ANALISIS SENTIMEN KOMENTAR CYBERBULLYING PADA SOSIAL MEDIA TIKTOK MENGGUNAKAN ARSITEKTUR TEXTCNN

TUGAS AKHIR

Diajukan sebagai syarat menyelesaikan jenjang strata Satu (S-1) di
Program Studi Teknik Informatika, Fakultas Teknologi Industri,
Institut Teknologi Sumatera

Oleh:

Nikola Arinanda

121140202

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
LAMPUNG SELATAN

2026

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR RUMUS

DAFTAR KODE

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang pesat telah membawa perubahan signifikan dalam cara manusia beraktivitas, khususnya dalam berkomunikasi. Berbagai teknologi baru bermunculan, seperti internet dan media sosial, yang memberikan kemudahan dalam berinteraksi. Namun, di balik kemudahan tersebut, muncul pula tantangan serius, salah satunya terkait dampaknya terhadap kesehatan mental [1]. Salah satu gangguan kesehatan mental yang muncul seiring kemajuan teknologi komunikasi, khususnya media sosial, adalah *cyberbullying*.

Cyberbullying adalah tindakan agresif yang disengaja dan berulang kali dilakukan menggunakan perangkat elektronik terhadap korban yang sulit membela diri [2]. Berdasarkan studi yang dilakukan oleh S. Hinduja *et al.*, *cyberbullying* memiliki beberapa kriteria, yaitu disengaja, berulang, dan berbahaya [3]. Dampak dari *cyberbullying* meliputi rasa malu, takut, dan terintimidasi yang dapat mengganggu kesehatan mental korban serta berpotensi menimbulkan depresi berlebih. Perilaku *cyberbullying* diamati sering terjadi di berbagai platform media sosial.

Perkembangan internet dalam bidang teknologi informasi telah menghadirkan berbagai alternatif media komunikasi, salah satunya media sosial. Salah satu platform media sosial yang paling populer adalah TikTok, yang dikembangkan oleh ByteDance dan menggunakan format video pendek sebagai media utama, sehingga memberikan pengalaman baru dibandingkan dengan media sosial lainnya. Berdasarkan laporan We Are Social tahun 2025, TikTok menempati peringkat ke-4 sebagai platform media sosial yang paling banyak digunakan di Indonesia [4]. Popularitasnya membuat TikTok digunakan

oleh jutaan orang dari berbagai usia, latar belakang, dan budaya [5]. Namun, seperti media sosial pada umumnya, karakteristik penggunaan bahasa di TikTok cenderung singkat, tidak baku, dan informal. Hal ini menimbulkan tantangan tersendiri dalam mendeteksi adanya komentar *cyberbullying*, karena proses identifikasi secara manual memerlukan waktu dan usaha yang besar.

Mengatasi masalah tersebut, B.A. Prameswari *et al* melalui penelitiannya menawarkan solusi berupa model analisis sentimen yang mampu mengklasifikasikan komentar *cyberbullying* dan *non-cyberbullying* [6]. Analisis sentimen memungkinkan proses identifikasi sentimen dalam teks, yang dapat dikategorikan menjadi sentimen positif dan negatif. Pendekatan modern dalam analisis sentimen banyak mengandalkan teknik *deep learning*. Metode ini, yang merupakan bagian dari *machine learning* dengan jaringan saraf tiruan berlapis-lapis, memiliki kemampuan yang baik dalam membaca pola kompleks dari data yang tidak terstruktur, termasuk data teks, tanpa perlu fitur yang dirancang secara manual [7]. Dengan kemampuan tersebut, model dapat menganalisis dan memahami teks secara lebih akurat secara otomatis.

Seiring dengan meningkatnya penggunaan *deep learning* dalam analisis sentimen, peran *Natural Language Processing* (NLP) juga menjadi krusial. NLP berfokus pada interaksi antara komputer dan bahasa manusia, memungkinkan mesin untuk memproses, menganalisis, memahami, dan menghasilkan bahasa alami. Hal ini sangat penting untuk mengurai makna di balik komentar-komentar pengguna dan mengidentifikasi sentimen yang terkandung di dalamnya [6]. Selain itu, kemajuan dalam arsitektur model *deep learning* telah meningkatkan kemampuan NLP dalam menangkap konteks dan nuansa bahasa yang kompleks. Implementasi NLP pada analisis sentimen tidak hanya membantu dalam klasifikasi polaritas teks, tetapi juga dapat digunakan untuk mendeteksi pola bahasa yang berpotensi mengarah pada perilaku tertentu seperti *cyberbullying*.

Penelitian sebelumnya juga telah mencoba mengatasi masalah analisis

sentimen *cyberbullying* dengan menggunakan arsitektur canggih seperti *Bidirectional Encoder Representations from Transformers* (BERT) melalui proses *fine-tuning* pada dataset spesifik. Model-model berbasis transformer seperti BERT memang menunjukkan performa tinggi dalam banyak tugas NLP [6]. Namun, hasil penelitian sebelumnya menunjukkan kecenderungan terjadinya *overfitting*, di mana model menjadi terlalu spesifik terhadap data pelatihan sehingga performanya menurun pada data baru yang belum pernah dilihat sebelumnya, terutama pada dataset dengan ukuran terbatas atau distribusi data yang tidak seimbang [8].

Melihat tantangan tersebut, arsitektur TextCNN muncul sebagai alternatif yang menjanjikan. Penelitian terdahulu terkait TextCNN telah membuktikan keunggulannya dalam mengekstraksi fitur penting dari rangkaian kata dan telah banyak digunakan dalam berbagai tugas klasifikasi teks, termasuk analisis sentimen serta deteksi spam atau misinformasi. Keunggulan utamanya adalah kemampuannya dalam mempelajari representasi fitur secara otomatis melalui *convolution layer*, yang efektif menangkap pola lokal dan global dalam teks [9]. Temuan ini menegaskan potensi penggunaan arsitektur TextCNN dalam melakukan analisis sentimen, khususnya untuk klasifikasi komentar *cyberbullying* di platform media sosial TikTok.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, maka permasalahan penelitian dirumuskan sebagai berikut:

1. Bagaimana mengklasifikan komentar yang termasuk dan tidak termasuk *cyberbullying* pada platform TikTok dengan arsitektur TextCNN?
2. Bagaimana menguji performa model TextCNN dalam mengklasifikasikan komentar yang termasuk dan tidak termasuk *cyberbullying*?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah diuraikan di atas, maka tujuan dari penelitian ini adalah:

1. Mengembangkan model klasifikasi untuk mendeteksi komentar yang termasuk dan tidak termasuk *cyberbullying* pada platform TikTok menggunakan arsitektur TextCNN.
2. Melakukan pengujian dan evaluasi performa model TextCNN dalam mengklasifikasikan komentar *cyberbullying* dan komentar non-*cyberbullying* menggunakan matriks evaluasi berdasarkan *confusion matrix*.

1.4 Batasan Masalah

Adapun batasan masalah dari penelitian ini agar sesuai dengan yang diharapkan adalah sebagai berikut:

1. Penelitian ini hanya menggunakan data komentar yang berasal dari platform TikTok.
2. Dataset komentar TikTok yang digunakan dalam penelitian ini diambil dari dataset yang telah dikumpulkan dan dipublikasikan oleh B.A. Prameswari *et al* (2023) [6].
3. Data yang dianalisis berjumlah 1.505 komentar dan seluruhnya menggunakan bahasa Indonesia.

1.5 Manfaat Penelitian

Adapun manfaat yang diperoleh dari hasil penelitian ini adalah sebagai berikut:

1. Memberikan kontribusi dalam pengembangan arsitektur TextCNN yang dapat diimplementasikan untuk mendeteksi komentar *cyberbullying* di media sosial, khususnya pada platform TikTok.
2. Menjadi acuan atau referensi dalam penerapan arsitektur TextCNN untuk

klasifikasi teks pendek, terutama pada komentar media sosial yang memiliki karakteristik bahasa informal dan ringkas.

1.6 Sistematika Penulisan

Sistematika penulisan berisi pembahasan apa yang akan ditulis disetiap Bab. Sistematika pada umumnya berupa paragraf yang setiap paragraf mencerminkan bahasan setiap Bab.

1.6.1 Bab I Pendahuluan

Bab ini membahas latar belakang yang melandasi penelitian, perumusan masalah yang ingin diselesaikan, serta tujuan yang ingin dicapai. Selain itu, dijelaskan juga batasan masalah, manfaat dari penelitian, dan sistematika penulisan sebagai panduan struktur laporan.

1.6.2 Bab II Tinjauan Pustaka

Bab ini menguraikan teori-teori yang menjadi dasar penelitian, seperti konsep *cyberbullying*, analisis sentimen, dan text processing. Juga dijelaskan arsitektur model TextCNN yang digunakan dalam penelitian ini.

1.6.3 Bab III

Bab ini menjelaskan metode yang digunakan dalam penelitian, mulai dari teknik pengumpulan data hingga tahapan prapemrosesan data. Selain itu, dibahas pula perancangan model TextCNN serta metode evaluasi performa model.

1.6.4 Bab IV

Bab ini menyajikan hasil pelatihan dan pengujian model yang telah dibangun. Dilengkapi dengan analisis performa dan interpretasi hasil klasifikasi sentimen untuk menilai keberhasilan model.

1.6.5 Bab V

Bab terakhir ini berisi kesimpulan dari hasil penelitian yang telah dilakukan. Penulis juga memberikan saran sebagai masukan untuk pengembangan penelitian di masa mendatang.

BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Penelitian ini menggunakan beberapa jurnal penelitian dan berbagai literatur yang diperoleh dari berbagai sumber sebagai dasar proses riset dan penulisan tugas akhir. Tinjauan pustaka akan mencakup berbagai informasi dan teori yang relevan dengan penelitian serta penulisan tugas akhir ini. Tinjauan Pustaka ini dapat dilihat pada Tabel 2.1 Tinjauan Pustaka. Penelitian yang dilakukan oleh B.A. Prameswari *et al* pada tahun 2023 yaitu membangun model prediksi untuk mendeteksi komentar *cyberbullying* dari media sosial TikTok. Penulis secara independen mengumpulkan dan memberi label pada 1.510 komentar TikTok menjadi kategori *cyberbullying* (CB) dan *non-cyberbullying* (Non_CB). Pendekatan *deep learning* digunakan dalam penelitian ini, khususnya arsitektur *Bidirectional Encoder Representations from Transformers* (BERT), untuk proses analisis sentimen pada dataset. Setelah *fine-tuning*, model prediksi mencapai akurasi validasi 0,63 pada epoch kesembilan. Studi ini menekankan pentingnya membangun model prediksi untuk mendeteksi komentar *cyberbullying* dan berkontribusi pada pemahaman serta pencegahan perilaku ini di platform media sosial [6].

Penelitian yang dilakukan oleh D. Nugraha dan P. Astuti pada tahun 2023 bertujuan untuk mengukur tingkat *cyberbullying* di Indonesia dengan menganalisis komentar di media sosial Instagram menggunakan algoritma *Support Vector Machine* (SVM). Penelitian ini mengumpulkan 400 data komentar *cyberbullying* dari Instagram, yang dibagi rata menjadi 200 data positif dan 200 data negatif. Data diproses melalui tahapan preprocessing seperti *cleansing, transform case, tokenize, stem, filter stopword,* dan *filter tokens* by length menggunakan alat RapidMiner. Model klasifikasi yang

dihasilkan menggunakan 400 dataset untuk pelatihan menunjukkan akurasi sebesar 84,25%, presisi 80,22%, recall 92,50%, dan nilai AUC sebesar 0,928 [10].

Penelitian yang dilakukan oleh A.Z. Abdullah *et al* pada tahun 2025 bertujuan menganalisis sentimen publik terhadap Tweet Pemilu Presiden Indonesia 2024. Penulis mengumpulkan 62.955 entri dari Twitter, 126.673 dari IndoNews, dan dataset gabungan berjumlah 189.628 entri, yang kemudian diberi label positif, netral, dan negatif. Pendekatan model hibrida *Convolutional Neural Network* (CNN)-*Long Short-Term Memory* (LSTM) dengan perluasan fitur *Word2Vec* dan optimasi *Genetic Algorithm* (GA) digunakan untuk analisis sentimen. Model hibrida CNN-LSTM yang dioptimalkan dengan GA mencapai akurasi tertinggi 84,78% untuk data berita, menunjukkan peningkatan 3,59%. Studi ini mengilustrasikan penerapan inovatif model hibrida CNN-LSTM untuk analisis sentimen dalam konteks pemilihan nasional, meningkatkan akurasi dan efisiensi dalam memahami opini publik dan dinamika politik [11].

Penelitian yang dilakukan oleh S.A. Ardiyansa *et al* pada tahun 2024 bertujuan mengklasifikasi sentimen tweet berbahasa Indonesia pada platform Twitter. Penulis mengumpulkan 6.137 tweet dengan 5 jenis label berbeda (joy, sadness, fear, love, dan anger). Proses pra-pemrosesan data mencakup pembersihan, pengubahan ekspresi/emoji menjadi kata, *stemming*, dan *embedding* kata. Penelitian ini membandingkan beberapa arsitektur, termasuk BERT, LSTM, CNN, serta arsitektur gabungan Transformer-LSTM, LSTM-CNN, dan Transformer-CNN. Model *Transformer-CNN* menunjukkan performa paling unggul dengan akurasi 85,71% pada data uji dan 99,90% pada data latih. Akurasi ini lebih baik dibandingkan arsitektur lainnya. Meskipun waktu pelatihannya 30,17 menit lebih lama dari beberapa model lain, *Transformer-CNN* juga sudah konvergen pada iterasi ke-5. Studi ini menyimpulkan bahwa *Transformer-CNN* adalah pilihan terbaik untuk klasifikasi sentimen yang membutuhkan akurasi tinggi [12].

Penelitian yang dilakukan oleh S. Chen pada tahun 2025 mengeksplorasi evolusi teknik analisis sentimen dari model tradisional hingga pendekatan *deep learning* yang lebih canggih. Penulis menggunakan dataset IMDb yang berisi 50.000 ulasan film untuk klasifikasi sentimen biner (positif atau negatif), dengan 20% data pelatihan digunakan untuk validasi. Metodologi melibatkan perbandingan kinerja model seperti CNN, RNN, LSTM, LSTM-CNN, dan BERT menggunakan metrik *F-Score, Precision, Accuracy,* dan *Recall*. Model CNN dan BERT mencapai akurasi tertinggi 0,90, dengan CNN unggul dalam *Recall* (0,95) dan BERT menunjukkan kinerja seimbang. Model RNN memiliki nilai *Accuracy* terendah yaitu 0,68. Studi ini menyimpulkan bahwa CNN dan BERT adalah model *deep learning* yang berkinerja lebih baik untuk analisis sentimen dan menyoroti potensi serta tantangan analisis sentimen di masa depan, termasuk penanganan multi-bahasa dan isu etika [13].

Tabel 2.1 Literasi Penelitian Terdahulu

No.	Peneliti (tahun)	Judul Penelitian [sitasi]	Permasalahan	Ekstraksi Fitur	Metode Klasifikasi	Hasil Penelitian
1.	B.A. Prameswari *et al* (2023) [6]	*Building Prediction Model for Detecting Cyberbullying using TikTok Comments*	Deteksi cyberbullying pada komentar TikTok dengan arsitektur BERT	IndoBERT	BERT	Akurasi validasi model mencapai 0.63 pada epoch kesembilan
2.	D. Nugraha dan P. Astuti (2023) [10]	Analisis Sentimen *Cyberbullying* Pada Sosial Media Instagram Menggunakan Metode *Support Vector Machine*	Menganalisis sentimen *bullying online* di kolom komentar Instagram	TF-IDF	*Support Vector Machine*	Akurasi sebesar 84,25%, presisi 80,22%, recall 92,50%, dan nilai AUC sebesar 0,928
3.	A.Z. Abdullah *et al.* (2025) [11]	*Sentiment Analysis Accuracy for 2024 Indonesian Election Tweets Using CNN-LSTM With Genetic Algorithm Optimization*	Menganalisis sentimen publik terhadap Tweet Pemilu Presiden Indonesia 2024 untuk wawasan opini publik yang lebih dalam	*Word2Vec*, *TF-IDF*	*Hybrid CNN-LSTM*, *Genetic Algorithm* (optimasi)	Akurasi tertinggi 84,78% untuk data berita (peningkatan 3,59%)

Tabel 2.2 Literasi Penelitian Terdahulu

No.	Peneliti (tahun)	Judul Penelitian [sitasi]	Permasalahan	Ekstraksi Fitur	Metode Klasifikasi	Hasil Penelitian
4.	S.A. Ardiyansa *et al.* (2024) [12]	*Klasifikasi Sentimen Tweet dengan Arsitektur Hybrid Transformers-CNN pada Platform Twitter*	Mengklasifikasi sentimen tweet berbahasa Indonesia dengan akurasi tinggi, mengatasi tantangan volume data besar dan kompleksitas bahasa informal	IndoBERT	*Hybrid Transformer-CNN*	Akurasi 85,71% pada data uji dan 99,90% pada data latih. Konvergen pada iterasi ke-5.
5.	S. Chen (2025) [13]	*Sentiment Analysis Techniques for Deep Learning Classification and Comparison*	Membandingkan performa CNN, RNN, LSTM, LSTM-CNN, dan BERT pada tugas analisis sentimen biner menggunakan dataset IMDb.	Word2Vec	CNN, RNN, LSTM, LSTM-CNN dan BERT	Model CNN secara umum memperoleh *confussion matrix* yang lebih baik dari model lain dengan akurasi 90%, presisi 87%, recall 95%, specificity 84%, dan F1-score 91%

2.2 Dasar Teori

Penelitian ini didasarkan pada beberapa dasar teori yang berperan sebagai acuan dalam proses analisis dan pengembangan. Berikut adalah beberapa dasar teori yang digunakan sebagai acuan dalam pelaksanaan penelitian ini.

2.2.1 *Cyberbullying*

Cyberbullying merupakan bentuk perundungan yang terjadi di ranah digital, dengan pelaku biasanya menyembunyikan identitas di balik akun anonim. Tindakan ini dapat berupa hinaan, ancaman, pelecehan verbal, atau penyebaran informasi yang merugikan korban melalui berbagai media sosial dan platform digital [14]. Berbeda dengan *bullying* yang terjadi secara fisik, *cyberbullying* bersifat persisten, karena jejak digital yang ditinggalkan dapat bertahan dalam waktu lama dan menyebar lebih cepat.

