PERBANDINGAN MODEL INDOBERT DAN MODEL *HYBRID* PADA KLASIFIKASI EMOSI PERSEPSI KUALITAS LAYANAN APLIKASI DANA

TUGAS AKHIR

Diajukan oleh:

IKHWANUL HAKIM

NIM. 200705058

Mahasiswa Fakultas Sains dan Teknologi

Program Studi Teknologi Informasi

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI AR-RANIRY

BANDA ACEH

2025

LEMBAR PERSETUJUAN

PERBANDINGAN MODEL INDOBERT DAN MODEL *HYBRID* PADA KLASIFIKASI EMOSI PERSEPSI KUALITAS LAYANAN APLIKASI DANA

TUGAS AKHIR

Diajukan Kepada Fakultas sains dan Teknologi
Universitas Islam Negeri (UIN) Ar-Raniry Banda Aceh
Sebagai Salah Satu Beban Studi Memperoleh Gelar Sarjana
pada Prodi Teknologi Informasi

Oleh:
Ikhwanul Hakim
NIM. 200705058
Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

Disetujui untuk Dimunaqasyahkan Oleh:

Pembimbing I	Pembimbing II
Hendri Ahmadian, S.Si., M.I.M	**Baihaqi, M.T**
NIP. 198301042014031002	**NIP. 198802212022031001**

Mengetahui.
Ketua Program Studi Teknologi Informasi

Malahayati, M.T
NIP. 198301272015032003

LEMBAR PENGESAHAN

PERBANDINGAN MODEL INDOBERT DAN MODEL HYBRID PADA KLASIFIKASI EMOSI PERSEPSI KUALITAS LAYANAN APLIKASI DANA

TUGAS AKHIR

Telah Diuji Oleh Panitia Ujian Munaqasah Tugas Akhir
Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh dan Dinyatakan Lulus
Serta Diterima Sebagai Salah Sayu Beban Studi Program Sarjana (S-1)
Pada Prodi Teknologi Informasi

Pada Hari/Tanggal: Rabu, 15 januari 2025
15 Rajab 1446 H
Di Darussalam, Banda Aceh

Panitia Ujian Munaqasah Tugas Akhir:

Ketua,	Sekretaris,
Hendri Ahmadian, S.Si., M.I.M	**Baihaqi, M.T**
NIP. 198301042014031002	**NIP. 198802212022031001**
Penguji I,	Penguji II,
Raihan Islamadina, M.T	**Malahayati, M.T**
NIP. 198901312020122011	**NIP. 198301272015032003**

Mengetahui:
Dekan Fakultas Sains dan Teknologi
UIN Ar-Raniry Banda Aceh,

Prof. Dr. Ir. Muhammad Dirhamsyah, M.T., I.P.U
NIP. 196210021988111001

LEMBAR PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: Ikhwanul Hakim
NIM	: 200705058
Program Studi	: Teknologi Informasi
Fakultas	: Sains dan Teknologi
Judul Tugas Akhir	: Perbandingan Model IndoBERT dan Model *Hybrid* pada Klasifikasi Emosi Persepsi Kualitas Layanan Aplikasi DANA

Dengan ini menyatakan bahwa dalam penulisan tugas akhir ini, saya:

1. Tidak menggunakan ide orang lain tanpa mampu mengembangkan dan mempertanggungjawabkan;
2. Tidak melakukan plagiasi terhadap naskah orang lain;
3. Tidak menggunakan karya orang lain tanpa menyebutkan sumber asli atau tanpa izin pemilik karya;
4. Tidak memanipulasi dan memalsukan data;
5. Mengerjakan sendiri karya ini dan mampu bertanggungjawab atas karya ini.

Bila dikemudian hari ada tuntutan dari pihak lain atas karya saya, dan telah melalui pembuktian yang dapat dipertanggungjawabkan dan ternyata memang ditemukan bukti bahwa saya telah melanggar pernyataan ini, maka saya siap dikenai sanksi berdasarkan aturan yang berlaku di Fakultas Sains dan teknologi UIN Ar-Raniry Banda Aceh.

Demikian pernyataan ini saya buat dengan sesungguhnya dan tanpa paksaan dari pihak manapun.

Banda Aceh,
Yang Menyatakan

(Ikhwanul Hakim)

ABSTRAK

Nama	: Ikhwanul Hakim
NIM	: 200705058
Program Studi	: Teknologi Informasi
Judul	: Perbandingan Model IndoBERT dan Model *Hybrid* pada Klasifikasi Emosi Persepsi Kualitas Layanan Aplikasi DANA
Tanggal Sidang	: 15 Januari 2025
Jumlah Halaman	: 72
Pembimbing I	: Hendri Ahmadian,S.Si., M.I.M
Pembimbing II	: Baihaqi, M.T

Indonesia berada di era digital, dimana teknologi telah berkembang dengan pesat. Dengan adanya perkembangan ini memunculkan inovasi di bidang finansial, salah satunya dengan adanya layanan aplikasi DANA. DANA merupakan salah satu *e-wallet* yang banyak digunakan di Indonesia yang dapat memudahkan pengguna dalam mengakses pembayaran tanpa uang tunai, memberikan layanan sesuai kebutuhan serta menjamin keamanan dalam transkasi pengguna. Seiring majunya teknologi, akan lebih memudahkan seseorang untuk mengekspresikan diri di media sosial misalnya dengan menyalurkan emosi. Penelitian ini menggunakan 2000 dataset yang diekstrak dari ulasan yang terdapat pada kolom komentar aplikasi DANA serta terdiri dari 5 label emosi yaitu senang, puas, netral, kecewa, dan marah.

