), metrics=['accuracy'])

# Training
early_stop = EarlyStopping(monitor='val_loss', patience=6, restore_best_weights=True)
history = model.fit(X_resampled, y_resampled, epochs=50, batch_size=64,
                    validation_data=(X_val, y_val), callbacks=[early_stop], verbose=0)
```

2. Implementasi arsitektur LSTM

```
# LSTM Model
inp = Input(shape=(max_len,))
x = Embedding(input_dim=vocab_size, output_dim=dim, weights=[embedding_matrix], input_length=max_len, trainable=True)(inp)
x = LSTM(64, return_sequences=True, kernel_regularizer=l2(0.001))(x)
x = Dropout(0.3)(x)
x = LSTM(32, kernel_regularizer=l2(0.001))(x)
x = Dropout(0.3)(x)
x = Dense(64, activation='relu', kernel_regularizer=l2(0.001))(x)
x = Dropout(0.3)(x)
out = Dense(1, activation='sigmoid')(x)

model = Model(inputs=inp, outputs=out)
model.compile(loss='binary_crossentropy', optimizer=Adam(1e-5), metrics=['accuracy'])

# Train
early_stop = EarlyStopping(monitor='val_loss', patience=6, restore_best_weights=True)
history = model.fit(X_resampled, y_resampled, epochs=50, batch_size=64,
                    validation_data=(X_val, y_val), callbacks=[early_stop], verbose=0)
```

3. Implementasi arsitektur CNN

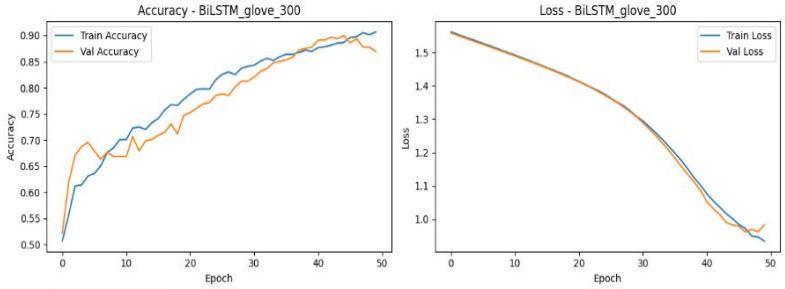
```
# CNN Model
inp = Input(shape=(max_len,))
x = Embedding(input_dim=vocab_size, output_dim=dim, weights=[embedding_matrix], input_length=max_len, trainable=True)(inp)
x = Conv1D(128, kernel_size=3, activation='relu', kernel_regularizer=l2(0.001))(x)
x = MaxPooling1D(pool_size=2)(x)
x = Dropout(0.3)(x)