2.2.2 Media Sosial

Media sosial adalah bentuk aplikasi berbasis internet yang memungkinkan penggunanya untuk menciptakan, berbagi, dan bertukar informasi secara interaktif dalam ruang digital [15]. Media sosial merupakan produk dari perkembangan Web 2.0 yang menekankan pada konten yang dihasilkan pengguna (user generated content) dan keterhubungan antarpengguna. Karakteristik utama media sosial meliputi interaktivitas, keterhubungan, serta visibilitas konten, sehingga ia tidak hanya berfungsi sebagai sarana komunikasi, tetapi juga sebagai ruang publik baru yang membentuk pola interaksi, identitas, dan komunitas dalam masyarakat digital.

2.2.2.1 TikTok

Menurut We Are Social, pada tahun 2025 TikTok adalah salah satu platform media sosial yang paling banyak digunakan di indonesia [4]. Platform ini memungkinkan penggunanya untuk mengekspresikan kreativitas dalam pembuatan video dengan berbagai konten. Banyak pengguna, utamanya

generasi muda, menilai TikTok sebagai media hiburan untuk melihat video-video kreatif yang menghibur. Konten yang ditampilkan di TikTok beragam, dan penggunanya dapat mengunggah atau melihat video yang terkadang sesuai umur maupun yang tidak sesuai umur. Sebagai media sosial, TikTok juga melibatkan elemen seperti teks, gambar, dan video [5].

2.2.3 *Natural Language Processing* (NLP)

Natural Language Processing (NLP) adalah cabang dari kecerdasan buatan yang mempelajari bagaimana komputer dapat memahami, memproses, dan menghasilkan bahasa manusia, baik dalam bentuk teks maupun ucapan. NLP menggabungkan teknik dari linguistik, ilmu komputer, dan pembelajaran mesin untuk mengubah data bahasa alami menjadi informasi yang dapat diolah oleh sistem. Ruang lingkupnya mencakup berbagai tugas, seperti penerjemahan otomatis, analisis sentimen, pengenalan suara, ringkasan teks, dan penjawaban pertanyaan. Proses NLP umumnya melibatkan tahapan seperti tokenisasi, penghapusan kata umum (stopword removal), stemming atau lemmatisasi, serta analisis sintaksis dan semantik untuk memahami struktur dan makna bahasa secara lebih mendalam [16].

2.2.4 Analisis Sentimen

Analisis sentimen atau sentiment analysis adalah suatu teknik dalam *Natural Language Processing* (NLP) yang bertujuan untuk mengetahui emosi, sikap, atau opini yang terkandung dalam suatu teks. Teknik ini berperan penting dalam memahami respons pengguna terhadap suatu topik, produk, atau layanan. Analisis sentimen ini dapat menganalisis komentar, ulasan, atau teks yang ditulis oleh pengguna, sistem dapat mengkategorikan apakah teks tersebut bersifat positif, negatif, atau netral [17]. Hasil klasifikasi ini dapat dimanfaatkan untuk pengambilan keputusan, evaluasi layanan, hingga perancangan strategi yang lebih tepat sasaran.

2.2.5 *Text Pre-processing*

Text pre-processing mencakup berbagai teknik untuk meningkatkan kualitas data agar lebih sesuai dalam proses analisis dan pelatihan model.

2.2.5.1 *Normalization*

Normalization adalah proses menyederhanakan dan menyeragamkan teks agar sesuai dengan kaidah bahasa standar. Tahap ini bertujuan untuk mengurangi variasi penulisan yang tidak relevan sehingga analisis data menjadi lebih konsisten dan akurat. Proses normalisasi mencakup berbagai langkah, seperti *case folding* mengubah seluruh huruf kapital menjadi huruf kecil, menghapus karakter khusus yang tidak diperlukan, menghilangkan spasi berlebih, serta mengonversi kata tidak baku atau singkatan menjadi bentuk baku [18]. Dengan melakukan normalisasi, data teks menjadi lebih terstruktur dan siap diproses pada tahap analisis berikutnya.

2.2.5.2 *Augmentation*

Augmentation adalah proses meningkatkan variasi data pelatihan dengan cara menghasilkan data baru yang berasal dari data yang sudah ada. Teknik ini banyak digunakan dalam bidang *machine learning* untuk memperkaya set data dan meningkatkan kemampuan generalisasi model. Pada penelitian kali ini metode augmentasi yang digunakan adalah *An Easy Data Augmentation* (*AEDA*), *random swap*, dan *random delete*. Augmentasi AEDA adalah augmentasi yang bekerja dengan menyisipkan tanda baca ”.”, ”;”, ”?”, ”:”, ”!”,”,” secara acak kedalam teks [19]. *Random swap* dan *random delete* adalah metode augmentasi yang dilakukan dengan menukar huruf dalam satu kata secara acak dan menghapus huruf secara acak dalam teks asli [20].

Dengan menambahkan keragaman data, augmentasi dapat mengurangi risiko *overfitting* dan membantu model belajar pola yang lebih umum, terutama pada kasus di mana set data tidak seimbang atau jumlah data terbatas. Tidak

semua teks input malalui proses augmentasi karena augmentasi memiliki probabilitas 50 persen untuk setiap teks input. Sehingga ada kemungkinan teks input tidak mengalami augmentasi. Jika teks input mengalami augmentasi, maka augmentasi akan dilakukan secara acak diantara augmentasi yang ada sejumlah antara 1 dan jumlah kata dalam teks input yang dibulatkan ke bawah. Ini dapat dirumuskan sebagai berikut:

$$n_{\text{insert}} = \begin{cases} 1, & \text{jika } |W| < 3, \\ \text{RandomInt}\left(1, \left\lfloor \frac{|W|}{3} \right\rfloor\right), & \text{jika } |W| \geq 3. \end{cases}$$

Rumus 2.1 *Augmentation formula*

Keterangan:

n_{insert} = Jumlah augmentasi yang dilakukan

$|W|$ = Jumlah kata dalam teks input

$\left\lfloor \frac{|W|}{3} \right\rfloor$ = Pembulatan ke bawah dari sepertiga jumlah kata dalam teks input

RandomInt(a, b) = Fungsi yang menghasilkan bilangan bulat acak antara a dan b (inklusif)

2.2.5.3 IndoBERT *Tokenizer*

IndoBERT Tokenizer adalah komponen pra-pemrosesan teks yang dikembangkan khusus untuk bahasa Indonesia sebagai bagian dari model IndoBERT. Tokenizer ini berfungsi untuk memecah teks bahasa Indonesia menjadi token-token yang dapat dipahami oleh model, dengan mempertimbangkan karakteristik linguistik khusus bahasa Indonesia [6]. Proses tokenisasi melibatkan pemisahan teks menjadi subkata (subword) menggunakan algoritma WordPiece, yang memungkinkan penanganan kata-kata yang jarang muncul dengan lebih efektif. IndoBERT Tokenizer juga menangani berbagai fitur bahasa Indonesia seperti imbuhan, partikel, dan struktur morfologi yang kompleks, serta mampu mengkonversi teks ke dalam representasi numerik dengan menambahkan token khusus seperti [CLS] di awal

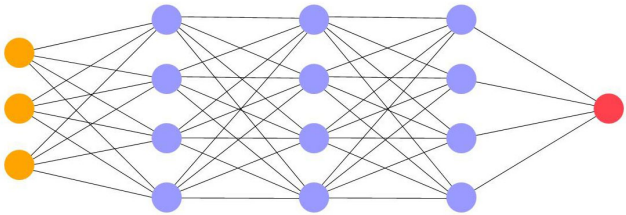
dan [SEP] di akhir kalimat untuk keperluan klasifikasi dan pemisahan segmen teks.

2.2.6 *Stratified K-fold Cross Validation*

Stratified K-fold Cross Validation adalah teknik evaluasi model yang membagi dataset menjadi K subset atau lipatan (folds) dengan mempertahankan proporsi kelas yang sama di setiap lipatan. Metode ini sangat berguna dalam situasi di mana dataset memiliki distribusi kelas yang tidak seimbang, karena memastikan bahwa setiap lipatan mencerminkan distribusi kelas asli dari keseluruhan dataset [21]. Proses ini melibatkan pembagian data menjadi K bagian, di mana pada setiap iterasi, satu lipatan digunakan sebagai data uji sementara sisanya digunakan untuk pelatihan. Setelah K iterasi selesai, hasil evaluasi dari setiap lipatan digabungkan untuk memberikan gambaran yang lebih akurat tentang performa model secara keseluruhan.

2.2.7 *Deep Learning*

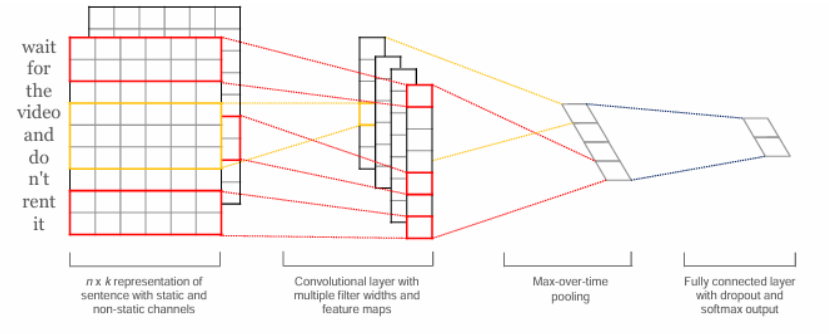
Deep learning adalah pendekatan dalam pembelajaran mesin yang memanfaatkan jaringan saraf tiruan berlapis-lapis untuk mempelajari representasi data secara otomatis. Teknik ini mampu mengenali pola yang kompleks dan abstrak pada berbagai jenis data, termasuk teks, tanpa memerlukan perancangan fitur secara manual [22]. Dalam pemrosesan bahasa alami, deep learning telah melahirkan berbagai arsitektur model yang efektif, seperti Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Transformer, serta varian berbasis Convolutional Neural Network (CNN) seperti TextCNN. Setiap arsitektur memiliki keunggulan masing-masing, mulai dari kemampuan memahami urutan kata, mempertahankan konteks jangka panjang, hingga memproses teks secara paralel untuk efisiensi yang lebih tinggi. Ilustrasi *Deep Learning* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Ilustrasi *Deep Learning*
Sumber: internet

2.2.8 TextCNN

TextCNN adalah varian dari arsitektur *Convolutional Neural Network* (CNN) yang merupkan bagian dari algoritma *deep learning* [23]. CNN adalah algoritma yang digunakan untuk memahami fitur atau karakteristik visual dalam gambar, sehingga dapat membedakan setiap gambar berdasarkan fitur yang sudah dikenali [7]. CNN tidak hanya dapat digunakan pada gambar namun juga dapat digunakan pada teks dengan beberapa penyesuaian seperti perubahan dari konvolusi 2 dimensi menjadi konvolusi 1 dimensi yang merepresentasikan setiap kalimat [23]. Ilustrasi arsitektur model TextCNN dapat dilihat pada Gambar 2.2.



Gambar 2.2 Arsitektur model TextCNN
Sumber: Yoon Kim, 2014 [9]

1. *Embedding Layer*

   Embedding layer berfungsi untuk mengubah setiap kata dalam kalimat menjadi representasi vektor berdimensi tetap. Representasi ini memungkinkan model untuk memahami makna semantik kata dan hubungan antar kata dalam ruang vektor. *Embedding layer* dapat diinisialisasi secara acak di awal dan diperbarui selama proses pelatihan, atau dapat menggunakan *embedding* yang sudah dilatih sebelumnya.

$$X \in \mathbb{R}^{V \times D}$$

Rumus 2.2 *Embedding formula*

   Keterangan:

   X = Matriks embedding

   V = Ukuran *vocabulary*

   D = Dimensi *vektor embedding*

2. *Permute Operation*

   Operasi permutasi dilakukan untuk menyesuaikan dimensi data sebelum memasuki *convolutional layer*. Proses ini mengubah urutan dimensi dari (*batch size, sequence length, embedding dimension*) menjadi (*batch size, embedding dimension, sequence length*). Hal ini penting karena *convolutional layer* dalam TextCNN dioperasikan pada dimensi embedding, sehingga perlu memastikan bahwa dimensi tersebut berada pada posisi yang benar untuk proses konvolusi.

$$X' \in \mathbb{R}^{D \times V}$$

Rumus 2.3 *Permute formula*

   Keterangan:

   X' = Matriks hasil permutasi

   V = Ukuran *vocabulary*

   D = Dimensi *vektor embedding*

3. *Convolutional Layer*

   CNN merupakan jenis *deep learning* yang memanfaatkan *convolutional layer* sebagai penyusun *neural network* yang dibangun. *Convolutional layer* merupakan lapisan pertama dari tahap dalam arsitektur CNN [38]. *Convolutional layer* menggunakan pendekatan *sliding window* dan *weight sharing* untuk menyederhanakan proses perhitungan sehingga mempercepat proses pelatihan.

$$Z[i] = \sum_{j=0}^{k-1} X[i+j] \times W[j] + b$$

Rumus 2.4 *1D Convolution formula*

   Keterangan:

   $Z[i]$ = Hasil konvolusi 1 dimensi pada posisi ke-i

   X = Vektor input 1D

   $X[i+j]$ = Elemen input pada posisi $i+j$

   W = Vektor bobot filter (*kernel*)

   $W[j]$ = Bobot kernel pada posisi ke-j

   k = Ukuran kernel (jumlah elemen filter)

   b = Nilai bias yang ditambahkan setelah operasi konvolusi

4. *Activation Layer* Activation Layer menggunakan fungsi aktivasi *Rectified Linear Unit* (ReLU) yang berfungsi untuk memperkenalkan non-linearitas ke dalam model. Fungsi ReLU bekerja dengan cara mengubah semua nilai negatif menjadi nol, sementara nilai positif tetap dipertahankan. Proses ini membantu jaringan dalam mempelajari pola yang lebih kompleks dan mempercepat konvergensi selama pelatihan model.

$$f(x) = \max(0, x)$$

Rumus 2.5 *ReLu formula*

Keterangan:

$f(x)$ = Keluaran setelah penerapan fungsi ReLU

x = Nilai masukan ke fungsi ReLU

5. *Max Pooling Layer*

Max Pooling Layer adalah komponen yang berfungsi melakukan *down sampling* secara *non-linear* untuk mereduksi ukuran data. Tujuannya adalah mengekstrak informasi yang paling relevan dari *feature map* dengan mengabaikan nilai-nilai yang kurang mempengaruhi hasil prediksi, sehingga beban komputasi dapat berkurang. Pada tugas akhir ini digunakan metode *max* pooling, khususnya *1D max pooling* yang banyak diaplikasikan dalam NLP. *1D Local Max pooling* bekerja dengan memilih nilai maksimum dari setiap area pada *feature map* sedangkan *1D Global Max pooling* memilih nilai maksimum dari seluruh *feature map*.

$$c_i = \max\left(x_{(i-1)s+1\,:\,(i-1)s+k}\right)$$

Rumus 2.6 *1D Local Max Pooling Formula*

Keterangan:

c_i = nilai keluaran pooling pada posisi ke- i

x = vektor input

k = ukuran jendela (*kernel size*) *pooling*

s = *stride*, yaitu jumlah pergeseran jendela setiap langkah

M = jumlah elemen pada output y

$$\hat{c} = \max\{c\}$$

Rumus 2.7 *1D Global Max Pooling Formula*

Keterangan:

$\hat{c}$ = nilai maksimum dari seluruh elemen dalam c

c = vektor input $(c_1, c_2, \ldots, c_n)$

6. *Concatenate Layer*

   Pada *concatenate layer*, fitur-fitur yang telah diekstrak dari berbagai filter dengan ukuran kernel berbeda kemudian digabungkan menjadi satu vektor fitur tunggal. Dengan menggabungkan fitur-fitur tersebut, model dapat memanfaatkan informasi yang lebih kaya dan beragam dari teks input, yang pada akhirnya dapat meningkatkan performa klasifikasi. Vektor fitur gabungan ini kemudian akan diteruskan ke lapisan berikutnya untuk proses klasifikasi lebih lanjut.

$$h = [h_1; h_n]$$

Rumus 2.8 *Concatenate formula*

7. *Fully Connected layer*

   Pada *fully connected layer*, fungsi softmax akan merubah skor menjadi distribusi probabilitas. Proses ini menghitung setiap skor secara eksponensial lalu menormalkannya dengan jumlah seluruh skor. Hasil akhirnya adalah probabilitas untuk tiap kelas dengan total selalu bernilai satu. Fungsi ini menghasilkan total probabilitas setiap kelas sejumlah 1 sehingga efektif untuk klasifikasi multi-kelas.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Rumus 2.9 *Softmax function*

Keterangan:

e = 2.71828 (basis dari logaritma natural)

i = Indeks kelas tertentu

j = Indeks kelas yang dijumlahkan

K = Jumlah total kelas

2.2.9 *Confusion Matrix*

Confusion matrix merupakan salah satu metode evaluasi yang sering digunakan dalam analisis sentimen. Metode ini berbentuk tabel yang berfungsi menilai kinerja model dengan membandingkan hasil prediksi terhadap nilai aktual dari dataset. Pada klasifikasi biner, confusion matrix terdiri dari empat komponen utama, yaitu true positive (TP), true negative (TN), false positive (FP), dan false negative (FN). Berdasarkan confusion matrix, berbagai metrik evaluasi seperti akurasi, presisi, recall, dan F1-score dapat dihitung untuk mengukur performa model secara lebih mendetail.

Tabel 2.3 *Confusion Matrix* untuk Klasifikasi Biner

Aktual	**Prediksi**	
	Positif	**Negatif**
Positif	*True Positive (TP)*	*False Negative (FN)*
Negatif	*False Positive (FP)*	*True Negative (TN)*

Berdasarkan Tabel 2.2, berikut adalah penjelasannya:

1. *True Positive* (TP): Banyaknya data yang memang termasuk kategori positif dan berhasil diidentifikasi dengan benar sebagai positif oleh model.
2. *True Negative* (TN): Banyaknya data yang sebenarnya tergolong negatif dan berhasil dikenali dengan tepat sebagai negatif oleh model.
3. *False Positive* (FP): Banyaknya data yang sebenarnya negatif namun salah diklasifikasikan oleh model sebagai positif.
4. *False Negative* (FN): Banyaknya data yang sebenarnya positif namun keliru diprediksi oleh model sebagai negatif.

Pada penelitian ini digunakan parameter tersebut untuk mengukur *accuracy*, *precission*, *recall* dan f1-score dengan rumus sebagai berikut.

1. *Accuracy*

   Accuracy merupakan ukuran yang menunjukkan sejauh mana model mampu mengklasifikasikan data dengan tepat. Metode ini sering digunakan untuk memberikan gambaran umum tentang kinerja model secara keseluruhan. Nilai *accuracy* dihitung dengan membandingkan jumlah prediksi yang benar dengan total jumlah data uji yang tersedia. Semakin besar persentase *accuracy*, semakin baik pula kemampuan model dalam mengenali dan memprediksi pola dari data yang dianalisis. Klasifikasi biner:

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

Rumus 2.10 *Accuracy Formula*

   Keterangan:

   TP = *True Positive*

   FP = *False Positive*

   TN = *True Negative*

2. *Precision*

   Precision adalah ukuran yang menunjukkan tingkat ketepatan model dalam mengklasifikasikan data positif. Metode ini menilai seberapa banyak prediksi positif yang benar dibandingkan dengan seluruh data yang diprediksi sebagai positif oleh model. Nilai *precision* yang tinggi mengindikasikan bahwa sebagian besar prediksi positif memang benar positif. Dengan demikian, *precision* membantu menilai seberapa andal model dalam menghindari kesalahan klasifikasi positif palsu.