Penelitian ini menerapkan model IndoBERT dan model *hybrid* yaitu kombinasi model IndoBERT dengan CNN dan IndoBERT dengan RNN. Berdasarkan hasil penelitan didapatkan bahwa model *hybrid* IndoBERT dengan RNN memperoleh nilai akurasi tertinggi yaitu sebesar 71%. Sedangkan model InoBERT base menghasilkan akurasi sebesar 70%, dan model *hybrid* IndoBERT dengan CNN mendapatkan akurasi 67%. Hal ini menunjukkan bahwa model *hybrid* IndoBERT dengan RNN lebih efektif dalam menganalisis emosi daripada model lain.

Kata Kunci: Klasifikasi Emosi, IndoBERT, Model Hybrid, CNN, RNN, DANA.

ABSTRACT

Name	: Ikhwanul Hakim
Student Number	: 200705058
Department	: Information Technology
Title	: Comparison of IndoBERT and Hybrid Models on Emotion Classification for Perceived Service Quality of the DANA Application
Bate	: 15 January 2025
Number of Pages	: 72
Supervisor I	: Hendri Ahmadian,S.Si., M.I.M
Supervisor II	: Baihaqi, M.T

Indonesia is currently in the digital era, where technology has advanced rapidly. This development has given rise to innovations in the financial sector, one of which is the DANA application service. DANA is one of the e-wallets widely used in Indonesia which can make it easier for users to access cashless payments, provide services according to needs and guarantee security in user transactions. As technology advances, it becomes easier for individuals to express themselves on social media, such as by sharing their emotions. This study used 2000 datasets extracted from reviews in the DANA application comment column and consisted of 5 emotional labels, namely happy, satisfied, neutral, disappointed, and angry.

This study implements the IndoBERT model and hybrid models, which combine IndoBERT with CNN and IndoBERT with RNN. Based on the research findings, the hybrid IndoBERT model with RNN achieved the highest accuracy of 71%. Meanwhile, the baseline IndoBERT model produced an accuracy of 70%, and the hybrid IndoBERT model with CNN reached an accuracy of 67%. These results indicate that the hybrid IndoBERT model with RNN is more effective in emotion analysis compared to other models.

Keywords: Emotion Classification, IndoBERT, Hybrid Model, CNN, RNN, DANA.

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيْم.

Alhamdulillah puji dan syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga skripsi dengan judul **"**Perbandingan Model IndoBERT dan Model *Hybrid* pada Klasifikasi Emosi Persepsi Kualitas Layanan Aplikasi DANA**"** ini dapat dibuat dan terselesaikan. Shalawat beserta salam juga tak lupa penulis sanjung sajikan kepada Nabi Muhammad SAW, yang telah membawa umat manusia ke alam yang penuh dengan ilmu pengetahuan.

Penulisan tugas akhir ini bertujuan untuk memenuhi beban studi guna memperoleh gelar sarjana pada Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh. Pada kesempatan ini penulis mengucapkan terima kasih yang sebesar-besarnya kepada kepada seluruh pihak yang telah terlibat dan membantu dalam menyukseskan penulisan skripsi ini. Dalam hal ini penulis akan mengucapkan terima kasih kepada:

1. Kedua orangtua Ayahanda Nasrullah dan Ibunda Cut Nurkamari yang selalu memberikan semangat, bantuan moril, dan mendoakan keberhasilan, kelancaran, serta keselamatan selama penulis menempuh pendidikan.
2. Saudara kandung Abang Rizki hakiki, Mulkan Sadikin, Malikul Bahri, Alfian Setiawan, dan Kakak Putri Maina yang selalu juga selalu memberikan doa dan motivasi kepada penulis
3. Bapak Dr. Ir. M. Dirhamsyah, M.T., IPU selaku Dekan Fakultas Sains dan Teknologi UIN Ar-Raniry.
4. Ibu Malahayti, M.T dan bapak Khairan AR, M.Kom selaku Ketua dan Sekretaris Program Studi Teknologi Informasi yang telah memberikan bimbingan dan arahan dalam penyusunan skripsi ini.
5. Bapak Hendri Ahmadian, S.Si., M.I.M selaku Pembimbing I dan bapak Baihaqi, M.T selaku Pembimbing II yang telah senantiasa memberikan bimbingan, kritik

dan saran, serta yang telah meluangkan waktu, tenaga dan ilmu untuk membimbing penulis dalam menyelesaikan skripsi ini.

6. Ibu Cut Ida Rahmadiana, S.Si selaku *staff* Prodi Teknologi Informasi, yang senantiasa membantu penulis dalam pemberkasan administrasi.
7. Seluruh dosen Program Studi Teknologi Informsi yang telah memberikan ilmu kepada penulis dalam bidang Teknologi Informasi.
8. Teman-teman seperjuangan Mahasiswa Prodi Teknologi Informasi UIN Ar-Raniry tercinta.
9. Segenap pihak yang telah membantu yang tidak dapat penulis ucapkan namanya satu-persatu

Penulis menyadari masih banyak terdapat kekurangan dalam skripsil ini, untuk itu penulis sangat mengharapkan kritik dan saran yang bersifat membangun demi kesempurnaan skripsi ini.

Banda Aceh, 05 Januari 2025

Penulis

Ikhwanul Hakim

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

I.1 Latar Belakang

Indonesia berada di era digital, dimana teknologi telah berkembang dengan pesat. Dengan adanya perkembangan ini memunculkan inovasi di bidang finansial, salah satunya dengan pembayaran digital *(digital payment)*. Menurut singh et al (2016), digital payment memberikan banyak kemudahan dengan dapat melakukan berbagai transaksi keuangan baik produk maupun jasa menggunakan smartphone. Salah satu instrument dari *digital payment* yaitu uang elektronik (Yuliastuti, 2017). Menurut peraturan Bank Indonesia No.11/12/PBI/2009 tentang uang elektronik *(electronic money)*, uang elektronik merupakan salah satu jenis pembayaran yang memiliki nilai yang sama dengan uang tunai, namun nilai uang disimpan dalam suatu media secara elektronik seperti chip (e-money) serta server (e-wallet) (Bank Indonesia, 2009). *E-money* merupakan alat pembayaran non tunai yang dapat disimpan dalam dompet digital sehingga memudahkan pengguna dalam mengaksesnya (Kumala & Mutia, 2020). Sedangkan e wallet merupakan suatu aplikasi dompet elektronik yang dapat digunakan untuk transaksi antar pengguna sehingga akan lebih mudah untuk di akses oleh seluruh masyarakat (Abrilia, 2020).