x = Conv1D(64, kernel_size=3, activation='relu', kernel_regularizer=l2(0.001))(x)
x = GlobalMaxPooling1D()(x)
x = Dense(64, activation='relu', kernel_regularizer=l2(0.001))(x)
x = Dropout(0.3)(x)
out = Dense(1, activation='sigmoid')(x)

model = Model(inputs=inp, outputs=out)
model.compile(loss='binary_crossentropy', optimizer=Adam(1e-5), metrics=['accuracy'])

# Train
early_stop = EarlyStopping(monitor='val_loss', patience=6, restore_best_weights=True)
history = model.fit(X_resampled, y_resampled, epochs=50, batch_size=64,
                    validation_data=(X_val, y_val), callbacks=[early_stop], verbose=0)
```

4. Implementasi arsitektur CNN LSTM

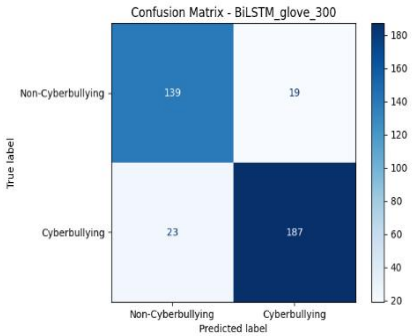
```
# === Build CNN + LSTM Model ===
inp = Input(shape=(max_len,))
x = Embedding(input_dim=vocab_size, output_dim=dim, weights=[embedding_matrix], input_length=max_len, trainable=True)(inp)
x = Conv1D(128, kernel_size=3, activation='relu', kernel_regularizer=l2(0.001))(x)
x = MaxPooling1D(pool_size=2)(x)
x = LSTM(64, return_sequences=False, kernel_regularizer=l2(0.001))(x)
x = Dropout(0.3)(x)
x = Dense(64, activation='relu', kernel_regularizer=l2(0.001))(x)
x = Dropout(0.3)(x)
out = Dense(1, activation='sigmoid')(x)

model = Model(inputs=inp, outputs=out)
model.compile(loss='binary_crossentropy', optimizer=Adam(1e-5), metrics=['accuracy'])

# Training
early_stop = EarlyStopping(monitor='val_loss', patience=6, restore_best_weights=True)
history = model.fit(
```

Lampiran 4 Grafik akurasi *loss* dan *confusion matrix* model lainnya