Precision Biner:

$$Precision = \frac{TP}{TP + FP}$$

Rumus 2.11 *Precision Formula*

Keterangan:

TP = *True Positive*

FP = *False Positive*

3. *Recall*

   Recall merupakan ukuran yang menunjukkan kemampuan model dalam mengenali data yang benar pada suatu kelas tertentu. Pengukuran ini menilai sejauh mana model dapat menemukan seluruh data relevan yang termasuk dalam kelas tersebut. Nilainya diperoleh dengan membandingkan jumlah data yang berhasil diklasifikasikan dengan benar dengan total data aktual pada kelas tersebut. Semakin tinggi nilai recall, semakin baik model dalam menangkap seluruh data positif yang seharusnya terdeteksi.

   Recall Biner:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Rumus 2.12 *Recall Formula*

Keterangan:

TP = *True Positive*

FN = *False Negative*

4. F1-*score*

F1-*score* adalah ukuran yang menggabungkan precision dan recall menjadi satu nilai tunggal. Metrik ini menghitung rata-rata tertimbang dari kedua nilai tersebut agar keseimbangan keduanya dapat dinilai. Tujuan F1-*score* adalah untuk memberikan gambaran performa model secara keseluruhan, khususnya ketika ketepatan dan kemampuan deteksi perlu diseimbangkan. Dengan demikian, F1-*score* menjadi indikator penting dalam evaluasi model klasifikasi.

F1-*Score* Biner:

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Rumus 2.13 *F1-Score Formula*

Keterangan:

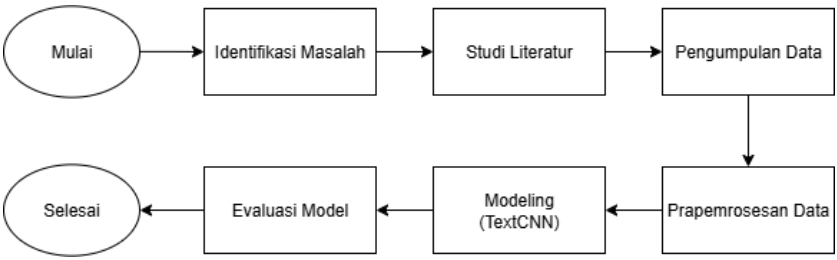
$Precission$ = Tingkat ketepatan prediksi positif yang dihasilkan model

$Recall$ = Tingkat keberhasilan model dalam menemukan seluruh data positif

BAB III
METODE PENELITIAN

3.1 Alur Penelitian

Penelitian ini disusun melalui serangkaian langkah yang terstruktur untuk mencapai tujuan yang telah ditetapkan. Fokus utama penelitian adalah memberikan solusi yang dapat diterapkan terhadap permasalahan yang ada, yaitu dengan merancang dan mengimplementasikan sistem klasifikasi komentar *cyberbullying* pada platform TikTok. Pendekatan yang digunakan mengombinasikan metode *Text Convolutional Neural Network* (TextCNN), di mana TextCNN dimanfaatkan untuk mengekstraksi fitur-fitur penting dari teks komentar, sekaligus mempelajari pola sekuensial dalam teks guna meningkatkan akurasi klasifikasi. Set data pada penelitian ini akan melalui beberapa tahap pemrosesan, meliputi prapemrosesan teks, ekstraksi fitur, serta pelatihan model. Alur lengkap dari proses penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

3.2 Penjabaran Langkah Penelitian

Untuk memperjelas setiap langkah-langkah yang telah didefinisikan pada Gambar 3.1, berikut ini akan dijelaskan secara rinci tahapan-tahapan yang dilakukan dalam penelitian ini.

3.2.1 Identifikasi Masalah

Tahap awal dalam penelitian ini dimulai dengan melakukan identifikasi terhadap permasalahan yang menjadi fokus utama. Penelitian ini bertujuan untuk mengklasifikasikan komentar *cyberbullying* berdasarkan data yang diperoleh dari media sosial *TikTok* dengan menggunakan metode *TextCNN*. Permasalahan utama yang diangkat adalah kesulitan dalam mengenali komentar *cyberbullying* pada kolom komentar *TikTok*, mengingat gaya bahasa yang digunakan di media sosial umumnya bersifat informal, singkat, dan sering kali ambigu.

Seiring meningkatnya penggunaan *TikTok* sebagai sarana berekspresi, risiko munculnya tindakan *cyberbullying* pun semakin besar, sehingga diperlukan sebuah model yang mampu mempelajari pola bahasa untuk melakukan klasifikasi komentar *cyberbullying*. Penelitian ini bertujuan untuk mengembangkan model klasifikasi yang dapat memetakan konten teks ke dalam dua kategori, yaitu *cyberbullying* dan *non-cyberbullying*.

Data yang digunakan pada penelitian ini diperoleh dari penelitian sebelumnya yang dilakukan oleh B.A. Prameswari *et al.*, yang menyediakan kumpulan komentar *TikTok* dengan dua label, yakni *cyberbullying* dan *non-cyberbullying* [6]. Pengembangan model dilakukan menggunakan metode *TextCNN*, yang berperan dalam mengekstraksi fitur penting dari teks dan mempelajari hubungan sekuensial antar kata dalam kalimat untuk kemudian mengklasifikasikannya ke dalam dua label tersebut. Model ini akan dilatih menggunakan data yang telah melalui proses prapemrosesan, sehingga diharapkan dapat memberikan prediksi yang lebih akurat dalam mengklasifikasikan komentar *cyberbullying* berdasarkan pola bahasa yang digunakan oleh pengguna *TikTok*.

3.2.2 Tinjauan Pustaka

Pada tahap ini, dilakukan tinjauan terhadap berbagai penelitian terdahulu yang membahas klasifikasi komentar *cyberbullying* berbasis media sosial, serta penelitian lain yang menggunakan metode serupa, khususnya dengan penerapan model *TextCNN*. Tinjauan ini melibatkan beragam sumber referensi, termasuk *paper*, jurnal, dan laporan penelitian yang relevan. Hasil dari studi literatur tersebut menjadi landasan penting bagi peneliti dalam memahami konsep, merancang metodologi, serta memperkuat kerangka teori yang mendasari penelitian ini.

3.2.3 Pengumpulan Dataset

Pada penelitian ini, proses pengumpulan data tidak dilakukan secara langsung, melainkan menggunakan *dataset* sekunder yang diperoleh dari penelitian sebelumnya oleh Prameswari *et al.* [6]. *Dataset* tersebut terdiri atas 1.510 baris data dengan dua kolom utama, yaitu kolom *sentiment* yang memuat dua label enumerasi: -1 untuk kategori *cyberbullying* dan 1 untuk kategori *non-cyberbullying*, serta kolom *comment* yang berisi teks komentar. Contoh data pada Tabel 3.1 diambil dari berbagai baris data untuk representasi data yang lebih baik. Data ini selanjutnya akan diolah dan dimanfaatkan sebagai dasar dalam proses pelatihan model untuk melakukan klasifikasi emosi pada unggahan di media sosial.

Tabel 3.1 Sample Isi Dataset

No Baris	Sentiment	Comment
1	-1	Makannya segentong buset
2	-1	Mirip ursula di little mermaid
474	1	Pede dulu, glow up belakangan
538	1	Dihh keren banget

No Baris	Sentiment	Comment
550	1	Pentingnya peradaban juga sis

3.2.4 Prapemrosesan Data

Data dari *dataset* yang diperoleh melalui sumber sekunder diproses terlebih dahulu melalui tahap prapemrosesan sebelum dilakukan analisis. Tahap prapemrosesan ini mencakup serangkaian teknik yang bertujuan untuk meningkatkan kualitas data sehingga lebih optimal digunakan dalam proses analisis dan pelatihan model. Beberapa langkah yang dilakukan meliputi *case folding* untuk mengonversi seluruh teks menjadi huruf kecil, *text cleaning* untuk menghapus karakter atau simbol yang tidak diperlukan, serta *stopword removal* guna menghilangkan kata-kata yang tidak memberikan makna signifikan pada analisis. Setelah tahap prapemrosesan selesai, data kemudian dibagi menjadi dua bagian, yaitu 80% untuk pelatihan (*training*) dan 20% untuk pengujian (*testing*).

3.2.5 *Word Embedding*

Data yang telah melewati tahap prapemrosesan selanjutnya diproses menggunakan teknik *word embedding* untuk mengubah teks menjadi representasi numerik yang dapat dipahami oleh model pembelajaran mesin. Pada penelitian ini, *word embedding* diimplementasikan menggunakan pustaka PyTorch. Lapisan *Embedding* ini bekerja dengan memetakan setiap kata dalam *dataset* ke dalam representasi vektor berdimensi tetap berdasarkan indeks kata.

Proses dimulai dengan membangun *vocabulary*, yaitu daftar kata unik yang diperoleh dari *dataset*. Setiap kata dalam *vocabulary* diberikan indeks numerik, kemudian indeks tersebut dimasukkan ke dalam lapisan *Embedding* untuk mendapatkan representasi vektor. Bobot awal pada lapisan *Embedding* diinisialisasi secara acak dan akan diperbarui selama proses pelatihan model, sehingga representasi vektor yang dihasilkan dapat menyesuaikan dengan pola

data dan meningkatkan kinerja model dalam klasifikasi.

3.2.6 *Modelling* (TextCNN)

Pada tahap ini, dilakukan proses klasifikasi terhadap data yang telah melewati tahap ekstraksi fitur agar model dapat mengenali pola-pola penting yang berkaitan dengan klasifikasi emosi. Proses klasifikasi dilakukan menggunakan arsitektur *TextCNN*, yang dirancang khusus untuk menangani data teks.

TextCNN bekerja melalui beberapa tahapan utama. Pertama, dilakukan proses *convolution* menggunakan beberapa filter dengan ukuran kernel yang berbeda untuk mengekstraksi berbagai fitur penting dari data teks. Setiap filter bertujuan untuk menangkap pola kata dan frasa yang memiliki peran signifikan dalam penentuan kelas. Selanjutnya, hasil konvolusi melewati proses *max pooling* yang berfungsi untuk mereduksi dimensi fitur tanpa kehilangan informasi esensial, sehingga membuat model lebih efisien dan fokus pada fitur paling relevan.

Setelah tahap *pooling*, hasilnya digabungkan dan diteruskan ke lapisan *fully connected* untuk menghasilkan prediksi akhir. Dataset dibagi menjadi dua bagian, yaitu *training data* yang digunakan untuk melatih model, serta *testing data* yang digunakan untuk mengevaluasi kinerja model dalam mengklasifikasikan data baru.

3.2.7 Evaluasi Model

Setelah model berhasil dibangun, tahap berikutnya adalah melakukan proses evaluasi untuk menilai kinerjanya dalam mengklasifikasikan emosi pada data berbasis media sosial. Evaluasi dilakukan dengan menghitung beberapa metrik utama, yaitu *accuracy*, *precision*, *recall*, dan *F1-score*.

Accuracy digunakan untuk mengukur sejauh mana model mampu mengklasifikasikan data secara benar secara keseluruhan. *Precision* menilai

tingkat ketepatan model dalam memprediksi data yang termasuk dalam kategori tertentu, sedangkan *recall* mengukur sejauh mana model dapat menangkap seluruh data yang seharusnya termasuk dalam kategori tersebut. Sementara itu, *F1-score* merupakan nilai harmonisasi antara *precision* dan *recall*, sehingga memberikan gambaran yang lebih seimbang mengenai performa model.

3.3 Alat dan Bahan Tugas Akhir

Adapun alat dan bahan yang digunakan dalam penelitian ini adalah sebagai berikut:

3.3.1 Alat

Berikut adalah alat yang digunakan dalam penelitian ini:

1. Pada penelitian ini penulis menggunakan laptop dengan spesifikasi Sistem Operasi Windows 11 Home, AMD Ryzen 5 5600H, Memori 24GB DDR4, SSD 500GB.
2. *Visual Studio Code* v1.101.2, Miniconda v24.5.0, Python v3.12.4

3.3.2 Bahan

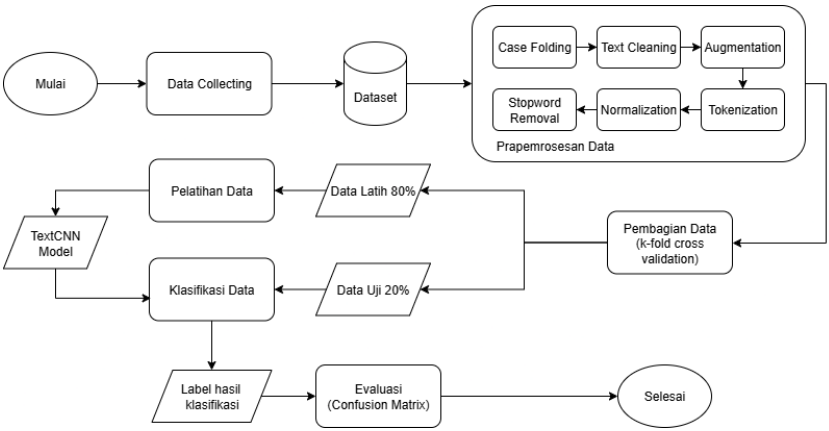
Berikut adalah bahan yang digunakan dalam penelitian ini:

1. *Dataset* yang digunakan dalam penelitian ini merupakan *dataset* sekunder yang diambil dari penelitian yang dilakukan oleh B.A. Prameswari *el al*., dengan judul *Building Prediction Model for Detecting Cyberbullying using TikTok Comments* [6].
2. Jurnal penelitian dari studi sebelumnya digunakan sebagai acuan untuk memberikan landasan teori, serta menyusun konsep dan gagasan yang mendukung pelaksanaan penelitian ini.

3.4 Metode Pengembangan Model

Tahapan yang akan dilakukan dalam proses membangun model klasifikasi emosi menggunakan TextCNN dan hasil modifikasinya seperti SEDepthwise

TextCNN dapat dilihat pada Gambar 3.2.



Gambar 3.2 Metode Pengembangan

3.4.1 Pengumpulan Data

Dataset yang digunakan pada penelitian ini terdiri dari 1.510 baris komentar TikTok yang diklasifikasikan ke dalam dua kategori sentimen, yaitu *cyberbullying* dan *non-cyberbullying*. Dataset tersebut disajikan dalam format berkas *CSV*. Setiap baris data berisi komentar *TikTok* beserta label sentimen dalam bentuk nilai enumerasi, di mana nilai *-1* menunjukkan *cyberbullying* dan nilai *1* menunjukkan *non-cyberbullying*. Gambaran umum mengenai dataset dapat dilihat pada Tabel 3.2 yang menampilkan contoh data awal.

Tabel 3.2 Sample Dataset Awal

Sentiment	***Comment***
-1	Makannya segentong buset
-1	Mirip ursula di little mermaid
...	...
1	Pentingnya peradaban juga sis
1	Pede dulu, glow up belakangan

3.4.2 Validasi Dataset

Validasi dataset merupakan proses pemeriksaan dan evaluasi untuk memastikan kualitas serta kelayakan data yang digunakan dalam penelitian. Tahap ini bertujuan untuk menjamin bahwa dataset bebas dari kesalahan yang berpotensi memengaruhi hasil analisis maupun kinerja model. Validasi dilakukan dengan cara memeriksa keberadaan data yang duplikat, hilang, atau tidak konsisten. Data yang duplikat dapat menyebabkan bias dalam pelatihan model, sehingga perlu diidentifikasi dan dihapus. Data yang hilang atau kosong juga perlu ditangani, baik dengan menghapus baris tersebut atau mengisi nilai yang sesuai berdasarkan konteks data. Selain itu, validasi juga mencakup pemeriksaan format data untuk memastikan kesesuaian dengan standar yang telah ditetapkan. Dengan melakukan validasi dataset secara menyeluruh, diharapkan data yang digunakan dalam penelitian ini memiliki kualitas yang baik dan dapat memberikan hasil analisis yang akurat serta dapat diandalkan.

3.4.3 Prapemrosesan Data

Prapemrosesan data merupakan serangkaian tahapan yang dirancang untuk mempersiapkan serta meningkatkan kualitas data teks agar siap digunakan pada proses analisis lebih lanjut. Tahapan ini umumnya mencakup konversi seluruh huruf menjadi huruf kecil (*case folding*), penghapusan karakter atau simbol yang tidak diperlukan (*text cleaning*), penambahan variasi data melalui proses augmentasi (*data augmentation*), pemecahan teks menjadi satuan-satuan kata atau token (*tokenization*), serta penghapusan kata-kata yang kurang relevan terhadap analisis (*stopword removal*).

1. *Case Folding*

   Case folding merupakan proses penyamaan format huruf pada teks dengan cara mengonversi seluruh karakter berhuruf besar (*uppercase*) menjadi huruf kecil (*lowercase*).

Tabel 3.3 Tahapan *Case Folding*

Sentiment	**Comment**	**Hasil**
-1	Cowo paling alay lebay se antero selebgram WKWKW NGAKAK	cowo paling alay lebay se antero selebgram wkwk ngakak

2. *Text Cleaning*

   Text Cleaning merupakan proses pembersihan teks dengan cara menghilangkan kata, karakter, atau elemen yang tidak diperlukan. Pada tahap ini, komponen yang dianggap tidak relevan atau dapat mengganggu analisis, seperti simbol, emoji, dan *hashtag*, akan dihapus dari teks.

Tabel 3.4 Tahapan *Text Cleaning*

Sentiment	**Comment**	**Hasil**
-1	udah sok tau salah pula @Ibnu Wardani ????	udah sok tau salah pula ibnu wardani ????

3. *Augmentation*

   Augmentation adalah proses meningkatkan variasi data pelatihan dengan cara menghasilkan data baru yang berasal dari data yang sudah ada. Dalam penelitian ini, metode augmentasi yang digunakan adalah *An Easier Data Augmentation* (AEDA), *random swap*, dan *random delete*. AEDA adalah metode augmentasi yang dilakukan dengan menyisipkan tanda baca ”.”, ”;”, ”?”, ”:”, ”!”,”,” secara acak kedalam teks asli [19]. Random swap dan random delete adalah metode augmentasi yang dilakukan dengan menukar huruf dalam satu kata secara acak dan menghapus huruf secara acak dalam teks asli [20]. Tujuan dari *augmentation* adalah meningkatkan variasi data sehingga model dapat belajar lebih baik dan menghasilkan performa klasifikasi yang lebih akurat.