Salah satu *e-wallet* yang sedang banyak digunakan oleh masyarakat Indonesia adalah DANA. DANA merupakan suatu bentuk pembayaran/pembelian yang menggunakan perangkat seluler dan merupakan salah satu uang elektronik yang menjadi layanan pembayaran berbasis open-platfrom. Tahun 2023, terdapat sebanyak 170 juta pengguna DANA dimana meningkat 23% dari tahun sebelumnya. Hal ini menunjukkan bahwa terdapat tingkat adopsi yang tinggi serta peningkatan kepercayaan oleh masyarakat terhadap penggunaan aplikasi DANA (Press Realase, 2024).

Seiring majunya teknologi, akan lebih memudahkan seseorang untuk mengekspresikan diri di media sosial misalnya dengan mengeluarkan opini. Dalam bersosial media, seseorang tidak dapat dipisahkan dengan opini, dimana dari opini itu

emosi dapat disalurkan baik verbal atau non-verbal (Syarief, 2017). Emosi verbal merupakan suatu bentuk emosi yang berupa dalam bentuk lisan ataupun tulisan, sedangkan emosi non verbal yaitu bentuk emosi yang diekspresikan dengan menggunakan bahasa tubuh (Chatterjee et al, 2019).

Emosi merupakan suatu bentuk reaksi terhadap seseorang maupun kejadian yang memiliki peranan penting dalam kehidupan sehari-hari. Emosi yang dihasilkan oleh manusi dibagi menjadi dua yaitu positif dan negatif yang meliputi perasaan senang, sedih, marah, takut, dan sebagainya (Bagus dan Fudholi, 2021).

Berdasarkan penelitian terdahulu, klasifikasi emosi pada teks telah diuji dengan beberapa metode, diantaranya LSTM (*Long Short Term Memory*), SVM (*Support Vector Machine*), BiLSTM (*Bi-Directional Long Short Term Memory*), CNN (*Support Vector Machine*), dan Word2Vec (*Word to Vector*). Penelitian tersebut mendapatkan nilai akurasi sebesar 73,15% (Widianto, 2022).

Chiorrini et at (2021) melakukan penelitian analisis emosi dengan menggunakan model Bidiretional Encoder Representations from Transformers (BERT), didapatkan nilai akurasi 90% yang membuktikan bahwa kekuatan pemodelan Bahasa BERT berkontribusi secara signifikan untuk mencapai hasil yang baik.

BERT merupakan teknik berbasis jaringan neural untuk pre-training natural language (Huang et al., 2019). Cara kerja BERT yaitu dengan melatih model bahasa sesuai dengan rangkaian kata yang ada dalam kalimat. BERT juga dapat memungkinkan model bahasa agar mengerti bahasa ambigu dalam teks lalu mengubahnya menjadi menjadi konteks yang benar. Saat ini, sudah terdapat beragam bahasa untuk Model *pre-trained* BERT salah satunya bahasa Indonesia. IndoBERT merupakan model *pre-trained* BERT dalam bentuk bahasa Indonesia (Wilie et al., 2020).

Model hybrid merupakan pendekatan yang menggabungkan berbagai teknik atau algoritma untuk meningkatkan akurasi dan efektivitas dalam mendeteksi emosi dari data seperti teks, suara, atau ekspresi wajah. Penelitian ini akan menggabungkan

model IndoBERT, model *Convolutional Neural Network* (CNN), serta model *Recurrent Neural Network* (RNN). Model hybrid ini akan menggabungkan kekuatan ketiga arsitektur untuk meningkatkan performa dalam tugas-tugas pemrosesan bahasa alami (NLP). IndoBERT sebagai varian BERT yang dioptimalkan untuk bahasa Indonesia, sangat baik dalam memahami konteks dan makna kata dalam teks. Model CNN terkenal karena kemampuannya dalam mengekstraksi fitur-fitur penting dari data spasial atau sekuensial. Di sisi lain, RNN sering digunakan untuk tugas-tugas seperti klasifikasi emosi, karena kemampuannya untuk menangkap hubungan temporal dan pola dalam data. Berbeda dengan jaringan saraf *feedforward*. Dengan mengintegrasikan ketiganya, model ini diharapkan dapat memanfaatkan keunggulan masing-masing arsitektur untuk memberikan hasil yang lebih akurat dan efektif.

Pada penelitian ini akan dilakukan klasifikasi emosi persepsi konsumen terhadap kualitas layanan aplikasi DANA yang diekstrak dari ulasan yang terdapat pada kolom komentar *website Google Play Store* dengan melakukan perbandingan model IndoBERT dengan model hybrid yaitu IndoBERT dengan CNN dan IndoBERT dengan RNN.

I. 2 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah:

1. Bagaimana penerapan model IndoBERT dalam klasifikasi emosi pada komentar di aplikasi DANA?
2. Bagaimana penerapan model IndoBERT dengan CNN dan IndoBERT dengan RNN dalam klasifikasi emosi pada komentar di aplikasi DANA?
3. Bagaimana perbandingan kinerja model IndoBERT, IndoBERT dengan CNN, dan IndoBERT dengan RNN dalam klasifikasi emosi pada komentar di aplikasi DANA?