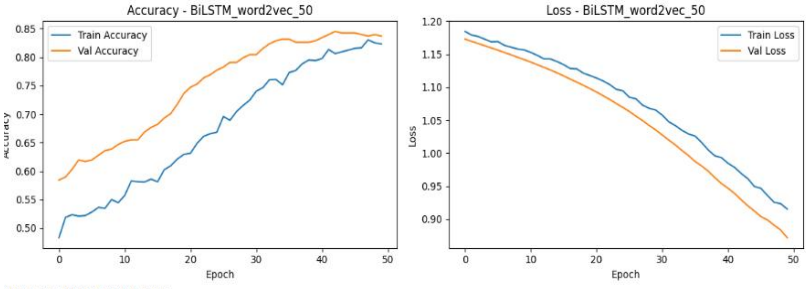
1. Grafik Akurasi *loss* dan *confusion matrix* model BiLSTM
   a. *Word embedding* GloVe

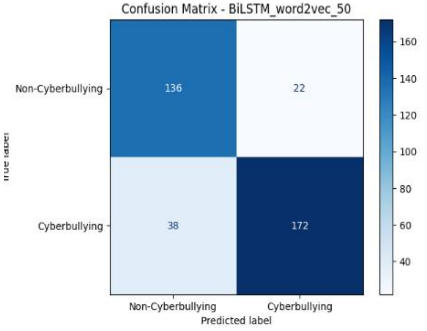


<Figure size 600x400 with 0 Axes>

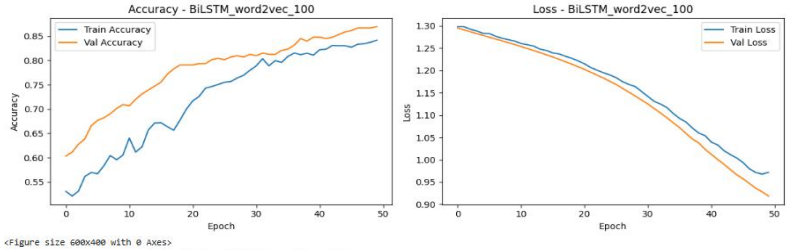


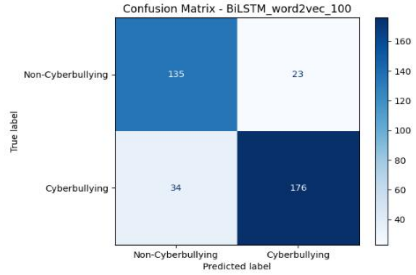
   b. *Word embedding* Word2Vec 50





c. *Word embedding* Word2Vec 100





d. *Word embedding* Word2Vec 200

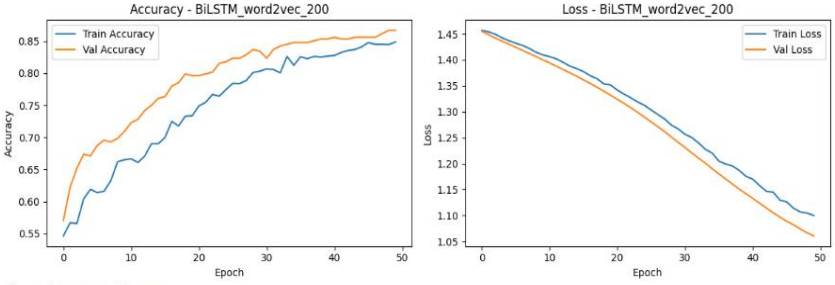
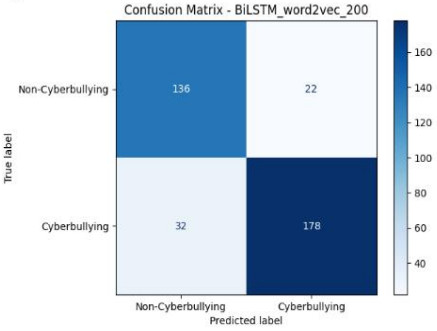
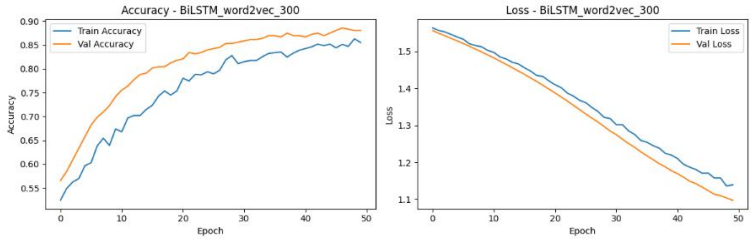


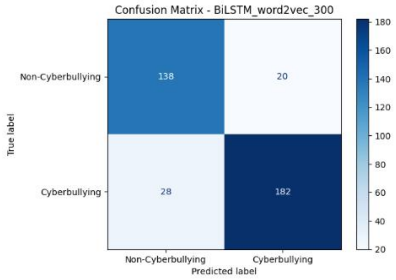
Figure size 600x400 with 0 Axes>



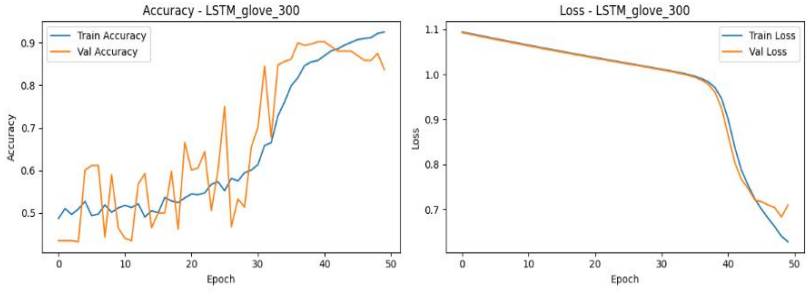
e. *Word embedding* Word2Vec 300

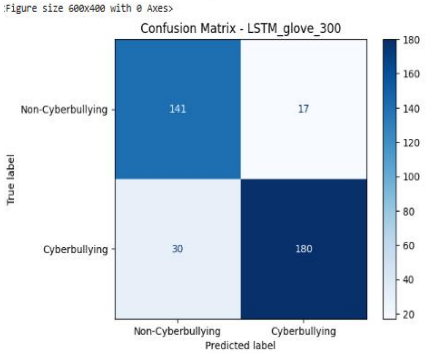


<Figure size 600x400 with 0 Axes>

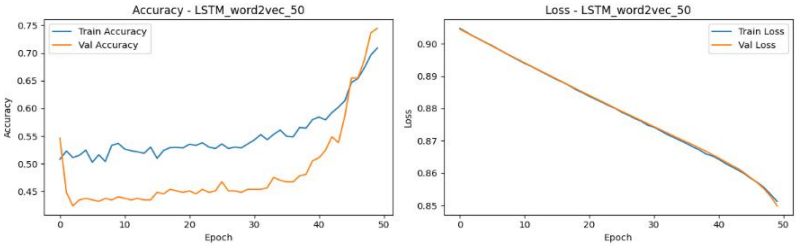


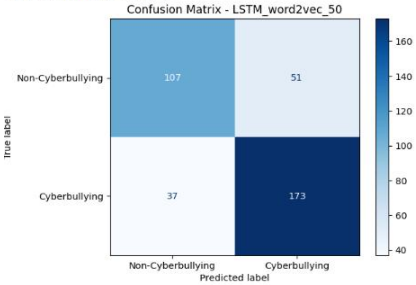
2. Grafik Akurasi *loss* dan *confusion matrix* model LSTM
   a. *Word embedding* GloVe





b. *Word embedding* Word2Vec 50





c. *Word embedding* Word2Vec 100

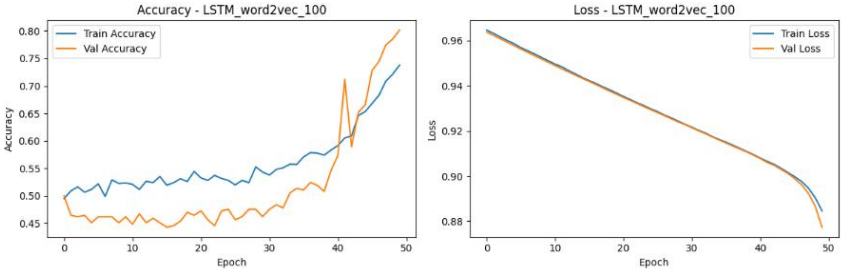
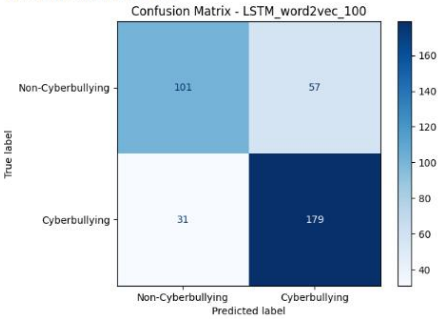


Figure size 600x400 with 0 Axes>



d. *Word embedding* Word2Vec 200

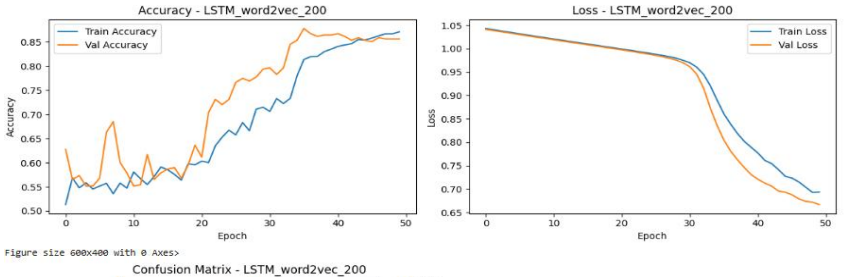
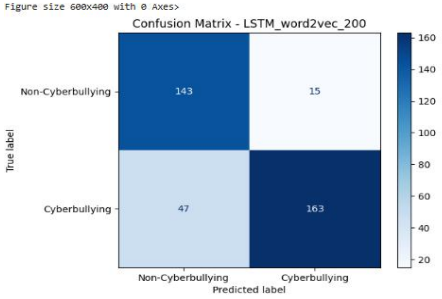
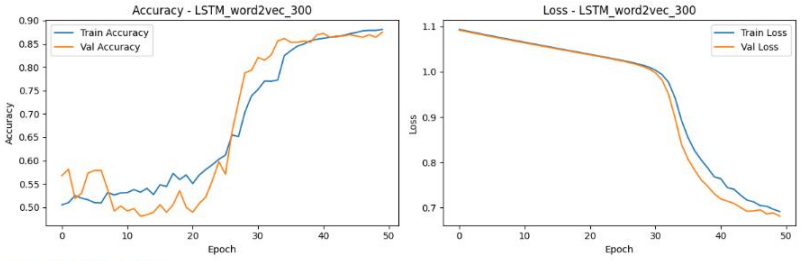
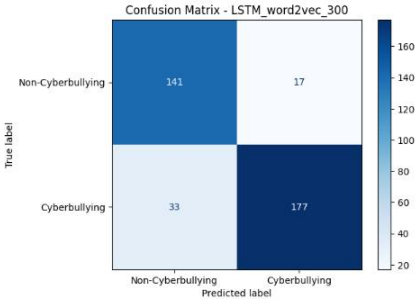


Figure size 600x400 with 0 Axes>

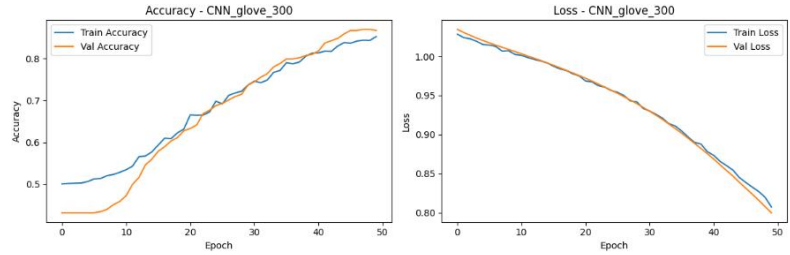


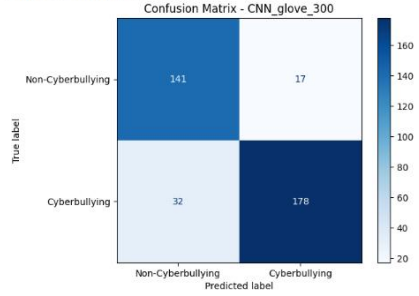
e. *Word embedding* Word2Vec 300



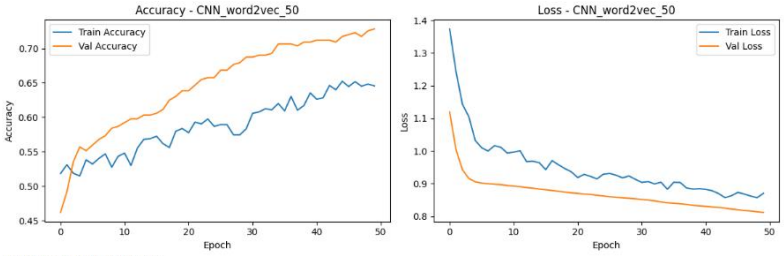


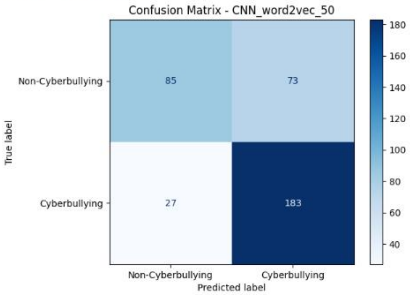