Tabel 3.5 Tahapan *Augmentation*

Sentiment	**Comment**	**Hasil**	**Kata/Frasa yang Diubah**	
			Kata Asli	**Kata Hasil**
-1	ngeliat mukanya aja udah gedeg	ngeliat mukanya aja udah kesal	ngeliat mukanya aja udah gedeg	ngeliat mukanya aja, udah gedeg!
1	gue gapaham dr dulu fashion style dia kyk gmna wkwkwk	gue gapaham dr duul fashion style dia kyk gmna wkwkwk	dulu	duul
-1	sebelum mangap di sikat dulu kale	sebelm mangap di sikat dulu kale	sebelum	sebelm

4. *Tokenization*

   Tokenization merupakan tahap pemecahan teks menjadi unit-unit terkecil, seperti kata, frasa, atau kalimat. Tahapan ini berperan penting dalam membantu sistem komputer memahami struktur dan makna dari teks yang sedang dianalisis. Pada penelitan kali ini, proses *tokenization* akan menggunakan *tokenizer* IndoBERT yang telah dilatih sebelumnya untuk bahasa Indonesia. *Tokenizer* ini mampu memecah teks menjadi token-token yang sesuai dengan konteks bahasa Indonesia, sehingga memudahkan model dalam memahami dan memproses data teks.

Tabel 3.6 Tahapan *Tokenization*

Sentiment	**Comment**	**Hasil**
1	aku penggemar mi burung dara enaknya nyambung terus	’aku’, ’penggemar’, ’mi’, ’burung’, ’dara’, ’enaknya’, ’nyambung’, ’terus’

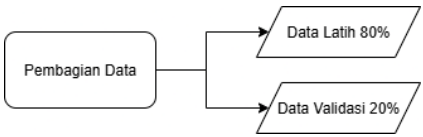
5. *Stopword Removal*

Stopword removal merupakan proses penghapusan kata-kata yang sering muncul dalam jumlah banyak namun dianggap tidak memiliki makna penting. Kata-kata yang umum tetapi kurang relevan, seperti konjungsi, kata kepunyaan, dan kata ganti orang, akan dihapus dari teks. Pada penelitian kali ini, proses *stopword removal* dilakukan menggunakan daftar *stopword* bahasa Indonesia yang telah disediakan oleh NLTK (*Natural Language Toolkit*).

Tabel 3.7 Tahapan *Stopword Removal*

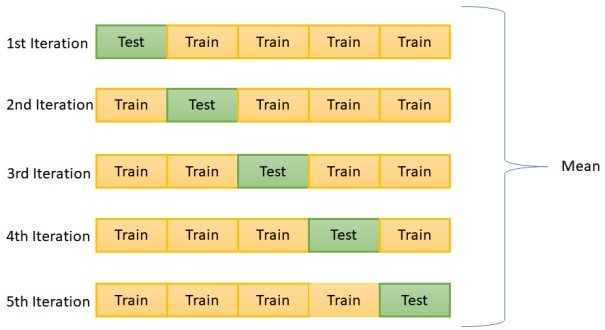
Sentiment	**Comment**	**Hasil**	**Kata yang Dihapus**
1	buatlah cita cita kamu setinggi bintang di langit tetapi jangan lupa untuk membuat anak tangga perencanaannya	buatlah cita cita setinggi bintang langit jangan lupa membuat anak tangga perencanaannya	’kamu’, ’di’, ’tetapi’, ’untuk’

3.4.4 Pembagian Data



Gambar 3.3 Pembagian Data

Pada Gambar 3.3 menunjukkan pembagian dataset dilakukan dengan rasio 80% untuk data latih dan 20% untuk data validasi. Data latih digunakan untuk melatih model, sementara data validasi berfungsi untuk menguji kinerja model yang telah dilatih. Pembagian data menggunakan *stratified k-fold cross-validation* dapat diperoleh evaluasi yang lebih akurat mengenai seberapa baik model dapat melakukan generalisasi terhadap data yang belum pernah dilihat sebelumnya dan menghindari overfitting.



Gambar 3.4 Arsitektur model TextCNN
Sumber: internet

Ilustrasi pembagian dataset dengan teknik *stratified k-fold cross validation* dapat dilihat pada Gambar 3.4. Keseluruhan dataset terdiri dari data komentar TikTok yang dibagi menjadi 5 *fold*, di mana setiap *fold* berisi 20% dari keseluruhan data. Lima *fold* tersebut kemudian disusun menjadi 5 variasi *split*, yang masing-masing terdiri atas 80% data pelatihan dan 20% data evaluasi.

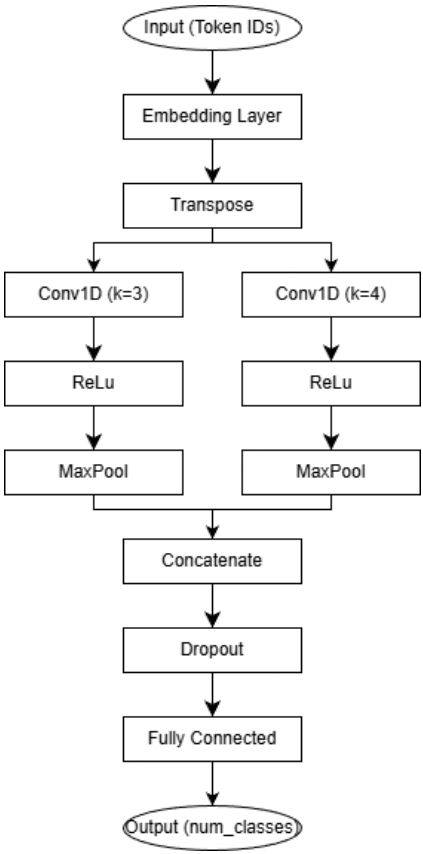
Setiap satu kali iterasi, proses pelatihan dilakukan sebanyak 5 kali menggunakan 5 *fold* yang berbeda, dengan setiap *fold* melatih model yang berbeda pula. Dengan demikian, pada proses pelatihan menggunakan *k*-fold ini dihasilkan 5 model yang telah dilatih menggunakan 5 variasi pembagian dataset.

3.4.5 Arsitektur Model

Arsitektur model yang digunakan dalam penelitian ini adalah *Text Convolutional Neural Network* (TextCNN) dan SEDepthwise TextCNN. Kedua model ini dirancang untuk menangani data teks dan melakukan klasifikasi berdasarkan fitur-fitur yang diekstraksi dari teks tersebut.

3.4.5.1 Arsitektur Model TextCNN

TextCNN merupakan model klasifikasi teks berbasis algoritma *Convolutional Neural Network* (CNN) yang dirancang khusus untuk memproses data teks. Diagram alur dari arsitektur TextCNN dapat dilihat pada Gambar 3.5.



Gambar 3.5 Pembagian Data

Gambar 3.5 menunjukkan rancangan arsitektur model dasar TextCNN yang terdiri atas embedding layer, conv1d layer, ReLU, max pooling layer, concatenate layer, dropout layer, dan fully connected layer.

1. *Embedding Layer*

   Embedding layer berfungsi untuk mengonversi indeks kata menjadi vektor berdimensi kontinu. Pada tahap awal, parameter *embedding* diinisialisasi secara acak. Selama proses pelatihan, parameter tersebut diperbarui melalui mekanisme *backpropagation* secara end-to-end bersamaan dengan pelatihan model utama. Dengan demikian,

representasi kata yang dihasilkan menjadi lebih kontekstual dan mampu menyesuaikan dengan karakteristik data yang digunakan. Pada penelitian ini, digunakan modul *torch.nn.Embedding* dari *PyTorch* untuk membangun representasi kata.

2. *Conv1D Convolutional Layer*

   Conv1D adalah operasi konvolusi yang banyak digunakan pada data sekuensial, seperti teks. Pada penelitian ini, Conv1D digunakan untuk mengekstraksi fitur penting dari data teks yang telah diubah menjadi representasi vektor melalui *embedding layer*. Proses konvolusi dilakukan dengan menerapkan beberapa filter (kernel) berukuran berbeda pada data input. Setiap filter bertujuan untuk menangkap pola kata dan frasa yang memiliki peran signifikan dalam penentuan kelas. Dengan menggunakan berbagai ukuran kernel, model dapat mengenali beragam pola dalam teks yang mungkin relevan untuk klasifikasi.

3. *Activation Layer (ReLu)*

   Activation Layer (ReLu) berfungsi untuk memperkenalkan non-linearitas ke dalam model setelah tahap konvolusi. Fungsi aktivasi ReLu (Rectified Linear Unit) digunakan untuk mengubah nilai negatif menjadi nol, sementara nilai positif tetap dipertahankan. Dengan demikian, ReLu membantu model untuk belajar representasi yang lebih kompleks dan mampu menangkap pola-pola non-linear dalam data teks. Penggunaan ReLu juga memiliki keuntungan dalam hal efisiensi komputasi, karena fungsi ini sederhana dan cepat untuk dihitung.

4. *Max Pooling Layer*

   Max Pooling Layer berfungsi untuk mereduksi dimensi fitur yang dihasilkan dari tahap konvolusi. Proses ini dilakukan dengan cara mengambil nilai maksimum dari setiap jendela (window) yang bergerak melintasi fitur yang dihasilkan oleh filter konvolusi. Dengan demikian, max pooling membantu mengurangi jumlah parameter dalam model,

sehingga meningkatkan efisiensi komputasi dan mengurangi risiko overfitting. Selain itu, max pooling juga menyoroti fitur-fitur paling dominan yang dianggap paling relevan untuk klasifikasi, sehingga model dapat fokus pada informasi penting yang diperlukan untuk membuat prediksi.

5. *Concatenate Layer*

   Concatenate Layer berfungsi untuk menggabungkan hasil dari beberapa filter konvolusi yang telah melalui tahap *max pooling*. Setiap filter dengan ukuran kernel yang berbeda akan mengekstraksi fitur yang berbeda pula dari data teks. Dengan menggabungkan hasil dari berbagai filter, model dapat memperoleh representasi fitur yang lebih kaya dan komprehensif. Proses *concatenate* ini memungkinkan model untuk memanfaatkan informasi dari berbagai perspektif, sehingga meningkatkan kemampuan dalam mengenali pola-pola penting yang berkaitan dengan klasifikasi.

6. *Dropout Layer*

   Dropout Layer berfungsi sebagai teknik regulasi untuk mencegah overfitting pada model. Pada tahap ini, sejumlah neuron dipilih secara acak dan ”dijatuhkan” atau diabaikan selama proses pelatihan. Dengan demikian, model tidak terlalu bergantung pada neuron-neuron tertentu, sehingga dapat belajar representasi yang lebih umum dan *robust*. Penggunaan *dropout* membantu meningkatkan kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya, sehingga kinerja klasifikasi menjadi lebih baik.

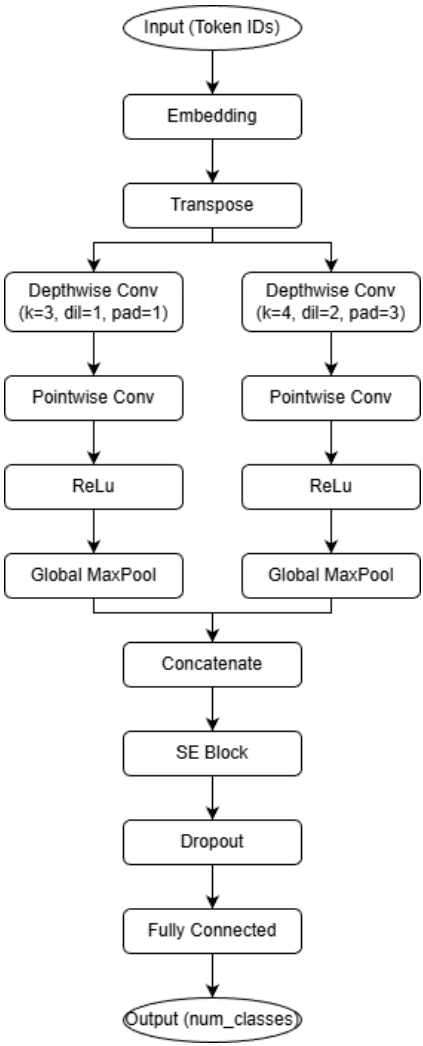
7. *Fully Connected Layer*

   Fully Connected Layer berfungsi untuk menggabungkan fitur-fitur yang telah diekstraksi dari tahap sebelumnya dan menghasilkan prediksi akhir. Pada tahap ini, semua neuron dari lapisan sebelumnya terhubung secara penuh ke setiap neuron di lapisan ini. Dengan demikian, model dapat memanfaatkan seluruh informasi yang telah dipelajari untuk membuat

keputusan klasifikasi. Hasil dari *fully connected layer* kemudian diteruskan ke fungsi aktivasi (seperti *softmax* atau *sigmoid*) untuk menghasilkan probabilitas prediksi terhadap kelas yang ditentukan.

3.4.5.2 Arsitektur Model SEDepthwise TextCNN

SEDepthwise TextCNN merupakan model klasifikasi teks yang didasarkan pada TextCNN asli menggunakan konvolusi terpisah dan squeeze-and-excitation (SE) blok untuk menangkap fitur yang lebih kaya dan relevan dari data teks. Diagram alur dari arsitektur SEDepthwise TextCNN dapat dilihat pada Gambar 3.6.



Gambar 3.6 Pembagian Data

Gambar 3.6 menunjukkan rancangan arsitektur model SEDepthwise TextCNN yang terdiri atas embedding layer, depthwise conv1d layer, pointwise conv1d layer, squeeze-and-excitation (SE) block, ReLU, max pooling layer, concatenate layer, dropout layer, dan fully connected layer.

1. *Depthwise Conv1D Layer*

Depthwise Conv1D Layer adalah varian dari konvolusi satu dimensi (Conv1D) yang memisahkan proses konvolusi untuk setiap saluran input secara independen. Pada tahap ini, setiap filter diterapkan pada masing-masing saluran input tanpa menggabungkan informasi antar saluran. Dengan demikian, depthwise conv1d memungkinkan model untuk mengekstraksi fitur yang lebih spesifik dari setiap saluran, sehingga meningkatkan efisiensi komputasi dan mengurangi jumlah parameter yang perlu dipelajari. Pendekatan ini membantu model dalam menangkap pola-pola penting dalam data teks dengan cara yang lebih terfokus dan hemat sumber daya.

2. *Pointwise Conv1D Layer*

   Pointwise Conv1D Layer adalah tahap konvolusi satu dimensi (Conv1D) yang menggunakan kernel berukuran 1. Pada tahap ini, setiap posisi dalam urutan data teks diproses secara individual, tanpa mempertimbangkan konteks dari posisi-posisi sekitarnya. Dengan demikian, pointwise conv1d memungkinkan model untuk menggabungkan informasi dari berbagai saluran yang telah diekstraksi pada tahap depthwise conv1d sebelumnya. Pendekatan ini membantu dalam mengintegrasikan fitur-fitur yang telah dipelajari dari setiap saluran, sehingga menghasilkan representasi yang lebih kaya dan komprehensif untuk klasifikasi teks.

3. *Squeeze-and-Excitation (SE) Block*

   Squeeze-and-Excitation (SE) Block adalah mekanisme yang dirancang untuk meningkatkan representasi fitur dengan cara menyesuaikan bobot saluran fitur berdasarkan pentingnya masing-masing saluran. Pada tahap ini, fitur-fitur yang telah diekstraksi dari tahap konvolusi diproses melalui dua langkah utama: *squeeze* dan *excitation*. Pada langkah *squeeze*, informasi spasial dari fitur-fitur tersebut diringkas menjadi vektor satu dimensi yang merepresentasikan setiap saluran. Selanjutnya, pada

langkah excitation, vektor tersebut digunakan untuk menghasilkan bobot yang menyesuaikan kontribusi masing-masing saluran fitur. Dengan demikian, SE block memungkinkan model untuk fokus pada saluran-saluran yang lebih relevan, sehingga meningkatkan kemampuan dalam menangkap pola-pola penting yang berkaitan dengan klasifikasi teks.

3.5 Ilustrai Perhitungan Metode

Pada tahap ini, dilakukan ilustrasi perhitungan metode *TextCNN* yang digunakan dalam penelitian ini. Ilustrasi ini mencakup langkah-langkah utama dalam proses klasifikasi teks, mulai dari representasi kata hingga prediksi akhir.

3.5.1 *Input Layer (Token IDs)*

Setelah melalui tahapan pemrosesan data, teks input diubah menjadi representasi numerik berupa token ID melalui *tokenizer* IndoBERT. Ini dilakukan agar teks dapat diproses oleh model pembelajaran mesin. Berikut ini adalah tabel yang menunjukkan contoh teks input beserta token ID yang dihasilkan.