I.3 Tujuan Penelitian

Penelitian ini memiliki tujuan sebagai berikut:

1. Menerapkan model IndoBERT dalam klasifikasi emosi pada komentar di aplikasi DANA.
2. Menerapkan model IndoBERT dengan CNN dan IndoBERT dengan RNN dalam klasifikasi emosi pada komentar di aplikasi DANA
3. Membandingkan dan menganalisis hasil kinerja antara model IndoBERT, IndoBERT dengn CNN dan IndoBERT dengan RNN dalam klasifikasi emosi pada komentar di aplikasi DANA

I.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut:

1. Pembuatan sistem dibangun menggunakan bahasa Python dan dieksekusi menggunakan *Google Colab*.
2. Pengklasifikasian berdasarkan 5 emosi, yaitu: marah, kecewa, netral, puas, senang.
3. Data yang digunakan merupakan data ulasan pengguna aplikasi DANA yang berbasa Indonesia di *website Google Play Store*.

I.5 Manfaat Penelitian

Penelitian ini memiliki manfaat sebagai berikut:

1. Bagi penulis dapat mengetahui perbandingan klasifikasi emosi menggunakan model IndoBERT, model IndoBERT dengan CNN, dan model IdoBERT dengan RNN.
2. Bagi *mobile app developer* dapat dimanfaatkan sebagai pertimbangan dalam menciptakan dan mengembangkan layanan aplikasi.
3. Bagi pembaca, dapat dimanfaatkan sebagai referensi serta bahan acuan untuk digunakan dalam penelitian berikutnya.

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian yang dilakukan oleh Viana (2023) tentang klasifikasi emosi opini Twitter menggunakan model *Benchmark* IndoNLU menggunakan 3000 dataset yang terdiri dari 6 klsifikasi emosi diantaranya *anger, fear, joy, love sad,* dan *neutral.* Penelitian ini menggunakan model IndoBERT yang dikembangkan oleh IndoNLU, dimana model yang diuji merupakan variasi dari model IndoBERT yaitu IndoBERT base p1, IndoBERT base p2, IndoBERT large p1, dan IndoBERT large p2. Hasil penelitian didapatkan nilai akurasi sebesar 71.47% untuk model IndoBERT base p1, 71.19% untuk IndoBERT base p2, 73.59% untuk IndoBERT large p1, dan 72.60% untuk IndoBERT large p2.

Penelitian lain juga dilakukan oleh Andika (2023) tentang analisis sentiment persepsi konsumen terhadap kualitas layanan aplikasi BSI Mobile menggunakan IndoBERT. Dataset penelitian ini sebanyak 23931 data yang terdiri dari data komentar dan label sentiment. Dari penelitian ini didapatkan hasil bahwa performa model IndoBERT didapatkan hasil terbaik pada model dengan *batch size* 16 dan *epoch* 5 yang menggunakan dataset pelabelan sesuai *rating* dengan *accuracy* 0.903, *precision* 0.900, *recall* 0.903 dan *f*1-*score* 0.901.

Penelitian yang dilakukan Widianto (2022) tentang Klasifikasi Emosi pada Teks Menggunakan Metode *Deep learning.* Penelitian ini menggunakan metode klasifikasi dengan Bidirectional Encoder Representations from Transformers (BERT). Data yang diuji sebanyak 2515 data dan diklasifikasikan ke dalam Sembilan klasifikasi emosi yaitu marah, bahagia, sedih, takut, netral, tertarik, percaya, jijik, dan kaget. Pada penelitian ini didapatkan nilai akurasi untuk klasifikasi emosi menggunakan BERT sebesar 90% pada BERT *Uncased* dan 81% pada model *Benchmark* IndoNLU.

Penggunaan model Bidirectional Encoder Representations from Transformers (BERT) untuk analisis sentimen dan pengenalan emosi data Twitter juga diteliti oleh Chiorrini et al (2021). Pada analisis sentimen diklasifikasi menjadi positif, negatif, dan netral. Sedangkan klasifikasi emosi terdiri dari empat emosi yaitu marah, takut, bahagia, dan sedih. Penelitian ini menunjukkan bahwa masing-masing memperoleh akurasi 92% untuk analisis sentiment dan 90% untuk analisis emosi.

Berdasarkan penelitian tentang klasifikasi emosi menggunakan *Convolutional Neural Networks* (CNN) yang dilakukan oleh Septian et al (2019) didapatkan hasil bahwa metode CNN berhasil mendeteksi ketujuh klasifikasi emosi yaitu marah, senang, sedih, jijik, terkejut, dan netral dengan nilai akurasi deteksi 67%, presisi 67% dan *recall* 66%.

Penelitian oleh Umutsalur dan Aydin (2017) tentang model hybrid deep learning untuk klasifikasi sentiment didapatkan hasil akurasi terbaik klasifikasi sebesar 80,44% dengan kombinasi BiLSTM dan FastText (yaitu M-7-B). Di sisi lain, hasil akurasi terbaik klasifikasi sebesar 75,67% dengan kombinasi CNN dan representasi tingkat karakter (yaitu M-1-A). Selain itu, juga didapatkan hasil sebesar 82,14% dengan model hybrid yang menggabungkan model M-1-A dan M-7-B.

Ringkasan tentang penelitian terdahulu yang digunakan pada penelitian ini dapat dilihat pada tabel 2.1.

Tabel 2.1 Perbandingan Penelitian Terdahulu

No	Nama Peneliti	Judul Penelitian	Hasil Penelitian
1	Viana (2023)	Klasifikasi Emosi Opini Twitter Menggunakan Model *Benchmark IndoNLU*	Hasil penelitian didapatkan nilai akurasi sebesar 71.47% untuk model IndoBERT base p1, 71.19% untuk IndoBERT base p2, 73.59% untuk IndoBERT large p1,