3. Grafik Akurasi *loss* dan *confusion matrix* model CNN
   a. *Word embedding* GloVe



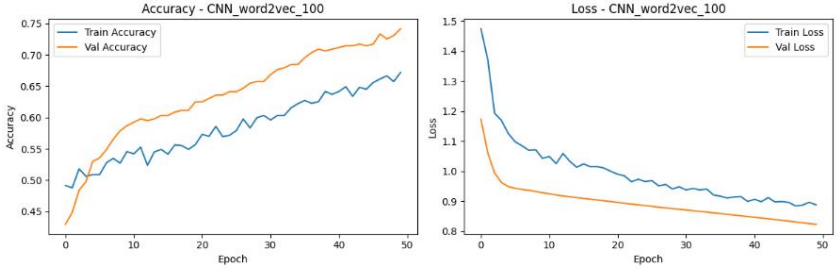


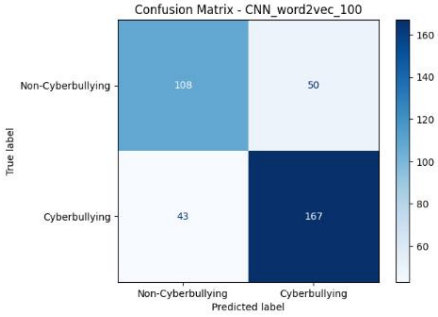
   b. *Word embedding* Word2Vec 50



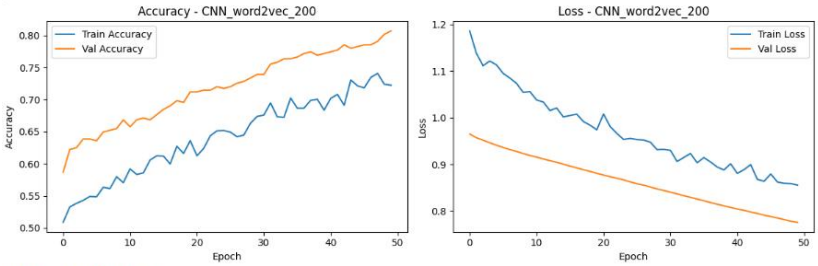


c. *Word embedding* Word2Vec 100

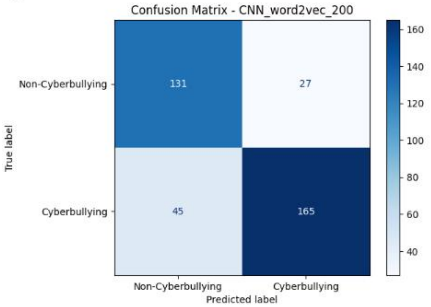




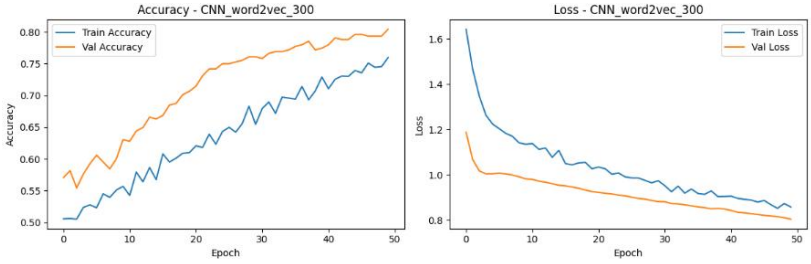
d. *Word embedding* Word2Vec 200



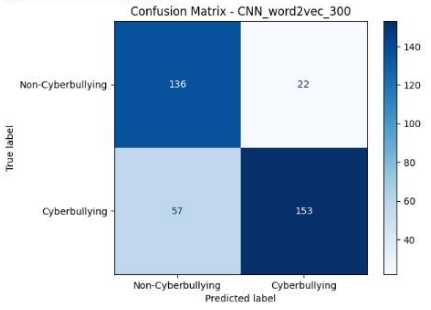
<Figure size 600x400 with 0 Axes>



e. *Word embedding* Word2Vec 300

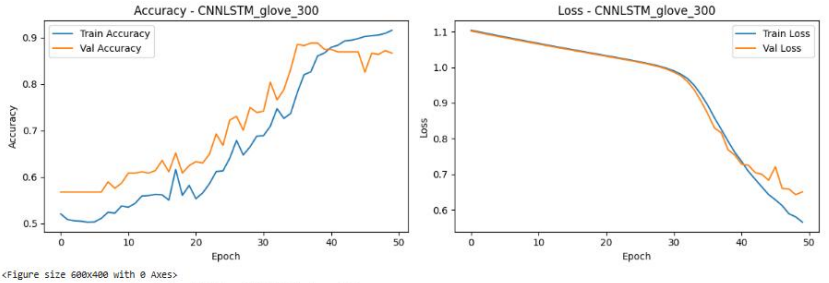


<Figure size 600x400 with 0 Axes>

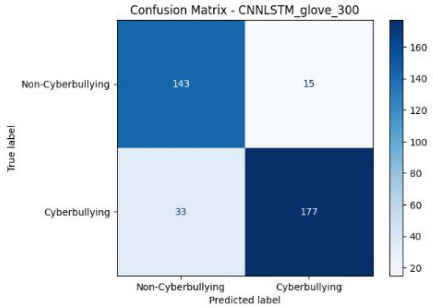


4. Grafik Akurasi *loss* dan *confusion matrix* model CNN LSTM

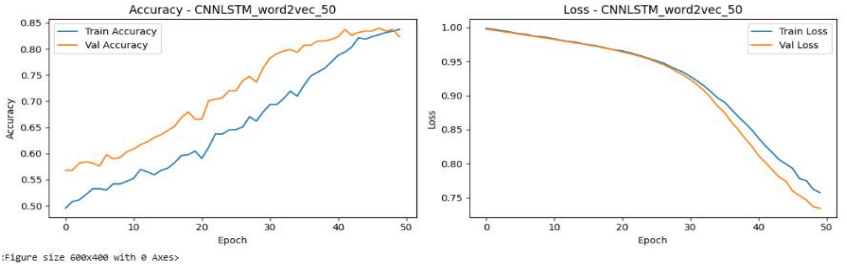
a. *Word embedding* GloVe



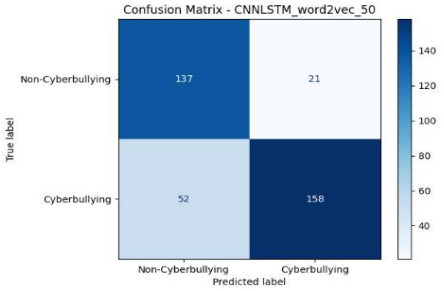
<Figure size 600x400 with 0 Axes>



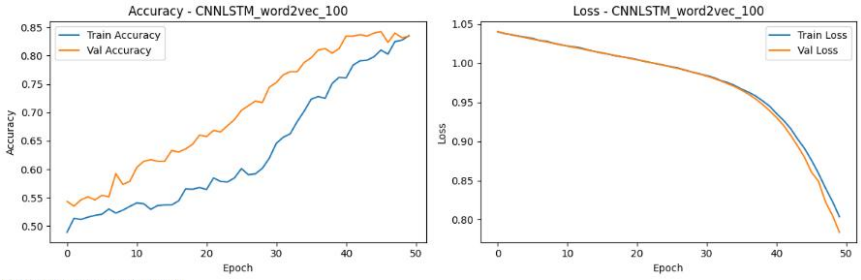
b. *Word embedding* Word2Vec 50



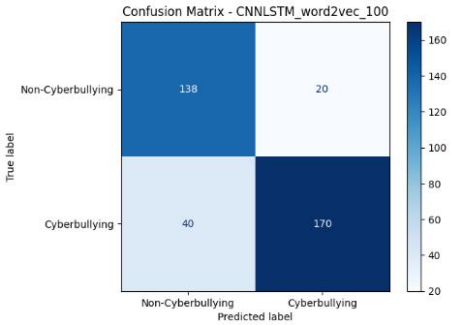
:Figure size 600x400 with 0 Axes>



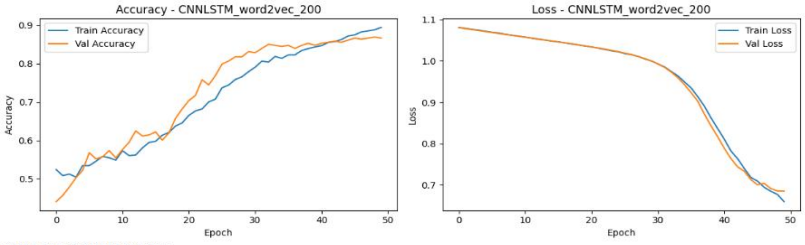
c. *Word embedding* Word2Vec 100



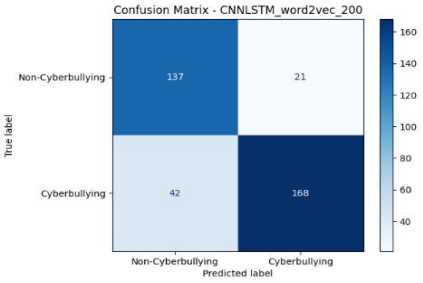
<Figure size 600x400 with 0 Axes>



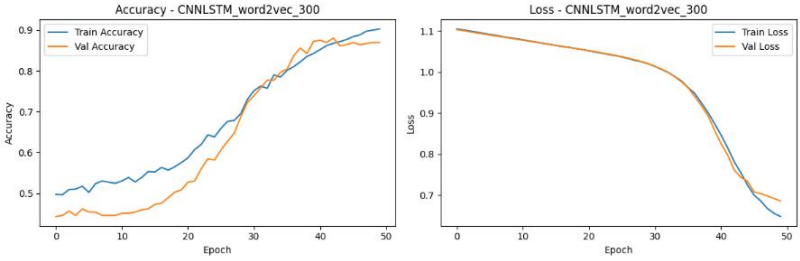
d. *Word embedding* Word2Vec 200



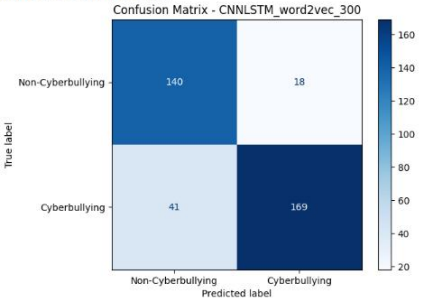
<Figure size 600x400 with 0 Axes>



e. *Word embedding* Word2Vec 300



<Figure size 600x400 with 0 Axes>



Lampiran 5 *Classification report* model lainnya

1. *Classification report* model BiLSTM
   a. *Word embedding* GloVe