Tabel 3.8 Sample Dataset Awal

Token ID	***Comment***
245	Makannya
...	...
678	Segentong
...	...
934	Buset

Dari token ID yang dihasilkan oleh proses tokenisasi, kemudian teks input berupa kalimat ”Makannya segentong buset” dipetakan menjadi token ID sebagai berikut:

$$\text{”Makannya segentong buset”} \quad \mapsto \quad [245, 678, 934]$$

3.5.2 *Embedding Layer*

Proses *word embedding* adalah proses awal dimana token ID diubah menjadi representasi vektor berdimensi tetap. Misalnya, token ID *245*, *678*, dan *934* dipetakan ke dalam vektor embedding berdimensi 4 sebagai berikut:

$$\mathbf{x}_1 = W_{245} = [0.1,\ 0.3,\ 0.5,\ 0.7]$$

$$\mathbf{x}_2 = W_{678} = [0.2,\ 0.4,\ 0.6,\ 0.8]$$

$$\mathbf{x}_3 = W_{934} = [0.3,\ 0.5,\ 0.7,\ 0.9]$$

Hasil embedding seluruh token ID akan membentuk matriks:

$$E \in \mathbb{R}^{3\times 4}, \quad E = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.3 & 0.5 & 0.7 \\ 0.2 & 0.4 & 0.6 & 0.8 \\ 0.3 & 0.5 & 0.7 & 0.9 \end{bmatrix}$$

3.5.3 *Permute Operation*

Agar sesuai dengan format Conv1D PyTorch, dilakukan transpose matriks embedding untuk merubah bentuk matrix dari (*vocabulary size, embedding dimmention*) menjadi (*embedding dimmention, vocabulary size*).

$$E' \in \mathbb{R}^{4\times 3}, \quad E' = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.3 & 0.4 & 0.5 \\ 0.5 & 0.6 & 0.7 \\ 0.7 & 0.8 & 0.9 \end{bmatrix}$$

3.5.4 *Convolutional Layer*

3.5.4.1 *Conv1D*

Pada layer konvolusi, dilakukan operasi konvolusi pada matriks embedding yang telah ditranspose. Perhitungan konvolusi pada posisi i dilakukan dengan rumus:

$$Z[i] = \sum_{j=0}^{k-1} X[i+j] \times W[j] + b$$

Misalkan pada tahap konvolusi dengan kernel size 2 setiap filter memiliki bobot sebagai berikut:

$$W = \begin{bmatrix} 0.2 & 0.1 & -0.1 & 0.5 \\ 0.3 & -0.2 & 0.4 & 0.1 \end{bmatrix}$$

Dengan asumsi:

kernel size = 2

stride = 1

bias = 0.2

1. Window 1: Makannya + segentong

$$\begin{aligned}\text{output} &= (0.1)(0.2) + (0.3)(0.1) + (0.5)(-0.1) + (0.7)(0.5) + \\ &\quad (0.2)(0.3) + (0.4)(-0.2) + (0.6)(0.4) + (0.8)(0.1) \\ &= 0.65 + 0.2 \text{ (bias)} \\ &= 0.85\end{aligned}$$

2. Window 2: segentong + buset

$$\begin{aligned}\text{output} &= (0.2)(0.2) + (0.4)(0.1) + (0.6)(-0.1) + (0.8)(0.5) + \\ &\quad (0.3)(0.3) + (0.5)(-0.2) + (0.7)(0.4) + (0.9)(0.1) \\ &= 0.68 + 0.2 \text{ (bias)} \\ &= 0.88\end{aligned}$$

Misalkan pada tahap konvolusi dengan kernel size 3 setiap filter memiliki bobot sebagai berikut:

$$W = \begin{bmatrix} 0.2 & 0.1 & -0.1 & 0.5 \\ 0.3 & -0.2 & 0.4 & 0.1 \\ 0.4 & 0.3 & 0.2 & 0.2 \end{bmatrix}$$

Dengan asumsi:

kernel size = 3

stride = 1

bias = 0.2

1. Window 1: Makannya + segentong + buset

$$\begin{aligned}\text{output} &= (0.1)(0.2) + (0.3)(0.1) + (0.5)(-0.1) + (0.7)(0.5) + \\ &\quad (0.2)(0.3) + (0.4)(-0.2) + (0.6)(0.4) + (0.8)(0.1) \\ &\quad (0.3)(0.4) + (0.5)(0.3) + (0.7)(0.2) + (0.9)(0.2) \\ &= 0.68 + 0.2 \text{ (bias)} \\ &= 0.88\end{aligned}$$

3.5.5 *Activation Layer (ReLU)*

Untuk mengeliminasi nilai negatif dan mempertahankan nilai positif, hasil konvolusi $Z[i]$ dilewatkan ke fungsi aktivasi ReLU:

$$c_i = f(Z[i]) = \max(0, Z[i])$$

Contoh hasil *feature map* setelah ReLU:

$$C_1 = [0.85,\ 0.88]$$

$$C_2 = [0.88]$$

3.5.6 *Max Pooling Layer*

Ambil nilai maksimum dari setiap *feature map* untuk merangkum informasi paling penting:

$$\text{MaxPool}(C_1) = \max(0.85, 0.88) = 0.88$$

$$\text{MaxPool}(C_2) = \max(0.88) = 0.88$$

Sehingga output pooling adalah:

$$\begin{bmatrix} 0.88 \\ 0.88 \end{bmatrix}$$

3.5.7 *Concatenation Layer*

Gabungkan hasil pooling dari semua filter menjadi satu vektor fitur gabungan:

$$\text{Fitur Gabungan} = \begin{bmatrix} 0.88 \\ 0.88 \end{bmatrix} = [0.88, 0.88]$$

Jika ada lebih banyak filter, tambahkan hasil pooling mereka ke vektor ini.

3.5.8 *Dropout Layer*

Dengan dropout rate 0.1 (10% neuron dimatikan acak saat training).

$$\text{Fitur Gabungan Setelah Dropout} = [0.88, 0.88]$$

3.5.9 *Fully Connected Layer*

Fully connected layer mengubah vektor fitur gabungan menjadi skor untuk setiap kelas. Fungsi aktivasi *softmax* kemudian digunakan untuk mengubah skor ini menjadi probabilitas prediksi untuk setiap kelas.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Misalnya hasil perhitungan adalah:

$$\text{Output} = \begin{bmatrix} 1.086 \\ 0.888 \end{bmatrix}$$

Maka probabilitas prediksi untuk setiap kelas adalah:

$$P(\text{Kelas 1}) = \frac{e^{1.086}}{e^{1.086} + e^{0.888}} \approx 0.57$$
$$P(\text{Kelas 2}) = \frac{e^{0.888}}{e^{1.086} + e^{0.888}} \approx 0.43$$

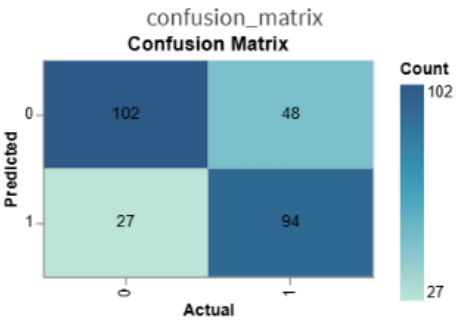
3.6 Rancangan Pengujian

Rancangan pengujian dilakukan secara sistematis agar hasil evaluasi bersifat andal, terukur, dan dapat direplikasi.

3.6.1 Skema Pengujian

Pada tahap ini, dilakukan proses pengujian untuk menilai kinerja model yang telah dibangun. Pengujian dilakukan dengan menggunakan 5 fold data yang telah dipisahkan sebelumnya sebagai data latih dan data validasi. Hasil prediksi dari model akan dibandingkan dengan label asli untuk menghitung metrik evaluasi seperti *accuracy* yang dihitung berdasarkan rumus Rumus 2.10, *precision* yang dihitung berdasarkan rumus Rumus 2.11, *recall* yang dihitung berdasarkan rumus Rumus 2.12, dan *F1-score* yang dihitung berdasarkan rumus Rumus 2.13. Metrik-metrik ini memberikan gambaran mengenai seberapa baik

model dalam mengklasifikasikan data baru dan membantu mengidentifikasi area yang perlu ditingkatkan. Disini diasumsikan *confusion matrix* seperti pada Gambar 3.7 dan Tabel 3.9.



Gambar 3.7 Inisialisasi *Confusion Matrix*

Tabel 3.9 Inisialisasi *Confusion Matrix*

Aktual	**Prediksi**		
	Positif	**Negatif**	**Total Prediksi**
Positif	94	48	142
Negatif	27	102	129
Total Aktual	121	150	271

Berdasarkan *confusion matrix* pada Gambar 3.7 dan Tabel 3.9 dapat ditarik beberapa kesimpulan. Dari total 142 data dengan label asli *Cyberbullying*, sebanyak 94 data berhasil diklasifikasikan dengan benar (True Positive). Sementara itu, terdapat 48 data yang salah diklasifikasikan sebagai *Non-Cyberbullying* (False Negative). selanjutnya, dari total 129 data dengan label asli *Non-Cyberbullying*, sebanyak 102 data berhasil diklasifikasikan dengan benar (True Negative), sedangkan 27 data salah diklasifikasikan sebagai

Cyberbullying (False Positive). Dengan informasi ini, dapat dihitung metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *F1-score* untuk menilai kinerja model dalam mengklasifikasikan data. Berikut adalah perhitungan metrik evaluasi berdasarkan nilai-nilai pada *confusion matrix* di atas:

TP = 94

FN = 48

FP = 27

TN = 102

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} = \frac{94 + 102}{94 + 27 + 48 + 102} = \frac{196}{271} = 0.7236$$

$$Precision = \frac{TP}{TP + FP} = \frac{94}{94 + 27} = \frac{94}{121} = 0.7777$$

$$Recall = \frac{TP}{TP + FN} = \frac{94}{94 + 48} = \frac{94}{142} = 0.6611$$

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = 2 \times \frac{0.7777 \times 0.6611}{0.7777 + 0.6611} = 0.7156$$

Berdasarkan nilai-nilai tersebut, diperoleh nilai accuracy sebesar 0,7236, yang mengindikasikan bahwa 72,36% dari semua prediksi adalah benar. Nilai precision sebesar 0,7777, yang mengindikasikan bahwa 77,77% dari semua prediksi Cyberbullying adalah benar. Nilai recall sebesar 0,6611 menunjukkan bahwa model berhasil mengidentifikasi 66,11% dari seluruh data kelas Cyberbullying secara tepat. Nilai F1-score, yaitu rata-rata harmonis dari precision dan recall, berada pada angka 0,7156.

Kelas	Accuracy	Precision	Recall	F1-Score
1	0.7236	0.7777	0.6611	0.7156

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Implementasi

Penelitian ini dilakukan melalui serangkaian tahapan sebagaimana yang ditunjukkan pada Gambar 3.1. Hasil luaran utama berupa model *deep learning* untuk klasifikasi biner dengan arsitektur TextCNN yang dibangun dari scratch menggunakan dataset komentar *cyberbullying*, dimana model mampu mengklasifikan suatu komentar kedalam kelas *cyberbullying* ataupun *non-cyberbullying*. Uraian lebih rinci mengenai proses pengembangan serta hasil penelitian dipaparkan pada bagian berikut.

4.1.1 Analisis Dataset

Dataset yang digunakan dalam penelitian ini adalah dataset komentar *cyberbullying* berbahasa Indonesia yang diperoleh dari *repository* Github Titus Rangga Wicaksono yang digunakan pada paper 'Building Prediction Model for Detecting Cyberbullying using TikTok Comments' [6]. Dataset ini berisi komentar-komentar yang diambil dari platform sosial media TikTok dengan total 1.505 komentar yang telah dilabeli kedalam dua kelas *label encoding*, yaitu kelas *cyberbullying* (-1) dan kelas *non-cyberbullying* (1). Beberapa contoh data dari dataset dapat dilihat pada Tabel 4.1.

Tabel 4.1 Contoh Data pada Dataset

Label	Komentar
-1	Cowo paling alay lebay seantero selebgram WKWKW NGAKAK
1	aku penggemar mi burung dara enaknya nyambung terus
-1	KU KIRA MUKA TERNYATA AMPELA
1	buatlah cita cita kamu setinggi bintang di langit tetapi jangan lupa untuk membuat anak tangga perencanaannya

Dataset ini telah melalui proses pembersihan data (data cleaning) dan prapemrosesan (preprocessing) sehingga siap untuk digunakan dalam pelatihan model *deep learning*. Distribusi data pada masing-masing kelas dapat dilihat pada Tabel 4.2.

Tabel 4.2 Distribusi Data pada Masing-masing Kelas

Label	**Jumlah Data**	**Persentase**
Cyberbullying	675	44.85%
Non-cyberbullying	830	55.15%

Dari Tabel 4.2, dapat dilihat bahwa jumlah data pada masing-masing kelas tidak seimbang, dimana kelas *non-cyberbullying* memiliki jumlah data yang lebih banyak dibandingkan dengan kelas *cyberbullying*. Ketidakseimbangan yang terjadi tergolong ringan, sehingga tidak memerlukan penanganan khusus seperti *oversampling* atau *undersampling*. Namun, hal ini tetap menjadi perhatian dalam proses pelatihan model agar tidak terjadi *overfitting* pada kelas mayoritas.

4.1.2 Prapemrosesan Data

Pada bagian ini akan dipaparkan hasil dari proses prapemrosesan dataset. Proses prapemrosesan meliputi beberapa tahapan, seperti *case folding*, *text cleaning*, *augmentation*, *tokenizing* dan *stopword removal*. Hasil dari proses processing adalah dataset yang telah siap untuk digunakan dalam pelatihan model *deep learning*.

4.1.2.1 *Case Folding*

Tahap *case folding* adalah tahap awal dalam prapemrosesan teks dimana seluruh karakter dalam teks diubah menjadi huruf kecil. Hal ini bertujuan untuk mengurangi variasi kata yang sama yang ditulis dengan huruf kapital dan huruf kecil, sehingga memudahkan proses analisis selanjutnya. Implementasi kode untuk tahap *case folding* dapat dilihat pada Kode 4.1.

Kode 4.1 Case Folding

```
def preprocess(self, text):
    # Konversi ke huruf kecil
    text = text.lower()

```

Pada Kode 4.1, fungsi *preprocess* menerima input berupa teks komentar dan mengubah seluruh karakter menjadi huruf kecil menggunakan metode *lower()* dari tipe data string di Python. Hasil dari tahap *case folding* dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil Case Folding pada Beberapa Komentar

Label	Komentar Asli	Hasil Case Folding
-1	Cowo paling alay lebay seantero selebgram WKWKW NGAKAK	cowo paling alay lebay seantero selebgram wkwkw ngakak
-1	KU KIRA MUKA TERNYATA AMPELA	ku kira muka ternyata ampela

4.1.2.2 *Text Cleaning*

Tahap *text cleaning* adalah proses pembersihan teks dari karakter-karakter yang tidak diperlukan seperti emoji dan karakter khusus. Proses ini bertujuan untuk menyederhanakan teks sehingga hanya mengandung kata-kata yang relevan untuk analisis. Implementasi kode untuk tahap *text cleaning* dapat dilihat pada Kode 4.2.

Kode 4.2 Text Cleaning

```
# Hapus mention (@) dan hashtag (#)
text = re.sub(r'@\w+|#\w+', '', text)

# # Hapus emoji dan karakter non-ASCII
text = re.sub(r'[^\x00-\x7F]+', '', text)

```

Pada Kode 4.2, fungsi *preprocess* menggunakan modul *re* untuk menghapus mention dan hashtag dengan pola reguler, serta menghapus emoji dan karakter non-ASCII. Hasil dari tahap *text cleaning* dapat dilihat pada Tabel 4.4.

Tabel 4.4 Hasil Text Cleaning pada Beberapa Komentar

Label	Komentar Asli	Hasil Text Cleaning
-1	udah sok tau salah pula @IbnuWardani ????	udah sok tau salah pula ????
-1	yukk hesteg #hujatjeje	yukk hesteg

4.1.2.3 *Augmentation*

Tahap *augmentation* adalah proses penambahan variasi pada data teks untuk meningkatkan jumlah data pelatihan dan memperkaya representasi kata. Teknik yang digunakan adalah *An Easier Data Augmentation* (AEDA), *random swap*, dan *random delete*. Implementasi kode untuk tahap *augmentation* dapat dilihat pada Kode 4.3.

Kode 4.3 Augmentation

```
def aeda_augment(self, text):
    # Tanda baca augmentasi AEDA
    punctuations = [".", ";", "?", ":", "!", ","]
    words = text.split()
    if len(words) == 0:
        return text
    # Pilih posisi acak untuk sisipan
    position = random.randint(0, len(words) - 1)
    # Pilih tanda baca acak
    punct = random.choice(punctuations)
    # Sisipkan tanda baca ke dalam list kata
    words.insert(position, punct)
    return " ".join(words)

def random_typo(self, text):
    words = text.split()
    if len(words) < 1:
        return text
    # Pilih satu kata secara acak untuk dimodifikasi
    idx = random.randint(0, len(words) - 1)
    word = words[idx]
    if len(word) > 1:
        char_list = list(word)
        # Pilih posisi acak untuk swap
        i = random.randint(0, len(char_list) - 2)
        # swap 2 huruf berdekatan
        char_list[i], char_list[i+1] = char_list[i+1], char_list[i]
        words[idx] = ''.join(char_list)
    return ' '.join(words)

def random_delete(self, text):
    words = text.split()
    if len(words) <= 1:
        return text
    # Pilih satu kata secara acak untuk dihapus
    idx = random.randint(0, len(words) - 1)
    # Hapus kata tersebut
    del words[idx]
    return ' '.join(words)

```

```python
def augmentation_text(self, text, probability=0.5):
    # Hanya lakukan augmentasi dengan probabilitas tertentu
    if random.random() > probability:
        return text
    # Daftar augmentasi
    augmentations = [
        self.aeda_augment,
        self.random_typo,
        self.random_swap,
        self.random_delete
    ]
    # Pecah kalimat menjadi kumpulan kata
    words = text.split()
    # Menentukan jumlah augmentasi yang dilakukan
    n_insert = random.randint(1, max(1, len(words) // 3))
    # Perulangan augmentasi
    for i in range(n_insert):
        # Pilih satu augmentasi secara acak
        augmentation_func = random.choice(augmentations)
        # Terapkan augmentasi yang dipilih
        text = augmentation_func(text)

    return text

def preprocess(self, text):
    # ... Other Code

    # Augmentasi
    text = self.augmentation_text(text)

```

Pada Kode 4.3, terdapat beberapa fungsi untuk melakukan augmentasi teks, yaitu *aeda_augment* untuk menyisipkan tanda baca secara acak, *random_typo* untuk melakukan kesalahan ketik dengan menukar dua huruf berdekatan, dan *random_delete* untuk menghapus satu kata secara acak. Fungsi *augmentation_text* menggabungkan ketiga teknik augmentasi tersebut dengan probabilitas tertentu. Hasil dari tahap *augmentation* dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil Augmentation pada Beberapa Komentar

Label	Comment	Hasil	Kata/Frasa yang Diubah	
			Kata Asli	Kata Hasil
-1	ngeliat mukanya aja udah gedeg	ngeliat mukanya aja udah kesal	ngeliat mukanya aja udah gedeg	mukanya aja, udah gedeg!
1	gue gapaham dr dulu fashion style dia kyk gmna wkwkwk	gue gapaham dr duul fashion style dia kyk gmna wkwkwk	dulu	duul
-1	sebelum mangap di sikat dulu kale	sebelm mangap di sikat dulu kale	sebelum	sebelm

4.1.2.4 *Tokenizing*

Tahap *tokenizing* adalah proses pemecahan teks menjadi unit-unit yang lebih kecil yang disebut token. Token ini biasanya berupa kata atau frasa yang akan digunakan sebagai input untuk model *deep learning*. Implementasi kode untuk tahap *tokenizing* dapat dilihat pada Kode 4.4.

Kode 4.4 Tokenizing

```
def preprocess(self, text):
    # ... Other Code

    # Tokenizing
    tokens = word_tokenize(text)
    return tokens

```

Pada Kode 4.4, fungsi *preprocess* menggunakan modul *nltk* untuk memecah teks menjadi token menggunakan metode *word_tokenize()*. Hasil dari tahap *tokenizing* dapat dilihat pada Tabel 4.6.