			dan 72.60% untuk IndoBERT large p2
2	Andika (2023)	Analisis Sentimen Persepsi Konsumen terhadap Kualitas Layanan Aplikasi BSI Mobile Menggunakan Model IndoBERT	Hasil penelitian menunjukkan bahwa performa model IndoBERT didapatkan hasil terbaik pada model dengan *batch size* 16 dan *epoch* 5 yang menggunakan dataset pelabelan sesuai *rating* dengan *accuracy* 0.903, *precision* 0.900, *recall* 0.903 dan *f*1-*score* 0.901
3	Widianto (2022)	Klasifikasi Emosi pada Teks Menggunakan Metode *Deep learning*	Nilai akurasi untuk klasifikasi emosi menggunakan BERT sebesar 90% pada BERT *Uncased* dan 81% pada model *Benchmark* IndoNLU
4	Chiorrini et al (2021)	Emotion and Sentiment Analysis of Tweets using BERT	Hasil penelitian menunjukkan bahwa nilai akurasi sebesar 92% untuk analisis sentiment dan 90% untuk analisis emosi.
5	Septian et al (2019)	Klasifikasi Emosi Menggunakan *Convolutional Neural Networks*	Hasil penelitian didapatkan kalsifikasi emosi menggunakan CNN mendapatkan nilai akurasi

			deteksi 67%, presisi 67% dan *recall* 66%
6	Umutsalur dan Aydin (2017)	A Novel Hybrid Deep Learning Model for Sentiment Classification	Hasil penelitian menunjukkan nilai akurasi sebesar 82,14% dengan model hybrid yang menggabungkan model M-1-A dan M-7-B.

2.2 DANA

Dana merupakan suatu layanan keuangan digital yang didirikan sejak tahun 2018 dan berperan sebagai aplikasi pembayaran digital yang digunakan untuk menggantikan dompet konvensional. DANA juga merupakan layanan dompet digital yang memiliki empat lisensi yang terdaftar di Bank Indonesia yaitu dompet digital, kirim uang, uang elektronik, dan Likuiditas Keuangan Digital(LKD). DANA mempunyai tiga pilar, yaitu *friendly* (memberikan pengalaman dan layanan yang terbaik sesuai kebutuhan pengguna), *accessible* (memudahkan pengguna dan pelaku usaha dalam mengakses pembayaran tanpa uang tunai), dan *trusted* (menjamin keamanan menyeluruh dalam transaksi pengguna (Idayanti dan Ulandari, 2023).

2.3 Klasifikasi Emosi

Pada tahun 1966 sistem survei analisi emosi pertama kali dilakukan dan merupakan fase awal dari analisis emosi pada teks. Menurut Robert Plutchik, terdapat delapan kategori utama pada emosi, dimana sebagian merupakan emosi positif dan sebagian lagi merupakan emosi negatif. Setiap emosi memiliki subkategori yang ada. Misalnya bahagia bisa mencerminkan keadaan kesenangan atau kegembiraan, tergantung pada konteks yang ada (Widianto, 2022).

Emosi merupakan sesuatu yang mendorong perasaan yang diwujudkan dalam bentuk tingkah laku. Menurut Krech (1969) dalam Nafisa (2024), emosi dapat diklasifikasikan menjadi 4 jenis, yaitu sebagai berikut:

2.3.1 Emosi Dasar

Emosi ini dapat dipicu oleh situasi-situasi sederhana. Jenis emosi yang termasuk dalam emosi dasar adalah senang, marah, takut, dan sedih. Rasa senang yaitu bentuk ungkapan kegembiraan atas keberhasilan dalam mencapai tujuan yang diinginkan. Rasa marah dapat dipicu karena adanya hambatan dalam mencapai tujuan atau karena sesuatu yang tidak disukai terjadi. Takut merupakan emosi penghindaran yang melibatkan pelarian diri dari bahaya yang mengancam dan dapat muncul karena tidak mampu menghadapi ancaman bahaya. Sedangkan sedih merupakan sesuatu yang berkaitan erat dengan kehilangan suatu yang berharga.

2.3.2 Emosi yang Dipicu oleh Stimulasi Sensorik

Emosi ini berhubungan erat dengan rangsangan indra, baik yang positif maupun negatif, yang berasal dari benda dan objek. Emosi ini terbagi menjadi tiga kategori utama yaitu rasa sakit, jijik, dan kenikmatan. Rasa sakit merujuk pada pengalaman buruk yang dirasakan oleh tubuh, pikiran, atau jiwa. Rasa jijik dapat muncul dari pengalaman melihat, mencium, mencicipi, atau menyentuh suatu objek. Sementara itu, kenikmatan adalah perasaan menyenangkan yang muncul ketika berinteraksi dengan sesuatu yang memiliki daya tarik positif.

2.3.3 Emosi yang Berhubungan dengan Penilaian Diri

Perasaan seperti keberhasilan dan kegagalann, malu, bangga, bersalah, dan penyesalan melibatkan penilaian diri terhadap perilaku dalam konteks berbagai standar penting. Penilaian ini bisa berasal dari penilaian pribadi atau tanggapan dari orang lain.

2.3.4 Emosi yang Berhubungan dengan Orang Lain

Sebagian besar pengalaman emosional melibatkan hubungan diri dengan orang lain. Orang-orang tersebut merupakan objek untuk menyalurkan perasaan. Yang termasuk dalam emosi ini adalah cinta dan benci. Cinta merupakan perasaan yang hadir dari perasaan tertarik dan keinginan untuk dapat meraih kesenangan bersama. Sedangkan benci merupakan perasaan tidak suka atau permusuhan terhadap seseorang, hewan, barang, atau peristiwa.

2.4 *Bidirectional Encoder Representations from Transformers* (BERT)

BERT merupakan teknik pembelajaran mesin berbasis transformator yang yang diterbitkan tahun 2018 oleh Jacob Devlin dari Google bersama rekannya (Saraswati et al, 2023). BERT adalah model berbasis *encoder* dalam arsitektur *Transformer* yang dirancang untuk *Natural Language Processing* (NLP). Model ini menggunakan mekanisme perhatian sebagai pengganti jaringan berulang, memungkinkan pengenalan hubungan kontekstual antar kata yang berjauhan. BERT menghasilkan representasi untuk setiap kata dalam sebuah kalimat, sehingga mampu meningkatkan kinerja model pada berbagai tugas kompleks dalam NLP yang melibatkan urutan data (Chandradev et al, 2023).