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.86	0.88	0.87	158
Cyberbullying	0.91	0.89	0.90	210
accuracy			0.89	368
macro avg	0.88	0.89	0.88	368
weighted avg	0.89	0.89	0.89	368

   b. *Word embedding* Word2Vec 50

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.78	0.86	0.82	158
Cyberbullying	0.89	0.82	0.85	210
accuracy			0.84	368
macro avg	0.83	0.84	0.84	368
weighted avg	0.84	0.84	0.84	368

   c. *Word embedding* Word2Vec 100

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.80	0.85	0.83	158
Cyberbullying	0.88	0.84	0.86	210
accuracy			0.85	368
macro avg	0.84	0.85	0.84	368
weighted avg	0.85	0.85	0.85	368

d. *Word embedding* Word2Vec 200

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.81	0.86	0.83	158
Cyberbullying	0.89	0.85	0.87	210
accuracy			0.85	368
macro avg	0.85	0.85	0.85	368
weighted avg	0.86	0.85	0.85	368

e. *Word embedding* Word2Vec 300

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.83	0.87	0.85	158
Cyberbullying	0.90	0.87	0.88	210
accuracy			0.87	368
macro avg	0.87	0.87	0.87	368
weighted avg	0.87	0.87	0.87	368

2. *Classification report* model LSTM

a. *Word embedding* GloVe

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.82	0.89	0.86	158
Cyberbullying	0.91	0.86	0.88	210
accuracy			0.87	368
macro avg	0.87	0.87	0.87	368
weighted avg	0.88	0.87	0.87	368

b. *Word embedding* Word2Vec 50

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.74	0.68	0.71	158
Cyberbullying	0.77	0.82	0.80	210
accuracy			0.76	368
macro avg	0.76	0.75	0.75	368
weighted avg	0.76	0.76	0.76	368