Tabel 4.6 Hasil Tokenizing pada Beberapa Komentar

Label	Comment	Hasil
1	aku penggemar mi burung dara enaknya nyambung terus	'aku', 'penggemar', 'mi', 'burung', 'dara', 'enaknya', 'nyambung', 'terus'

4.1.2.5 *Stopword Removal*

S Tahap *stopword removal* adalah proses penghapusan kata-kata yang dianggap tidak memiliki makna penting dalam analisis teks, seperti kata hubung dan kata depan. Implementasi kode untuk tahap *stopword removal* dapat dilihat pada Kode 4.5.

Kode 4.5 Stopword Removal

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

def preprocess(self, text):
    # ... Other Code

    # Stopword Removal
    tokens = [word for word in tokens if word not in stop_words]
    return tokens

```

Pada Kode 4.5, fungsi *preprocess* menggunakan modul *nltk* untuk menghapus kata-kata yang terdapat dalam daftar *stopwords* bahasa Indonesia. Hasil dari tahap *stopword removal* dapat dilihat pada Tabel 4.7.

Tabel 4.7 Hasil Stopword Removal pada Beberapa Komentar

Label	Comment	Hasil	Kata yang Dihapus
1	buatlah cita cita kamu setinggi bintang di langit tetapi jangan lupa untuk membuat anak tangga perencanaannya	buatlah cita cita setinggi bintang langit jangan lupa membuat anak tangga perencanaannya	’kamu’, ’di’, ’tetapi’, ’untuk’

4.1.3 Pembagian *K-Fold* Pada Dataset

Kode 4.6 Pembagian *k-fold*

```
def create_folds(self):
    print(f"Membuat n-fold CV dengan random state
     {self.random_state}")
    skf = StratifiedKFold(n_splits=self.n_folds, shuffle=True,
     random_state=self.random_state)
    fold_indices = {}
    for fold, (train_idx, val_idx) in enumerate(skf.split(self.df,
     self.df['sentiment'])):
        fold_indices[f"fold_{fold}"] = {
            'train_indices': train_idx.tolist(),
            'val_indices': val_idx.tolist()
        }
```

```
10
11      # Simpan fold ke file
12      with open(self.folds_file, 'w') as f:
13          json.dump({
14              'fold_indices': fold_indices,
15              'n_samples': len(self.df),
16              'n_folds': self.n_folds,
17              'random_state': self.random_state
18          }, f)
19          self.fold_indices = fold_indices
20          print(f'Created {self.n_folds}-fold indices and saved to
        {self.folds_file}')
21
```

Fungsi generate kfold splits pada Kode 4.6 membangun skema validasi silang berbasis kelompok untuk mencegah kebocoran data antar kelompok. Metode ini menginisialisasi proses pembagian data menggunakan teknik *Stratified K-Fold Cross Validation* dengan *random state* yang telah ditentukan untuk menjamin reproduksibilitas eksperimen. Proses pembagian dilakukan pada tingkat baris *dataframe* dengan merujuk pada kolom *sentiment* sebagai dasar stratifikasi, sehingga setiap *fold* memiliki distribusi kelas yang seimbang dan representatif terhadap keseluruhan *dataset*. Hal ini memastikan bahwa evaluasi model memberikan hasil yang objektif karena proporsi kategori sentimen pada setiap *subset training* dan *validation* tetap konsisten. Hasil akhir dari fungsi ini adalah sebuah struktur data berbentuk kamus yang menyimpan daftar indeks *train* dan *val* untuk setiap *fold*. Seluruh informasi pembagian ini, beserta metadata pendukung seperti jumlah sampel dan konfigurasi *random state*, disimpan ke dalam sebuah berkas *.json* eksternal. Prosedur penyimpanan ini memungkinkan sistem untuk memuat kembali (*load*) pembagian data yang identik pada tahap pelatihan model berikutnya, sehingga memastikan konsistensi data yang digunakan di seluruh tahapan penelitian.

Tabel 4.8 Hasil pembagian *K-Fold* pada dataset

Iterasi	*K-Fold* per Folder					Data Latih	Data Evaluasi
	1	2	3	4	5		
1	301	301	301	301	301	1204	301
2	301	301	301	301	301	1204	301
3	301	301	301	301	301	1204	301
4	301	301	301	301	301	1204	301
5	301	301	301	301	301	1204	301

Pada Tabel 4.8 ditunjukkan skema *k-fold cross-validation* sejumlah 5 *fold* untuk mengukur kinerja arsitetur *deep learning* secara lebih akurat dan andal. Pada tiap iterasi (1–5), satu *fold* digunakan untuk validasi (sel berwarna kuning), sementara empat kelompok lainnya dipakai untuk pelatihan. Total terdapat 1505 baris data pada dataset, sehingga dibagi menjadi 5 kelompok: empat kelompok dengan masing-masing berisi 301 data dengan total 1204 data untuk pelatihan dan satu kelompok dengan 301 data untuk validasi. Jumlah data pada setiap kelompok data tetap, ini terjadi karena tiap kelompok data menyimpan jumlah data yang sama. Dengan rancangan ini, setiap kelompok data tepat sekali menjadi validasi dan empat kali masuk pelatihan, memastikan tidak ada data yang sama muncul di train dan validasi secara bersamaan (*no leakage*).

4.1.4 Perancangan Model Klasifikasi

Pada subbab ini akan dijelaskan secara rinci mengenai proses perancangan model klasifikasi yang digunakan dalam penelitian. Arsitektur yang digunakan adalah arsitektur TextCNN sebagai model utama, kemudian dilanjutkan dengan pengembangan arsitektur melalui integrasi mekanisme Squeeze-and-Excitation (SE) dan Depthwise Separable Convolution pada model SEDepthwise TextCNN. Setiap bagian akan membahas struktur, komponen utama, serta alur inferensi model secara sistematis untuk memberikan gambaran menyeluruh terkait strategi ekstraksi fitur dan klasifikasi yang diimplementasikan pada tugas deteksi komentar *cyberbullying*.

4.1.4.1 TextCNN

Kode 4.7 Kode model TextCNN

```
class TextCNN(nn.Module):
    def __init__(
        self,
        vocab_size,
        in_channels=300,
        num_classes=2,
        conv_filters=100,
        kernel_sizes=[3, 4, 5],
        dropout_rate=0.5
    ):
        super(TextCNN, self).__init__()

        self.embedding = nn.Embedding(vocab_size, in_channels)

        self.convs = nn.ModuleList([
            nn.Conv1d(in_channels, conv_filters, kernel_size=k)
            for k in kernel_sizes
        ])

        self.dropout = nn.Dropout(dropout_rate)
        self.fc = nn.Linear(conv_filters * len(kernel_sizes),
     num_classes)

    def forward(self, x):
        # x: (batch, seq_len)
        x = self.embedding(x)           # (batch, seq_len, embed_dim)
        x = x.permute(0, 2, 1)          # (batch, embed_dim, seq_len)

        conv_outs = []
        for conv in self.convs:
            h = F.relu(conv(x))         # (batch, conv_filters, L')
            h = torch.max(h, dim=2)[0]
            conv_outs.append(h)

        x = torch.cat(conv_outs, dim=1)
        x = self.dropout(x)
        return self.fc(x)

```

Arsitektur model TextCNN diinisialisasi pada Gambar **??** dengan membangun komponen ekstraksi fitur berbasis teks secara paralel. Lapisan *embedding* dipersiapkan untuk memetakan indeks kata ke dalam ruang vektor kontinu berdimensi $in\_channels$. Inti dari model ini terletak pada penggunaan beberapa lapisan konvolusi satu dimensi (Conv1d) dengan ukuran kernel yang bervariasi (3, 4, dan 5) yang disusun dalam *ModuleList*. Pendekatan *multi-kernel* ini bertujuan untuk menangkap informasi *n-gram* dengan cakupan berbeda dalam satu langkah komputasi. Lapisan *dropout* disiapkan sebagai mekanisme regularisasi, sementara bagian klasifikasi menggunakan satu lapisan *linear* (*fc*) yang berfungsi memetakan akumulasi fitur dari seluruh filter

konvolusi ke ruang kelas target sesuai parameter *num_classes*.

Fungsi *forward* pada Gambar 4.y merealisasikan alur inferensi dari tingkat kata hingga prediksi klasifikasi. Data *input* terlebih dahulu ditransformasikan melalui lapisan *embedding*, yang kemudian diikuti oleh operasi permutasi dimensi (*permute*) untuk menyelaraskan struktur tensor dengan kebutuhan input lapisan konvolusi 1D, yaitu menempatkan dimensi kanal sebelum dimensi sekuens. Setiap blok konvolusi mengekstraksi fitur lokal yang kemudian diproses melalui fungsi aktivasi *ReLU* untuk memperkenalkan sifat non-linearitas. Operasi *Global Max Pooling* (*torch.max*) diterapkan pada setiap *output* konvolusi untuk mereduksi dimensi temporal, sehingga hanya fitur paling dominan dari setiap filter yang diambil sebagai representasi tingkat kalimat. Fitur-fitur dari berbagai ukuran kernel tersebut kemudian digabungkan (*concatenate*) menjadi satu vektor representasi global. Sebelum mencapai lapisan luaran, mekanisme *dropout* diterapkan untuk mencegah *overfitting*, diikuti oleh proyeksi akhir melalui lapisan *fully connected* guna menghasilkan nilai logit klasifikasi.

4.1.4.2 *SEDepthwise* TextCNN

Kode 4.8 Kode model SEDepthwise TextCNN

```
class SEDepthwiseTextCNN(nn.Module):
    def __init__(self, vocab_size, in_channels=300, num_classes=2,
     conv_filters=100, kernel_sizes=[3,4,5], dropout_rate=0.5):
        super(SEDepthwiseTextCNN, self).__init__()
        self.embedding = nn.Embedding(vocab_size, in_channels,
     padding_idx=0)

        # Depthwise + Pointwise convolution blocks
        self.convs = nn.ModuleList([
            nn.Sequential(
                nn.Conv1d(in_channels, in_channels, kernel_size=ks,
     groups=in_channels, padding=0),  # depthwise
                nn.Conv1d(in_channels, conv_filters, kernel_size=1),
      # pointwise
                nn.ReLU()
            )
            for ks in kernel_sizes
        ])

        self.dropout = nn.Dropout(dropout_rate)
        self.se_block = SEBlock(conv_filters * len(kernel_sizes))
        self.fc = nn.Linear(conv_filters * len(kernel_sizes),
     num_classes)

```

```
    # Helper block untuk konvolusi + aktivasi + pooling
    def conv_block(self, x, depthwise, pointwise):
        x = depthwise(x)
        x = F.relu(pointwise(x))
        x = F.max_pool1d(x, kernel_size=x.size(2)).squeeze(2)
        return x

    def forward(self, x):
        x = self.embedding(x)   # (batch, seq_len, in_channels)
        x = x.permute(0, 2, 1)  # (batch, in_channels, seq_len)

        conv_outputs = []
        for conv in self.convs:
            y = conv(x)
            y = F.max_pool1d(y, kernel_size=y.size(2)).squeeze(2)
            conv_outputs.append(y)

        x_cat = torch.cat((conv_outputs), dim=1)    # (batch,
 conv_filters * 3)
        x_cat = self.se_block(x_cat)
        x_cat = self.dropout(x_cat)

        return self.fc(x_cat)

```

Model *SE-Depthwise TextCNN* pada Kode 4.8 dirancang sebagai pengembangan dari struktur *convolutional neural network* konvensional dengan mengintegrasikan efisiensi parameter dan mekanisme atensi kanal. Inisialisasi model dimulai dengan lapisan *embedding* yang dilengkapi *padding_idx=0* untuk menangani sekuens teks dengan panjang bervariasi. Berbeda dengan konvolusi standar, arsitektur ini menerapkan *Depthwise Separable Convolution* melalui pasangan *depthwise_conv* dan *pointwise_conv*. Penggunaan parameter *groups=in_channels* pada *depthwise convolution* memungkinkan ekstraksi fitur spasial pada tiap kanal secara independen, yang kemudian dikombinasikan secara linear oleh *pointwise convolution* (1×1). Strategi ini secara signifikan mereduksi beban komputasi tanpa mengorbankan kualitas representasi fitur. Selain itu, penggunaan *dilation* pada blok kedua bertujuan untuk memperluas *receptive field* tanpa menambah jumlah parameter.

Kode 4.9 Kode se block

```
            class SEBlock(nn.Module):
                def __init__(self, channels, reduction=16):
                    super(SEBlock, self).__init__()
                    self.fc1 = nn.Linear(channels, channels //
    reduction)
                    self.fc2 = nn.Linear(channels // reduction,
    channels)