Transformer memiliki dua komponen utama, yaitu *encoder* dan *decoder*. *Encoder* berfungsi untuk memproses teks input dan menghasilkan *representasi* berbasis *attention* yang mampu mengidentifikasi informasi penting dari konteks secara efektif. *Encoder* terdiri dari enam lapisan identik yang tersusun secara berurutan. Setiap lapisan memiliki dua sub-komponen utama: lapisan *self-attention* dan jaringan saraf *feedforward*. Pada lapisan *self-attention*, setiap node dapat memperhatikan semua kata yang sedang diproses sekaligus memahami konteksnya. Selain itu, setiap posisi di *encoder* mampu menganalisis seluruh posisi dari lapisan sebelumnya dalam *encoder* (Saraswati et al, 2023).

Decoder berfungsi untuk menghasilkan urutan keluaran berdasarkan prediksi. *Decoder* juga terdiri dari N=6 lapisan, dimana setiap lapisan memiliki dua sub-lapisan yang serupa dengan lapisan *encoder*. Dengan *attention layer* tambahan di antara keduanya untuk membantu node saat ini mengakses konten utama yang diinginkan dengan melakukan perhatian *multi-head* pada *output encoder*. Sama seperti pada *encoder*, lapisan *self-attention* di *decoder* memungkinkan setiap posisi dalam decoder untuk memperhatikan semua posis sebelumnya dan posisi saat ini.

Terdapat 2 tahapan dalam kinerja *framework* BERT, yaitu *pre-training* dan *fine-tuning*. Pada tahapan *pre-training,* model dilatih pada dataset besar yang belum memiliki label. Model belajar tentang bahasa melalui proses gabungan *Masked Language Modeling* (MLM) dan *Next Sentence Prediction* (NSP). MLM mengubah kata acak dalam kalimat menjadi token [MASK] dan tugas model memprediksi kata-kata tersebut berdasarkan konteks kiri dan kanan. NSP membantu model memahami hubungan antara dua kalimat dalam pasangan teks (Devlin et al., 2018).

2.4.1 Pre-Training *Bidirectional Encoder Representations from Transformers* (BERT)

a. *Masked Language Modelling* (MLM), merupakan teknik pembelajaran mesin yang digunakan dalam pemrosesan Bahasa alami. Teknik ini memiliki tujuan untuk memprediksi kata yang dihilangkan dalam teks yang diberikan. Biasanya MLM digunakan dalam model arsitektur pemrosesan Bahasa alami yang berbasis *transformer* seperti BERT.

   Teknik MLM bekerja dengan cara mengambil dan menghilangkan beberapa kata secara acak dalam setiap aklimat. Model tersebut kemudian memprediksi kata sebenarnya sesuai konteks kalimat *vocabulary*. Supaya model berhasil memprediksi kata, model harus ditambahkan lapisan klasifikasi di atas *output encoder*. Lalu vector output dikalikan dengan matriks embedding untuk mengubahnya ke dalam dimensi *vocabulary*. Kemudian dilakukan perhitungan probabilitas dari setiap kata dalam *vocabulary* menggunakan *softmax*.

b. *Next Sentence Prediction* (NSP), dalam model ini diberikan dua kalimat dan harus memprediksi apakah kalimat kedua mengikuti kalimat pertama. Tugas ini bertujuan untuk melatih model dalam memahami hubungan antara dua kalimat dan memperoleh pemahaman yang lebih baik tentang konteks kalimat pada dataset.

Hasil dari *pre-training* menghasilkan model BERT dengan bobot (*weight*) dan representasi vector dari kata-kata dalam korpus teks yang digunakan untuk pelatihan. Dalam tahap *pre-training*, BERT belajar untuk memahami hubungan antar kata dalam konteks Bahasa dengan tujuan akhir mengembangkan representasi kata yang kaya kontekstual. Representasi vector ini kemudian dapat digunakan untuk tugas-tugas pemrosesan bahasa alami seperti klasifikasi, question-answering, dan lainnya melalui tahap *fine-tuning* (Rahmatullah, 2022).

2.4.2 *Fine-Tuning Bidirectional Encoder Representations from Transformers* (BERT)

Fine-tuning Bidirectional Encoder Representations from Transformers (BERT) adalah teknik yang digunakan untuk memodifikasi model pemrosesan bahasa alami BERT dengan melatih model pada tugas tertentu yang spesifik, seperti klasifikasi teks atau pengenalan entitas bernama. Proses ini dilakukan dengan cara mengubah sedikit parameter atau struktur model yang ada sehingga dapat melakukan tugas yang baru.

Dalam fase *fine-tuning*, semua *hyperparameter* yang digunakan dalam proses *pre-training* tidak bervariasi kecuali *learning rate*, *batch size* dan *epoch*. Lapisan terakhir dari model BERT yang disebut "*classifier layer*", biasanya diubah untuk mengakomodasi tugas yang spesifik. Kemudian, model diterapkan pada tugas spesifik data tersebut dan bobotnya ditingkatkan secara iteratif melalui algoritma pembelajaran mesin.

2.5 IndoBERT

IndoBERT merupakan *pretrained* model yang didasarkan pada model BERT yang dilatih dengan data teks bahasa Indonesia sebanyak 4 miliar kata yang berasal dari berita daring, media sosial, Wikipedia, artikel daring, subtitle dari rekaman video, dan parallel dataset yang kemudian disebut dengan Indo4B. Untuk IndoBERT, pada tahap pretraining, dilatih menggunakan dataset Indo4B. Kemudian pada tahap fine tuning, model yang telah dilatih pada tahap pra-pelatihan disesuaikan dan kemudian dilatih dengan kumpulan data tertentu yang memiliki label (Jayadianti et al, 2022).