c. *Word embedding* Word2Vec 100

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.77	0.64	0.70	158
Cyberbullying	0.76	0.85	0.80	210
accuracy			0.76	368
macro avg	0.76	0.75	0.75	368
weighted avg	0.76	0.76	0.76	368

d. *Word embedding* Word2Vec 200

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.75	0.91	0.82	158
Cyberbullying	0.92	0.78	0.84	210
accuracy			0.83	368
macro avg	0.83	0.84	0.83	368
weighted avg	0.85	0.83	0.83	368

e. *Word embedding* Word2Vec 300

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.81	0.89	0.85	158
Cyberbullying	0.91	0.84	0.88	210
accuracy			0.86	368
macro avg	0.86	0.87	0.86	368
weighted avg	0.87	0.86	0.86	368

3. *Classification report* model CNN

a. *Word embedding* GloVe

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.82	0.89	0.85	158
Cyberbullying	0.91	0.85	0.88	210
accuracy			0.87	368
macro avg	0.86	0.87	0.87	368
weighted avg	0.87	0.87	0.87	368

b. *Word embedding* Word2Vec 50

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.76	0.54	0.63	158
Cyberbullying	0.71	0.87	0.79	210
accuracy			0.73	368
macro avg	0.74	0.70	0.71	368
weighted avg	0.73	0.73	0.72	368

c. *Word embedding* Word2Vec 100

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.72	0.68	0.70	158
Cyberbullying	0.77	0.80	0.78	210
accuracy			0.75	368
macro avg	0.74	0.74	0.74	368
weighted avg	0.75	0.75	0.75	368

d. *Word embedding* Word2Vec 200

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.74	0.83	0.78	158
Cyberbullying	0.86	0.79	0.82	210
accuracy			0.80	368
macro avg	0.80	0.81	0.80	368