                def forward(self, x):
```

```
8                       w = F.relu(self.fc1(x))
9                       w = torch.sigmoid(self.fc2(w))
10
11                      return x * w
12
```

Mekanisme *Squeeze-and-Excitation* (SE) *Block* yang ada pada Kode 4.9 diintegrasikan setelah tahap penggabungan fitur (*concatenation*) untuk memberikan bobot adaptif pada setiap kanal fitur. Secara umum, SE *Block* bekerja dengan mengekstraksi informasi global dari tiap kanal melalui operasi *squeeze*, yang kemudian diproses melalui struktur *bottleneck* menggunakan dua lapisan *fully connected* (*fc1* dan *fc2*). Lapisan pertama mereduksi dimensi kanal dengan rasio tertentu (*reduction ratio*) untuk efisiensi, sementara lapisan kedua mengembalikan dimensi kanal ke ukuran semula. Hasil akhirnya adalah sekumpulan bobot (*excitation*) yang merepresentasikan tingkat kepentingan relatif dari setiap fitur, yang kemudian digunakan untuk mengalibrasi ulang (*re-weighting*) fitur masukan melalui perkalian elemen-per-elemen.

Fungsi *forward* merealisasikan alur data yang dimulai dari transformasi *embedding* dan permutasi dimensi tensor. Proses ekstraksi fitur dilakukan secara paralel melalui dua blok konvolusi utama yang masing-masing menerapkan *depthwise-pointwise convolution*, aktivasi *ReLU*, dan *Global Max Pooling*. Hasil dari kedua blok tersebut digabungkan menggunakan *torch.cat* dan dilewatkan menuju *SE Block*. Di dalam blok ini, operasi *torch.mean* digunakan untuk merangkum statistik global kanal, diikuti oleh aktivasi *Sigmoid* untuk menghasilkan koefisien atensi antara 0 dan 1. Fitur yang telah dikalibrasi kemudian melewati lapisan *dropout* untuk regularisasi dan diakhiri dengan lapisan *linear* untuk memproyeksikan representasi final ke dalam ruang kelas prediksi.

4.2 Evaluasi Hasil

Pada subbab ini akan dijelaskan secara rinci bagaimana hasil pelatihan model menggunakan konfigurasi hyperparameter default dan konfigurasi

hyperparameter studi ablasi.

4.2.1 Evaluasi Model dengan *Hyperparameter Default*

Pada subbab ini akan dipaparkan hasil evaluasi model klasifikasi komentar *cyberbullying* menggunakan konfigurasi *hyperparameter default* pada berbagai varian arsitektur, yaitu TextCNN ringan, sedang, berat, serta SEDepthwise TextCNN.

4.2.1.1 Evaluasi Model TextCNN Ringan

Tabel 4.9 Konfigurasi *hyperparameter* default pada model TextCNN Ringan

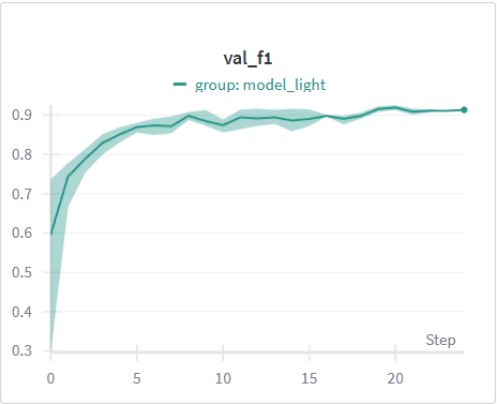
No	***Hyperparameter***	**Nilai**
1	*optimizers*	AdamW
2	*batch_sizes*	50
3	*learning_rates*	5e-3
4	*embedding_dimension*	100
5	*kernel_sizes*	[3, 4]
6	*convolution_filters*	50
7	*dropout_rates*	0.5

Konfigurasi *hyperparameter* pada Tabel 4.9 menunjukkan pengaturan dasar model TextCNN Ringan yang digunakan pada eksperimen ini. Model menggunakan optimizer AdamW untuk pembaruan bobot yang stabil, batch size sebesar 50, dan learning rate 5×10^{-3} untuk efisiensi konvergensi. Dimensi embedding ditetapkan 100, sementara kernel size yang digunakan adalah $[3, 4]$ untuk mengekstraksi fitur dari berbagai n-gram. Setiap kernel memiliki 50 convolution filters, dan regularisasi dilakukan dengan dropout rate 0,5 untuk mengurangi risiko overfitting. Seluruh konfigurasi ini dipilih untuk menjaga keseimbangan antara kompleksitas model, kemampuan ekstraksi fitur, dan stabilitas pelatihan.

Tabel 4.10 Hasil evaluasi model TextCNN ringan dengan *hyperparameter default*

Fold	**Hasil *Confussion Metrix***				**Hasil *Evaluation Metrix***			
	TP	**TN**	**FP**	**FN**	**Acc**	**Pre**	**Rec**	**F1**
1	119	111	14	26	0,851	0,894	0,820	0,856
2	137	108	17	8	**0,907**	**0,889**	**0,944**	**0,916**
3	134	105	20	11	0,885	0,870	0,924	0,896
4	137	108	17	8	**0,907**	**0,889**	**0,944**	**0,916**
5	129	105	20	16	0,866	0,865	0,889	0,877
Rata-rata					0,883	0,881	0,904	0,892

Tabel 4.10 menampilkan ringkasan *confusion matrix* dan metrik evaluasi utama untuk setiap *fold* pada model *TextCNN* ringan dengan *hyperparameter default*. Kinerja model menunjukkan hasil yang sangat kompetitif dengan akurasi berkisar antara 0,851 hingga 0,907, dengan rerata 0,883. Rata-rata *recall* (0,904) tercatat sedikit lebih tinggi dibandingkan dengan rata-rata *precision* (0,881), yang menghasilkan rata-rata *f1-score* sebesar 0,892. Pola ini mengindikasikan bahwa model memiliki kemampuan yang sangat baik dalam menangkap kelas positif dengan jumlah *false negatives* (*FN*) yang relatif rendah. Sebagai contoh, pada *Fold* 2 dan *Fold* 4, model mencapai performa puncak dengan akurasi 0,907 dan *f1-score* 0,916, didukung oleh nilai *FN* yang sangat minim (8). Sebaliknya, *Fold* 1 memberikan hasil terendah dengan akurasi 0,851 karena jumlah *FN* (26) dan *FP* (14) yang lebih tinggi dibandingkan *fold* lainnya. Variasi performa antar *fold* ini dipengaruhi oleh karakteristik distribusi data pada mekanisme *k-fold cross validation*. Secara keseluruhan, model *TextCNN* ringan terbukti stabil dan efektif dalam melakukan klasifikasi dengan keseimbangan metrik yang tinggi.



Gambar 4.1 F1-Score Model TextCNN Ringan dengan *Hyperparameter Default*
Sumber: dokumen pribadi

Kurva pada Gambar 4.1 menunjukkan rata-rata *f1-score* validasi selama proses pelatihan untuk kelompok *model_light*. Terlihat adanya pola peningkatan yang sangat signifikan di awal tahapan pelatihan, di mana nilai *f1-score* melonjak tajam dari kisaran $\approx 0,60$ pada *step* 0 menjadi $\approx 0,85$ pada *step* 5. Setelah fase kenaikan awal yang agresif tersebut, kurva mulai *converge* dan menunjukkan tren penguatan yang lebih stabil hingga mencapai puncaknya di sekitar $\approx 0,91$ pada akhir *step*. Area bayangan di sekitar garis utama merepresentasikan variansi antar *fold* yang cenderung menyempit seiring bertambahnya *step*, menandakan konsistensi model yang semakin membaik selama proses pelatihan. Tidak ditemukan indikasi *overfitting* yang berarti karena performa pada data validasi tetap terjaga pada level tinggi tanpa adanya penurunan sistematis hingga akhir pelatihan. Visualisasi kurva ini sangat selaras dengan hasil pada Tabel 4.8, di mana rata-rata *f1-score* akhir berada pada angka 0,892, yang menegaskan stabilitas dan efektivitas pembelajaran model pada pengaturan *hyperparameter default*.

4.2.1.2 Evaluasi Model TextCNN Sedang

Tabel 4.11 Konfigurasi *hyperparameter* default pada model TextCNN Sedang

No	***Hyperparameter***	**Nilai**
1	*optimizers*	AdamW
2	*batch_sizes*	50
3	*learning_rates*	5e-3
4	*embedding_dimension*	300
5	*kernel_sizes*	[3, 4, 5]
6	*convolution_filters*	100
7	*dropout_rates*	0.5

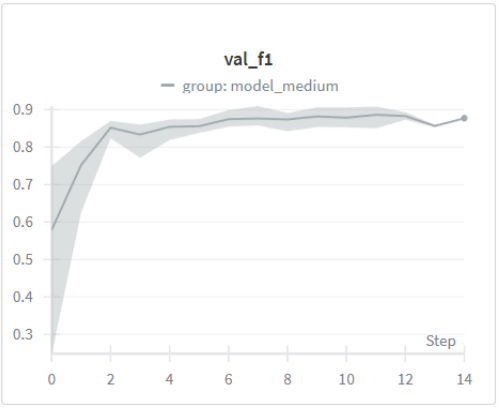
Konfigurasi *hyperparameter* pada Tabel 4.11 menunjukkan pengaturan dasar model TextCNN Sedang yang digunakan pada eksperimen ini. Model menggunakan optimizer AdamW untuk pembaruan bobot yang stabil, batch size sebesar 50, dan learning rate 5×10^{-3} untuk efisiensi konvergensi. Dimensi embedding ditetapkan 300, sementara kernel size yang digunakan adalah $[3, 4, 5]$ untuk mengekstraksi fitur dari berbagai n-gram. Setiap kernel memiliki 100 convolution filters, dan regularisasi dilakukan dengan dropout rate 0,5 untuk mengurangi risiko overfitting. Seluruh konfigurasi ini dipilih untuk menjaga keseimbangan antara kompleksitas model, kemampuan ekstraksi fitur, dan stabilitas pelatihan.

Tabel 4.12 Hasil evaluasi model TextCNN sedang dengan *hyperparameter default*

Fold	**Hasil *Confussion Metrix***				**Hasil *Evaluation Metrix***			
	TP	**TN**	**FP**	**FN**	**Acc**	**Pre**	**Rec**	**F1**
1	128	105	20	17	**0,862**	0,864	**0,882**	**0,873**
2	122	107	18	23	0,848	0,871	0,841	0,856
3	125	106	19	20	0,855	0,868	0,862	0,865
4	117	115	10	28	0,859	**0,921**	0,806	0,860
5	123	102	23	22	0,833	0,842	0,848	0,845
Rata-rata					0,851	0,873	0,847	0,859

Tabel 4.11 menampilkan ringkasan *confusion matrix* dan metrik evaluasi

utama untuk setiap *fold* pada model *TextCNN* sedang dengan *hyperparameter default*. Kinerja model menunjukkan stabilitas yang baik dengan akurasi berkisar antara 0,833 hingga 0,862, menghasilkan rata-rata akurasi sebesar 0,851. Rata-rata *precision* tercatat sebesar 0,873, sedikit lebih tinggi dibandingkan rata-rata *recall* yang berada pada angka 0,847, sehingga menghasilkan rata-rata *f1-score* sebesar 0,859. Hasil ini mengindikasikan bahwa model memiliki kecenderungan untuk meminimalkan *false positives*, yang terlihat jelas pada *Fold* 4 dengan *precision* tertinggi mencapai 0,921. Meskipun demikian, *Fold* 1 memberikan keseimbangan performa terbaik dengan *f1-score* tertinggi sebesar 0,873, didukung oleh nilai *TP* sebanyak 128 dan *TN* sebanyak 105. Variasi antar *fold* yang relatif kecil menunjukkan bahwa model *TextCNN* sedang cukup tangguh terhadap perbedaan distribusi data dalam *k-fold cross validation*. Secara keseluruhan, model ini menunjukkan performa yang konsisten dan andal pada kategori ukuran sedang.



Gambar 4.2 Grafik Rata-rata F1-Score Model TextCNN Sedang dengan *Hyperparameter Default*

Kurva pada Gambar 4.2 menunjukkan rata-rata *f1-score* validasi selama proses pelatihan untuk kelompok *model_medium*. Grafik menunjukkan proses pembelajaran yang sangat cepat di awal, di mana *f1-score* meningkat tajam dari

nilai awal $\approx 0,58$ hingga mencapai $\approx 0,85$ hanya dalam 2 *step* pertama. Setelah lonjakan awal tersebut, kurva bergerak lebih stabil dan mulai *converge* pada kisaran nilai $0,86 - 0,88$ hingga akhir pelatihan di *step* 14. Area bayangan di sekitar kurva utama, yang merepresentasikan variansi antar *fold*, terlihat cukup lebar pada tahap awal namun berangsur menyempit seiring dengan kemajuan *step*, menandakan peningkatan konsistensi model. Tidak ditemukan gejala *overfitting* yang signifikan karena tren performa pada data validasi tetap terjaga stabil tanpa adanya penurunan yang tajam menjelang akhir *step*. Hasil visualisasi ini memperkuat temuan pada Tabel 4.10, di mana rata-rata *f1-score* akhir sebesar 0,859 mencerminkan keberhasilan model dalam mencapai titik optimal pembelajaran pada konfigurasi *hyperparameter* yang ditentukan.

4.2.1.3 Evaluasi Model TextCNN Berat

Tabel 4.13 Konfigurasi *hyperparameter* default pada model TextCNN Berat

No	***Hyperparameter***	**Nilai**
1	*optimizers*	AdamW
2	*batch_sizes*	50
3	*learning_rates*	5e-3
4	*embedding_dimension*	600
5	*kernel_sizes*	[3, 4, 5, 6]
6	*convolution_filters*	200
7	*dropout_rates*	0.5

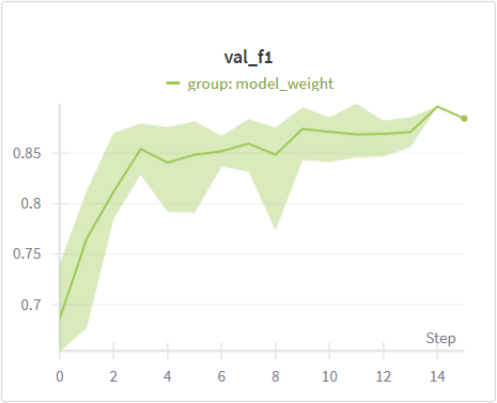
Konfigurasi *hyperparameter* pada Tabel 4.13 menunjukkan pengaturan dasar model TextCNN Berat yang digunakan pada eksperimen ini. Model menggunakan optimizer AdamW untuk pembaruan bobot yang stabil, batch size sebesar 50, dan learning rate 5×10^{-3} untuk efisiensi konvergensi. Dimensi embedding ditetapkan 600, sementara kernel size yang digunakan adalah $[3, 4, 5, 6]$ untuk mengekstraksi fitur dari berbagai n-gram. Setiap kernel memiliki 200 convolution filters, dan regularisasi dilakukan dengan dropout rate 0,5 untuk mengurangi risiko overfitting. Seluruh konfigurasi ini dipilih untuk

menjaga keseimbangan antara kompleksitas model, kemampuan ekstraksi fitur, dan stabilitas pelatihan.

Tabel 4.14 Hasil evaluasi model TextCNN berat dengan *hyperparameter default*

Fold	**Hasil *Confussion Metrix***				**Hasil *Evaluation Metrix***			
	TP	**TN**	**FP**	**FN**	**Acc**	**Pre**	**Rec**	**F1**
1	121	111	14	24	0,859	0,896	0,834	0,864
2	118	96	29	27	0,792	0,802	0,813	0,808
3	118	115	10	27	**0,862**	**0,921**	0,813	0,864
4	124	109	16	21	**0,862**	0,885	**0,855**	**0,870**
5	123	84	41	22	0,766	0,750	0,848	0,796
Rata-rata					0,828	0,850	0,832	0,840

Tabel 4.14 menampilkan ringkasan *confusion matrix* dan metrik evaluasi utama untuk setiap *fold* pada model *TextCNN* berat dengan *hyperparameter default*. Kinerja model menunjukkan variansi yang cukup terlihat dengan akurasi berkisar antara 0,766 hingga 0,862, serta rerata akurasi sebesar 0,828. Rata-rata *precision* tercatat sebesar 0,850, sedikit lebih tinggi dibandingkan rata-rata *recall* sebesar 0,832, yang menghasilkan rata-rata *f1-score* pada angka 0,840. Pola ini mengindikasikan bahwa model berat cenderung memiliki tingkat presisi yang baik namun mengalami fluktuasi dalam mengenali seluruh sampel positif di beberapa *fold*. Sebagai contoh, pada *Fold* 4, model mencapai performa terbaik dengan akurasi 0,862 dan *f1-score* 0,870. Namun, pada *Fold* 5, akurasi menurun menjadi 0,766 akibat tingginya jumlah *false positives* (41) yang berdampak pada rendahnya *precision* (0,750). Secara keseluruhan, meskipun memiliki arsitektur yang lebih kompleks, model *TextCNN* berat menunjukkan performa yang sedikit di bawah varian model ringan dan sedang dalam kondisi *default*.



Gambar 4.3 Grafik Rata-rata F1-Score Model TextCNN Berat dengan *Hyperparameter Default*

Sumber: dokumen pribadi

Kurva pada Gambar 4.3 menunjukkan rata-rata *f1-score* validasi selama proses pelatihan untuk kelompok *model_weight*. Terlihat pola peningkatan awal yang cukup tajam dari kisaran $\approx 0,68$ menuju $\approx 0,85$ pada *step* 3, namun proses menuju konvergensi menunjukkan volatilitas yang lebih tinggi dibandingkan varian model lainnya. Hal ini ditandai dengan adanya fluktuasi yang signifikan, terutama penurunan pada *step* 8 sebelum akhirnya kembali meningkat dan mencapai puncak di kisaran $\approx 0,89$ pada *step* 14. Area bayangan di sekitar kurva utama terlihat cukup lebar di sepanjang proses pelatihan, yang mencerminkan variansi antar *fold* yang besar akibat sensitivitas model terhadap distribusi data. Ketidakstabilan di pertengahan *step* menunjukkan bahwa model berat mungkin memerlukan regularisasi yang lebih kuat atau penyesuaian *learning rate* untuk mencapai stabilitas yang lebih baik. Secara keseluruhan, grafik ini selaras dengan temuan pada Tabel 4.12 yang menunjukkan rerata *f1-score* akhir di angka 0,840 dengan tingkat variansi performa yang cukup terasa.

4.2.1.4 Evaluasi Model SEDepthwise TextCNN

Tabel 4.15 Konfigurasi *hyperparameter* default pada model SEDepthwise TextCNN

No	***Hyperparameter***	**Nilai**
1	*optimizers*	AdamW
2	*batch_sizes*	50
3	*learning_rates*	5e-3
4	*embedding_dimension*	300
5	*kernel_sizes*	[3, 4, 5]
6	*convolution_filters*	100
7	*dropout_rates*	0.5

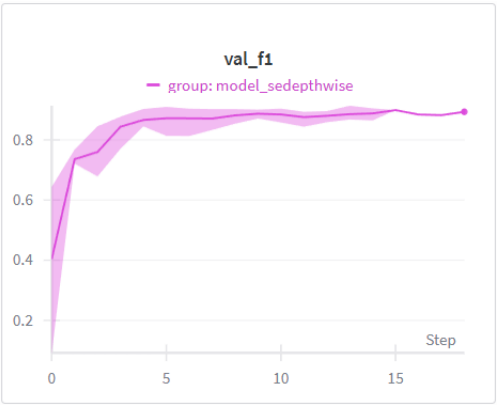
Konfigurasi *hyperparameter* pada Tabel 4.15 menunjukkan pengaturan dasar model SEDepthwise TextCNN yang digunakan pada eksperimen ini. Model menggunakan optimizer AdamW untuk pembaruan bobot yang stabil, batch size sebesar 50, dan learning rate 5×10^{-3} untuk efisiensi konvergensi. Dimensi embedding ditetapkan 100, sementara kernel size yang digunakan adalah $[3, 4, 5]$ untuk mengekstraksi fitur dari berbagai n-gram. Setiap kernel memiliki 64 convolution filters, dan regularisasi dilakukan dengan dropout rate 0,5 untuk mengurangi risiko overfitting. Seluruh konfigurasi ini dipilih untuk menjaga keseimbangan antara kompleksitas model, kemampuan ekstraksi fitur, dan stabilitas pelatihan.

Tabel 4.16 Hasil evaluasi model SEDepthwise TextCNN dengan *hyperparameter default*

Fold	**Hasil *Confussion Metrix***				**Hasil *Evaluation Metrix***			
	TP	**TN**	**FP**	**FN**	**Acc**	**Pre**	**Rec**	**F1**
1	132	101	24	13	0,862	0,846	0,910	0,877
2	131	113	12	14	**0,903**	**0,916**	0,903	**0,909**
3	128	104	21	17	0,859	0,859	0,881	0,870
4	138	102	23	7	0,888	0,857	**0,951**	0,901
5	120	106	19	25	0,837	0,863	0,827	0,845
Rata-rata					0,869	0,868	0,894	0,880

Tabel 4.16 menampilkan hasil evaluasi model *SEDepthwise TextCNN*

yang menunjukkan performa sangat baik dengan rerata akurasi 0,869 dan *f1-score* sebesar 0,880. Model ini terbukti sangat efektif dalam menangkap sampel positif, terlihat dari nilai rerata *recall* yang tinggi (0,894), dengan capaian tertinggi pada *Fold* 4 sebesar 0,951. *Fold* 2 memberikan hasil paling optimal secara keseluruhan dengan akurasi 0,903 dan *f1-score* 0,909, yang membuktikan bahwa integrasi mekanisme *Squeeze-and-Excitation* dan *depthwise convolution* memberikan kontribusi positif pada akurasi klasifikasi.



Gambar 4.4 Grafik rata-rata F1-score model SEDepthwise TextCNN dengan *hyperparameter default*

Sumber: dokumen pribadi

Kurva pada Gambar 4.4 untuk *model_sedepthwise* menunjukkan tren kenaikan yang konsisten dari nilai awal $\approx 0,4$ menuju $\approx 0,85$ pada *step* 4. Setelah fase tersebut, model mencapai titik konvergensi yang sangat halus dengan fluktuasi minimal dan variansi antar *fold* yang sangat rendah hingga akhir *step* 18. Karakteristik kurva yang stabil dan presisi ini menegaskan bahwa arsitektur *SEDepthwise* memiliki reliabilitas yang tinggi dan mampu melakukan generalisasi pola data secara lebih baik dibandingkan arsitektur *TextCNN* standar.

4.2.2 Perbandingan Evaluasi Model