IndoBERT memiliki empat varian model, yakni IndoBERT-Base, IndoBERT-Large, IndoBERT-liteBase, IndoBERT-liteLarge. Semua model IndoBERT dilatih menggunakan TPU v3-8 dalam dua tahapan. Tahapan pertama model dilatih dengan *maximum sequence length* 128, dan tahapan kedua model dilatih dengan *maximum sequence length* 512 (Wilie et al., 2020). Dari semua model IndoBERT, IndoBERT-Large memiliki keakuratan paling tinggi, namun memerlukan kapasitas memori yang besar (Rahmatullah, 2021). IndoBERT-Large merupakan modifikasi dari BERT-large yang sudah mengikuti kofigurasi dari BERT-Large (uncased) yang memiliki 24 *hidden layers*, 1024 *hidden size*, 16 *attention heads*, 4096 dimensi *feed-forward hidden layers* dan *languange type* Monolingual (Wilie et al., 2020).

2.6 Python

Python merupakan bahasa pemograman yang menggunakan interpreter untuk menjalankan kode secara langsung. Bahasa ini dapat beroperasi di berbagai platform seperti windows dan Linux. Selain itu, python menggabungkan beberapa paradigma pemograman dari bahasa lain, seperti pemograman procedural dari bahasa C, pemograman berorientasi objek dari java, dan pemograman fungsional dari Lips. Kombinasi paradigma ini memudahkan programmer dalam mengembangkan berbagai proyek. Python juga menawarkan integrase dengan sistem basis data dan kemampuan untuk membaca serta momodifikasi file, menjadikannya pilihan popular untuk

prototyping dan pengembang perangkat lunak yang cepat dan andal (Rahman et al, 2023).

Berikut ini merupakan berbagai macam *librabry* Python yang dapat digunakan untuk melakukan analisis data (Nongthombam & Sharma, 2021):

2.6.1 Numpy (Numerical Python)

Numpy merupakan *library* Python yang berfokus pada komputasi dan numerik. Numpy menyediakan struktur data dan fungsi untuk mendukung operasi matematika yang efisien dan cepat terutama untuk array dan matriks multimedia

2.6.2 Scrapy

Scrapy merupakan *framework open-source* yang digunakan untuk mengekstraksi data dari situs web. Scrapy juga dapat membuat program yang bisa mengekstraksi informasi dari berbagai situs web dan menyimpannya dalam format yang terstruktur, seperti CSV atau JSON.

2.6.3 Scikit-Learn

Scikit-Learn (sklearn) merupakan python *open source* yang dapat digunakan dalam *machine learning* serta dapat digunakan untuk membagi data menjadi subset pelatihan dan pengujian.

2.6.4 Panda

Panda atau sering disebut pandas yaitu sebuah *library* python yang digunakan dalam analisis data dan kecerdasan buatan. Panda menyediakan dukungan untuk struktur data yang fleksibel serta efisien untuk memanipulasi dan menganalisis data terutama tabular data seperti *spreadsheet* atau database.

2.6.5 TextBlob

TextBlob merupakan *library* python yang digunakan untuk memproses data teks secara sederhana. *Library* ini menyediakan antarmuka yang mudah digunakan

untuk melakukan berbagai tgas pemrosesan teks, misalnya analisis sentiment, pengklasifikasian teks, pemecahan frasa, dan lainnya.

2.6.6 Matplotlib

Matplotlib merupakan *library* python yang digunakan untuk membuat visualisasi data dengan berbagai jenis plot, grafik dan gambar.

2.7 *Google Colaboratory*

Google colaboratory merupakan platform cloud yang disediakan oleh google untuk menjalankan dan menulis kode python melalui browser tanpa memerlukan konfigurasi tambahan. *Google colaboratory* juga memanfaatkan infrastruktur cloud google dan dapat diakses melalui peramban web, tidak memerlukan instalasi perangkat lunak tambahan, dan menyediakan kemudahan kolaborasi dengan memungkinkan pengguna untuk berbagi motebook secara langsung dengan orang lain (Nazar, 2024).

2.8 *Confusion Matrix*

Confusion matrix merupakan suatu tabel yang menunjukkan klasifikasi jumlah data yang diuji baik yang benar dan yang salah. Tabel ini menggambarkan jumlah data yang diprediksi dengan benar atau salah oleh model dan dibagi menjadi empat bagian yaitu *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (Normawati dan Prayogi, 2021).

True Positive (TP) merupakan data dengan kondisi aktual yang berhasil diprediksi secara tepat. *False Positive* (FP) merupakan sebuah data positif yang diprediksi tidak sesuai. *True Negative* (TN) menggambarkan data negatif yang diprediksi dengan benar. *False Negative* (FN) mengacu pada sebuah data negatif yang diprediksi tidak sesuai.

Bentuk dasar *confusion matrix* dapat dilihat pada tabel 2.2.

Tabel 2.2 *Confusion Matrix*

		Predicted Class	
		Positive	*Negative*
Actual Class	*Positive*	TP	FN
	Negative	FP	TN

Dengan menggunakan tabel *confusion matrix*, berbagai nilai yang digunakan untuk evaluasi performa model seperti *accuracy*, *precision*, *recall*, dan *f*1-*score* dapat dihitung untuk mengetahui performa model dalam melakukan analisis sentimen.

2.8.1 *Accuracy* (Akurasi)

Accuracy merupakan salah satu matriks evaluasi performa model yang mengukur seberapa banyak prediksi yang benar dari total jumlah data yang diprediksi. Untuk menghitung nilai *accuracy* menggunakan persamaan berikut (Imron et al, 2023):

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

2.8.2 *Recall*

Recall merupakan salah satu matriks evaluasi performa model yang mengukur seberapa banyak data positif yang berhasil diidentifikasi dari total jumlah data positif yang sebenarnya. *Recall* juga merupakan rasio antara jumlah data positif yang diprediksi benar dengan jumlah total data positif yang sebenarnya. Untuk menghitung nilai *recall* menggunakan persamaan berikut (Imron et al, 2023):