weighted avg	0.81	0.80	0.81	368

e. *Word embedding* Word2Vec 300

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.70	0.86	0.77	158
Cyberbullying	0.87	0.73	0.79	210
accuracy			0.79	368
macro avg	0.79	0.79	0.78	368
weighted avg	0.80	0.79	0.79	368

4. *Classification report* model CNN LSTM
   a. *Word embedding* GloVe

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.81	0.91	0.86	158
Cyberbullying	0.92	0.84	0.88	210
accuracy			0.87	368
macro avg	0.87	0.87	0.87	368
weighted avg	0.87	0.87	0.87	368

   b. *Word embedding* Word2Vec 50

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.72	0.87	0.79	158
Cyberbullying	0.88	0.75	0.81	210
accuracy			0.80	368
macro avg	0.80	0.81	0.80	368
weighted avg	0.81	0.80	0.80	368

   c. *Word embedding* Word2Vec 100

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.78	0.87	0.82	158
Cyberbullying	0.89	0.81	0.85	210
accuracy			0.84	368
macro avg	0.84	0.84	0.84	368
weighted avg	0.84	0.84	0.84	368

   d. *Word embedding* Word2Vec 200

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.77	0.87	0.81	158
Cyberbullying	0.89	0.80	0.84	210
accuracy			0.83	368
macro avg	0.83	0.83	0.83	368
weighted avg	0.84	0.83	0.83	368

e. *Word embedding* Word2Vec 300

Classification Report:

	precision	recall	f1-score	support
Non-Cyberbullying	0.77	0.89	0.83	158
Cyberbullying	0.90	0.80	0.85	210
accuracy			0.84	368
macro avg	0.84	0.85	0.84	368
weighted avg	0.85	0.84	0.84	368

Lampiran 6 Rekapitulasi perbandingan model lainnya

1. Model BiLSTM

	Model	Validation Accuracy	AUC	Precision	Recall	F1-Score