Perbandingan evaluasi antar *fold* pada keempat model menunjukkan pola konsistensi yang menarik dalam konteks *cross-validation*, di mana model *TextCNN* Ringan dan *SEDepthwise* menunjukkan stabilitas performa yang lebih tinggi dibandingkan varian lainnya. Variasi performa yang terjadi antar *fold* dapat dijelaskan melalui perbedaan distribusi karakteristik fitur linguistik dalam *subset* data, di mana penurunan akurasi pada *fold* tertentu menandakan adanya sampel dengan tingkat ambiguitas sentimen yang lebih tinggi atau ketidakseimbangan kelas pada set validasi tersebut. Sebaliknya, peningkatan metrik pada *fold* spesifik mengindikasikan bahwa *validation set* tersebut memiliki karakteristik yang sangat representatif terhadap *training set*, sehingga model dapat melakukan generalisasi fitur secara lebih efektif untuk menghasilkan prediksi yang akurat. Secara khusus, *Fold* 2 dan *Fold* 4 menunjukkan performa puncak pada model-model dengan akurasi tinggi, di mana *TextCNN* Ringan mencapai akurasi 0,907 dan *SEDepthwise* mencapai akurasi 0,903 pada *Fold* 2. Fenomena ini dapat dijelaskan melalui pembagian data menggunakan *stratified k-fold* yang menghasilkan konfigurasi data dengan keseimbangan optimal antara representasi kelas sentimen dan variasi pola linguistik pada iterasi tersebut. *Training set* pada iterasi ini mengandung sampel yang lebih kaya akan informasi sehingga model dapat membedakan fitur-fitur penting dengan tingkat kepercayaan (*confidence*) yang lebih tinggi, sementara *validation set* memberikan tantangan klasifikasi yang sesuai dengan kapasitas fitur yang telah dipelajari oleh model selama proses pelatihan. Berdasarkan urutan prioritas *f1-score*, *recall*, dan *precision*, model *TextCNN* Ringan menempati urutan pertama sebagai model paling optimal dengan rerata *f1-score* 0,892, *recall* 0,904, dan *precision* 0,881, diikuti berturut-turut oleh model *SEDepthwise* (*f1-score* 0,880), *TextCNN* Sedang (0,859), dan *TextCNN* Berat (0,840). Fakta bahwa model Ringan mengungguli model Berat

menunjukkan adanya kecenderungan *over-parameterization* pada arsitektur yang lebih kompleks, di mana jumlah parameter yang besar pada model Berat justru menurunkan efektivitas generalisasi pada dataset ulasan yang digunakan. Model *SEDepthwise* memberikan performa yang sangat kompetitif dengan stabilitas pelatihan yang paling baik berkat mekanisme *Squeeze-and-Excitation*, namun model Ringan tetap menjadi pilihan terbaik karena efisiensinya dalam mencapai nilai *recall* dan *f1-score* tertinggi secara konsisten. Stabilitas performa yang terlihat pada seluruh varian menunjukkan bahwa penggunaan 5 *fold cross-validation* sudah mencukupi untuk menghasilkan evaluasi yang andal (*reliable*) tanpa perlu memperluas hingga 10 *fold*. Rentang rerata akurasi yang berada pada kisaran 0,828 hingga 0,883 menunjukkan bahwa model telah mencapai titik performa yang stabil pada konfigurasi *hyperparameter default* yang diberikan. Konsistensi tren performa antar *fold* yang relatif minim variasinya membuktikan bahwa hasil perbandingan ini sudah cukup kuat untuk menarik kesimpulan mengenai efektivitas masing-masing arsitektur, sehingga penambahan jumlah *fold* hanya akan meningkatkan beban komputasi tanpa memberikan peningkatan pemahaman yang signifikan terhadap kualitas model.

4.2.3 Evaluasi Studi Ablasi

Studi ablasi pada penelitian ini bertujuan untuk menganalisis kontribusi masing-masing *hyperparameter* terhadap kinerja model TextCNN Ringan dalam tugas analisis sentimen *cyberbullying*. *Hyperparameter* yang dievaluasi meliputi *optimizer*, *batch size*, *learning rate*, *embedding dimension*, *kernel size*, *convolution filters*, dan *dropout rate*. Pemilihan *hyperparameter* tersebut didasarkan pada pengaruh langsungnya terhadap proses ekstraksi fitur teks, stabilitas pelatihan, kemampuan generalisasi model, serta efisiensi komputasi pada arsitektur *TextCNN ringan*.

Seluruh eksperimen dilakukan untuk semua fold sejumlah 5 fold data, yang kemudian diambil nilai rerata dari *accuracy, precission, recall,* dan *f1-*

score untuk generalisasi model terhadap semua fold. Untuk menjaga konsistensi dan keadilan evaluasi, pada setiap percobaan hanya satu *hyperparameter* yang diubah sementara *hyperparameter* lainnya dipertahankan tetap. Setiap variasi konfigurasi dibandingkan dengan model *baseline* menggunakan *confusion matrix* dengan fokus pada metrik akurasi. Tujuan dari analisis ini adalah untuk mengidentifikasi *hyperparameter* yang paling berpengaruh terhadap performa model, menilai sensitivitas *TextCNN ringan* terhadap perubahan parameter, serta menentukan konfigurasi optimal untuk eksperimen lanjutan dan implementasi akhir.

4.2.3.1 *Optimizer*

Tabel 4.17 Hasil studi ablasi *optimizer*

Optimizer	**Hasil *Evaluation Metrix***			
	Acc	**Pre**	**Rec**	**F1**
AdamW	0,883	0,881	0,904	0,892
Muon	**0,897**	**0,889**	**0,923**	**0,905**

Berdasarkan evaluasi *optimizer* yang terlihat pada Tabel 4.17, *Muon* memberikan kinerja terbaik dengan *Acc* 0,897, *Pre* 0,889, *Rec* 0,923, dan *F1* 0,905; hasil ini menunjukkan keunggulan *Muon* dalam mencapai titik konvergensi yang lebih optimal dengan nilai *recall* tertinggi, yang berarti model mampu menangkap lebih banyak sampel positif secara akurat. *AdamW* berada di posisi kedua dengan *Acc* 0,883 dan *F1* 0,892, di mana meskipun memiliki nilai *recall* yang tinggi (0,904), metrik keseluruhannya masih berada di bawah capaian *Muon*. Perbedaan performa ini mengindikasikan bahwa penggunaan *Muon* memberikan peningkatan efisiensi pembelajaran pada arsitektur *TextCNN*, menghasilkan keseimbangan metrik yang lebih superior dibandingkan penggunaan *AdamW* pada konfigurasi yang sama. Berdasarkan data tersebut, urutan prioritas *optimizer* yang berfokus pada *f1-score* dan *recall* adalah *Muon* (0,905), kemudian diikuti oleh *AdamW* (0,892). Dengan

demikian, *Muon* adalah pilihan terbaik untuk mendapatkan hasil klasifikasi yang paling maksimal, sementara *AdamW* tetap merupakan alternatif yang sangat stabil dan andal untuk digunakan.

4.2.3.2 *Batch Size*

Tabel 4.18 Hasil studi ablasi *batch size*

Batch Size	**Hasil *Evaluation Metrix***			
	Acc	**Pre**	**Rec**	**F1**
25	0,866	0,893	0,857	0,871
50	0,883	0,881	**0,904**	**0,892**
100	**0,885**	**0,895**	0,890	**0,892**

Berdasarkan evaluasi *batch size* yang terlihat pada Tabel 4.18, konfigurasi *batch size* 50 dan 100 memberikan *f1-score* tertinggi yang sama yaitu 0,892; namun, *batch size* 50 menjadi pilihan yang lebih unggul karena memiliki nilai *recall* tertinggi (0,904) dibandingkan *batch size* 100 (0,890), sehingga lebih banyak sampel positif yang berhasil diidentifikasi. *Batch size* 100 berada di posisi yang sangat kompetitif dengan akurasi tertinggi (0,885) dan presisi tertinggi (0,895), yang berarti memberikan tingkat kepastian prediksi sedikit lebih baik meskipun *recall*-nya lebih rendah dari *batch size* 50. Konfigurasi *batch size* 25 menghasilkan metrik terendah secara keseluruhan (*Acc* 0,866; *F1* 0,871), menandakan proses pembelajaran yang kurang optimal pada ukuran *batch* yang terlalu kecil. Berdasarkan data tersebut, urutan prioritas konfigurasi adalah *batch size* 50, diikuti oleh *batch size* 100, dan *batch size* 25 sebagai opsi terakhir; sehingga *batch size* 50 direkomendasikan untuk mendapatkan keseimbangan *f1-score* dan *recall* yang paling optimal.

4.2.3.3 *Learning Rate*

Tabel 4.19 Hasil studi ablasi *learning rate*

Learning Rate	**Hasil *Evaluation Metrix***			
	Acc	**Pre**	**Rec**	**F1**
5e-2	0,822	0,857	0,803	0,828
5e-3	0,883	**0,881**	0,904	0,892
5e-4	**0,892**	**0,881**	**0,923**	**0,901**

Data studi ablasi *learning rate* pada Tabel 4.19 mengonfirmasi bahwa nilai *5e-4* memberikan hasil paling optimal dengan *Acc* 0,892, *Rec* 0,923, dan *F1* 0,901, yang menandakan bahwa laju pembelajaran yang lebih kecil memungkinkan model melakukan penyesuaian bobot secara lebih presisi. *Learning rate 5e-3* menempati posisi kedua (*Acc* 0,883; *F1* 0,892), sementara nilai *5e-2* menghasilkan kinerja terendah (*Acc* 0,822; *F1* 0,828) karena laju pembelajaran yang terlalu agresif sering kali melompati titik *minimum loss* yang diinginkan. Berdasarkan prioritas *f1-score* dan *recall*, urutan efektivitasnya adalah *5e-4* (0,901), kemudian *5e-3* (0,892), dan terakhir *5e-2* (0,828), sehingga *5e-4* menjadi parameter standar yang paling disarankan untuk model ini.

4.2.3.4 *Embedding Dimension*

Tabel 4.20 Hasil studi ablasi *embedding dimension*

Embedding Dimension	**Hasil *Evaluation Metrix***			
	Acc	**Pre**	**Rec**	**F1**
50	0,867	0,879	0,874	0,876
100	**0,883**	0,881	**0,904**	**0,892**
200	0,881	**0,883**	0,898	0,890

Hasil studi ablasi dimensi *embedding* pada Tabel 4.20 menunjukkan bahwa penggunaan dimensi 100 memberikan performa terbaik dengan *Acc* 0,883, *Rec* 0,904, dan *F1* 0,892; nilai ini terbukti paling efektif dalam merepresentasikan fitur semantik dari data teks. Penggunaan dimensi 200 berada di posisi kedua

(*Acc* 0,881; *F1* 0,890), sementara penggunaan dimensi 50 memberikan hasil terendah (*Acc* 0,867; *F1* 0,876) karena keterbatasan kapasitas representasi vektor kata. Berdasarkan prioritas metrik, urutan konfigurasinya adalah dimensi 100 (0,892), diikuti oleh dimensi 200 (0,890), dan dimensi 50 (0,876), sehingga konfigurasi embedding dimension 100 dipilih sebagai *hyperparameter* yang paling efisien dan optimal.

4.2.3.5 *Kernel Size*

Tabel 4.21 Hasil studi ablasi *kernel size*

Kernel Size	**Hasil *Evaluation Metrix***			
	Acc	**Pre**	**Rec**	**F1**
[3]	0,877	0,876	0,898	0,887
[3, 4]	**0,883**	**0,881**	**0,904**	**0,892**
[3, 4, 5]	0,877	0,879	0,894	0,886

Evaluasi *kernel size* pada Tabel 4.21 menunjukkan bahwa penggunaan kombinasi *kernel* [3, 4] memberikan hasil paling optimal dengan *Acc* 0,883, *Rec* 0,904, dan *F1* 0,892; kombinasi ini terbukti paling efektif dalam menangkap variasi *n-gram* yang relevan pada dataset ulasan. Penggunaan *kernel* tunggal [3] berada di posisi kedua (*F1* 0,887), sementara penggunaan tiga *kernel* [3, 4, 5] memberikan hasil terendah (*F1* 0,886) yang mengindikasikan adanya redundansi fitur pada *kernel* berukuran lebih besar. Berdasarkan urutan prioritas *f1-score* dan *recall*, urutan konfigurasinya adalah [3, 4] (0,892), kemudian [3] (0,887), dan terakhir [3, 4, 5] (0,886), sehingga ukuran *kernel* [3, 4] ditetapkan sebagai konfigurasi terbaik untuk model ini.

4.2.3.6 *Convolution Filters*

Tabel 4.22 Hasil studi ablasi *convolution filters*

Conv Filters	**Hasil *Evaluation Metrix***			
	Acc	**Pre**	**Rec**	**F1**
25	0,863	0,862	0,887	0,874
50	**0,883**	**0,881**	**0,904**	**0,892**
100	0,855	0,847	0,892	0,868

Evaluasi *convolution filters* pada Tabel 4.22 menunjukkan bahwa penggunaan 50 *filters* tetap menjadi konfigurasi paling optimal dengan *Acc* 0,883, *Rec* 0,904, dan *F1* 0,892, yang menegaskan bahwa jumlah filter ini memberikan kapasitas representasi fitur yang paling seimbang untuk model tersebut. Penggunaan 25 *filters* memberikan hasil yang lebih baik dalam hal *f1-score* (0,874) dibandingkan penggunaan 100 *filters* (0,868), meskipun penggunaan 100 *filters* memiliki nilai *recall* yang sedikit lebih tinggi (0,892) daripada 25 *filters* (0,887). Berdasarkan urutan prioritas *f1-score* dan *recall*, urutan efektivitasnya adalah 50 *filters* (0,892), kemudian 25 *filters* (0,874), dan terakhir 100 *filters* (0,868), sehingga 50 *filters* direkomendasikan sebagai parameter standar untuk meminimalkan *loss* informasi tanpa mengorbankan performa klasifikasi.

4.2.3.7 *Dropout Rate*

Tabel 4.23 Hasil studi ablasi *dropout rate*

Dropout Rate	**Hasil *Evaluation Metrix***			
	Acc	**Pre**	**Rec**	**F1**
0,4	0,877	0,875	0,900	0,887
0,5	**0,883**	**0,881**	**0,904**	**0,892**
0,6	0,866	0,873	0,879	0,876

Berdasarkan evaluasi *dropout rate* pada Tabel 4.23, penggunaan nilai 0,5 memberikan performa paling optimal dengan *Acc* 0,883, *Rec* 0,904,

dan *F1* 0,892; hal ini menunjukkan bahwa tingkat *dropout* tersebut memberikan keseimbangan regulasi yang tepat untuk mencegah *overfitting* tanpa mengorbankan kapasitas pembelajaran model secara berlebihan. Nilai *dropout* 0,4 berada di posisi kedua dengan *F1* 0,887, sementara penggunaan nilai 0,6 menghasilkan metrik terendah (*Acc* 0,866; *F1* 0,876) karena tingkat regulasi yang terlalu tinggi cenderung menghambat model dalam mengekstraksi fitur-fitur linguistik penting secara mendalam. Berdasarkan urutan prioritas *f1-score* dan *recall*, urutan efektivitasnya adalah *dropout* 0,5 (0,892), diikuti oleh 0,4 (0,887), dan terakhir 0,6 (0,876), sehingga konfigurasi *dropout* 0,5 dipilih sebagai *hyperparameter* regulasi yang paling stabil untuk arsitektur ini.

4.3 Perbandingan Hasil *Hyperparameter* Default dengan *hyperparameter* Studi Ablasi

Tabel 4.24 Perbandingan Konfigurasi *hyperparameter Default* dan hasil Studi Ablasi

No	***Hyperparameter***	**Default**	**Studi Ablasi**
1	*optimizers*	AdamW	**Muon**
2	*batch_sizes*	50	50
3	*learning_rates*	5e-3	**5e-4**
4	*embedding_dimension*	100	100
5	*kernel_sizes*	[3, 4]	[3, 4]
6	*convolution_filters*	50	50
7	*dropout_rates*	0.5	0.5

Tabel 4.25 Hasil evaluasi dengan *hyperparameter* default dan hasil studi ablasi

Fold	**Hasil *Confussion Metrix***				**Hasil *Evaluation Metrix***			
	TP	**TN**	**FP**	**FN**	**Acc**	**Pre**	**Rec**	**F1**
1								
2								
3								
4								
5								
Rata-rata								

Tabel 4.26 Perbandingan Hasil evaluasi konfigurasi *hyperparameter* default dan studi ablasi

Konfigurasi *Hyperparameter*	**Rerata Metrik Evaluasi**			
	Acc	**Pre**	**Rec**	**F1**
Default				
Studi Ablasi				

DAFTAR PUSTAKA

[1] Queene Br Sembiring. "Dampak Teknologi Digital Terhadap Kesehatan Mental Generasi Muda". *Circle Archive* 1.4 (2024).

[2] Yessi Mareta Andari Putri et al. "Cyberbullying di media sosial tiktok terhadap remaja sekolah menengah pertama". *Jurnal common* 7.1 (2023), pp. 33–44.

[3] Sameer Hinduja and Justin W Patchin. "The Role of Hope in Bullying and Cyberbullying Prevention". *Frontiers in Sociology* 10 (), p. 1576372.

[4] Simon Kemp. *Digital 2025: Indonesia*. Diakses: 11 Agustus 2025. 2025.

[5] Puput Silva Rosiana et al. "Analisis aplikasi tiktok berdasarkan prinsip dan paradigma interaksi manusia dan komputer menggunakan evaluasi heuristic". *Jurnal Informatika dan Teknik Elektro Terapan* 11.3 (2023).

[6] Bunga Aura Prameswari et al. "Building Prediction Model for Detecting Cyberbullying using TikTok Comments". *2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*. 2023, pp. 1–7.

[7] Shiyang Liao et al. "CNN for situations understanding based on sentiment analysis of twitter data". *Procedia computer science* 111 (2017), pp. 376–381.

[8] Xue Ying. "An overview of overfitting and its solutions". *Journal of physics: Conference series*. Vol. 1168. IOP Publishing. 2019, p. 022022.

[9] Yoon Kim. "Convolutional neural networks for sentence classification". *arXiv preprint arXiv:1408.5882* (2014).

[10] Damar Nugraha and Puji Astuti. “Analisis sentimen cyberbullying pada sosial media Instagram menggunakan metode Support Vector Machine”. *INFORMATION SYSTEM FOR EDUCATORS AND PROFESSIONALS: Journal of Information System* 8.2 (2023), pp. 153–164.

[11] Athallah Zacky Abdullah and Erwin Budi Setiawan. “Sentiment Analysis Accuracy for 2024 Indonesian Election Tweets Using CNN-LSTM With Genetic Algorithm Optimization”. *INTENSIF: Jurnal Ilmiah Penelitian dan Penerapan Teknologi Sistem Informasi* 9.1 (2025), pp. 1–14.

[12] Safrizal Ardana Ardiyansa et al. “Klasifikasi Sentimen Tweet dengan Arsitektur Hybrid Transformers-CNN pada Platform Twitter”. *The Indonesian Journal of Computer Science* 14.3 (2025).

[13] Shuaiyu Chen. “Sentiment Analysis Techniques for Deep Learning Classification and Comparison”. *CONF-CIAP 2025: Proceedings of the 4th International Conference on Computing Innovation and Applied Physics*. Ed. by Omer Burak Istanbullu Marwan Omar Anil Fernando. 2025.

[14] Ahya Ghina Qolbya, Aleissya Sahira Siswandi, and Raissa Dwifandra Putri. “Empati dan Cyberbullying pada Remaja Pengguna Media Sosial: Sebuah Kajian Literatur”. *Flourishing Journal* 3.9 (2023), pp. 352–359.

[15] Sapta Sari. “Literasi media pada generasi milenial di era digital”. *Professional: Jurnal komunikasi dan administrasi publik* 6.2 (2019), pp. 30–42.

[16] Yuan-Fu Liao and Yih-Ru Wang. “Some experiences on applying deep learning to speech signal and natural language processing”. *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*. IEEE. 2018, pp. 83–94.

[17] Agung Pambudi and Suprapto Suprapto. “Effect of sentence length in sentiment analysis using support vector machine and convolutional neural network method”. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)* 15.1 (2021), pp. 21–30.

[18] Ruishuang Wang et al. “Convolutional recurrent neural networks for text classification”. *2019 international joint conference on neural networks (IJCNN)*. IEEE. 2019, pp. 1–6.

[19] Akbar Karimi, Leonardo Rossi, and Andrea Prati. “AEDA: An easier data augmentation technique for text classification”. *arXiv preprint arXiv:2108.13230* (2021).

[20] Jason Wei and Kai Zou. “EDA: Easy data augmentation techniques for boosting performance on text classification tasks”. *arXiv preprint arXiv:1901.11196* (2019).

[21] Slamet Widodo, Herlambang Brawijaya, and Samudi Samudi. “Stratified K-fold cross validation optimization on machine learning for prediction”. *Sinkron: jurnal dan penelitian teknik informatika* 6.4 (2022), pp. 2407–2414.

[22] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[23] Ronan Collobert et al. “Natural language processing (almost) from scratch.” *Journal of machine learning research* 12.7 (2011).