0	BiLSTM_fasttext_300	0.9130	0.9276	0.9113	0.8810	0.8959
1	BiLSTM_glove_300	0.8995	0.9307	0.9078	0.8905	0.8990
2	BiLSTM_word2vec_300	0.8859	0.9152	0.9010	0.8667	0.8835
3	BiLSTM_word2vec_100	0.8696	0.9022	0.8844	0.8381	0.8606
4	BiLSTM_word2vec_200	0.8668	0.8975	0.8900	0.8476	0.8683
5	BiLSTM_word2vec_50	0.8451	0.8700	0.8866	0.8190	0.8515

2. Model LSTM

	Model	Validation Accuracy	AUC	Precision	Recall	F1-Score
0	LSTM_fasttext_300	0.9130	0.9250	0.9101	0.8190	0.8622
1	LSTM_glove_300	0.9022	0.9242	0.9137	0.8571	0.8845
2	LSTM_word2vec_200	0.8777	0.9092	0.9157	0.7762	0.8402
3	LSTM_word2vec_300	0.8750	0.9191	0.9124	0.8429	0.8762
4	LSTM_word2vec_100	0.8016	0.7776	0.7585	0.8524	0.8027
5	LSTM_word2vec_50	0.7446	0.7585	0.7723	0.8238	0.7972

3. Model CNN

	Model	Validation Accuracy	AUC	Precision	Recall	F1-Score
0	CNN_fasttext_300	0.9076	0.9260	0.9154	0.8762	0.8954
1	CNN_glove_300	0.8696	0.9213	0.9128	0.8476	0.8790
2	CNN_word2vec_200	0.8071	0.8626	0.8594	0.7857	0.8209
3	CNN_word2vec_300	0.8043	0.8589	0.8743	0.7286	0.7948
4	CNN_word2vec_50	0.7283	0.7713	0.7148	0.8714	0.7854
5	CNN_word2vec_100	0.7418	0.8220	0.7696	0.7952	0.7822

4. Model CNN LSTM

	Model	Validation Accuracy	AUC	Precision	Recall	F1-Score
0	CNNLSTM_fasttext_300	0.9076	0.9312	0.9038	0.8952	0.8995
1	CNNLSTM_glove_300	0.8886	0.9308	0.9219	0.8429	0.8806
2	CNNLSTM_word2vec_300	0.8804	0.8860	0.9037	0.8048	0.8514
3	CNNLSTM_word2vec_100	0.8424	0.8393	0.8947	0.8095	0.8500
4	CNNLSTM_word2vec_200	0.8696	0.8784	0.8889	0.8000	0.8421
5	CNNLSTM_word2vec_50	0.8397	0.8353	0.8827	0.7524	0.8123