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

2.8.3 *Precision*

Precision merupakan salah satu matriks evaluasi performa model yang mengukur seberapa banyak prediksi positif yang benar dari total jumlah data yang diprediksi positif. *Precision* merupakan rasio antara jumlah data positif yang diprediksi

benar dengan jumlah total data yang diprediksi positif. Untuk menghitung nilai *precision* menggunakan persamaan berikut (Imron et al, 2023):

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

2.8.4 F1-Score

*F*1-*score* didefinisikan sebagai salah satu matriks evaluasi performa model yang mengintegrasikan *precision* dan *recall* ke dalam satu nilai untuk menggambarkan keseimbangan antara kedua matriks tersebut. *F*1-*score* dapat dianggap sebagai rata-rata harmonik dari *precision* dan *recall*, skor tertinggi diperoleh ketika *precision* dan *recall* sama-sama tinggi. Untuk menghitung nilai *f1-score* menggunakan persamaan berikut (Imron et al, 2023):

$$\text{F1-Score} = \frac{2TP}{2TP+FP+FN}$$

2.9 Model *Hybrid*

Model *hybrid* adalah pendekatan yang menggabungkan berbagai teknik atau algoritma untuk meningkatkan akurasi dan efektivitas dalam mendeteksi emosi dari data seperti teks, suara, atau ekspresi wajah. Pendekatan ini sering mengintegrasikan metode pembelajaran mesin dengan teknik berbasis aturan atau model statistik. Sebagai contoh, model hybrid dapat memadukan algoritma *deep learning* untuk ekstraksi fitur kompleks dengan metode tradisional seperti pembobotan kata. Dengan memanfaatkan kekuatan masing masing metode, seperti kemampuan *deep learning* dalam mengenali pola kompleks dan kelebihan teknik berbasis aturan dalam memahami konteks, model *hybrid* bertujuan untuk memberikan hasil klasifikasi emosi yang lebih akurat.

2.10 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah jenis jaringan neural tiruan yang sangat cocok untuk menangani tugas-tugas yang berhubungan dengan data visual, seperti pengenalan wajah, analisis dokumen, klasifikasi gambar dan klasifikasi video.

Meskipun pada awalnya CNN dirancang untuk mengolah data visual, penggunaannya juga telah terbukti efektif dalam klasifikasi emosi pada teks.

Dalam model CNN, setiap kalimat yang telah melalui tahapan pemrosesan awal dan vektorisasi akan di proses lebih lanjut dengan memasukkannya ke dalam embedding layer, yang menghasilkan embedding matrix dari vector input tersebut. Selanjutnya setiap kolom dalam embedding matrix dikenakan proses konvolusi pada convolutional layer, menghasilkan feature maps. *Feature maps* yang dihasilkan kemudian diproses lebih lanjut oleh max-pooling layer untuk mengekstrak fitur dengan nilai yang maksimum. Setelah itu, *feature maps* yang telah dipilih diteruskan ke *fully connected layer*, dimana fungsi aktivasi *Rectified Lenear Unit* (ReLu) digunakan untuk mengubah nilai negatif menjadi nol, sedangkan pembobotan dilakukan pada layer ini. Hasil dari *fully connected layer* kemudian dikirim ke output layer yang terdiri dari enam neuron dengan fungsi aktivasi sigmoid, yang memungkinkan setiap neuron menghasilkan nilai dalam rentang tertentu. Setiap neuron pada output layer mewakili salah satu lebel dari emosi dasar (Aripin et al, 2021).

2.10.1 *Embedding Layer*

Setiap input diproses melalui *word embedding layer,* yang berfungsi mengonversi setiap vector input menjadi matriks embedding. Matriks ini memetakan setiap kata ke dalam ruang vector berdimensi lebih tinggi.

2.10.2 *Convolutional Layer*

Convolutional layer bertugas mengambil fitur dari lapisan embedding dengan menggunakan konvolusi satu dimensi.

2.10.3 *Max Pooling Layer*

Feature maps yang dihasilkan dari convolutional layer satu dimensi diteruskan ke *max-pooling layer*. Di *max-pooling layer*, setiap *feature maps* diproses dengan mengambil nilai maksimum dari setiap *feature maps*. Proses ini mempercepat pelatihan sambil tetap mempertahankan informasi penting dari data.

2.10.4 *Output Layer*

Output layer dalam penelitian ini terdiri dari lima neuron yang masing-masing mewakili satu dari lima jenis emosi yang tersedia. Pada *output layer,* digunakan f*ungsi* aktivasi sigmoid yang menghasilkan nilai independent untuk setiap neuron. Hal ini memungkinkan setiap neuron yang mewakili emosi tertentu memiliki nilai dalam rentang tertentu, sehingga dapat memungkinkan terbentuknya kombinasi beberapa emosi yang dihasilkan.

2.11 *Recurrent Neural Network (*RNN)

Recurrent Neural Network (RNN) merupakan salah satu jenis arsitektur jaringan saraf tiruan yang dirancang untuk memproses data berurutan, seperti teks, suara, atau video. Dalam konteks *machine learning*, RNN sering digunakan untuk tugas-tugas seperti klasifikasi emosi, karena kemampuannya untuk menangkap hubungan temporal dan pola dalam data. Berbeda dengan jaringan saraf *feedforward*, RNN memiliki memori internal yang memungkinkan informasi dari langkah sebelumnya digunakan dalam proses pengambilan keputusan pada langkah berikutnya.

Pada klasifikasi emosi, misalnya dalam analisis teks, RNN dapat digunakan untuk memahami konteks dari suatu kalimat atau komentar. Dengan mempertimbangkan urutan kata, RNN mampu mengenali pola emosional yang terkandung dalam teks, seperti kebahagiaan, kesedihan, kemarahan, atau kekecewaan. Setiap kata dalam kalimat diproses secara berurutan, dan informasi dari kata-kata sebelumnya digunakan untuk membantu memahami makna keseluruhan. Namun, RNN tradisional memiliki keterbatasan dalam menangani dependensi jangka panjang karena masalah peluruhan gradien (vanishing gradient). Oleh karena itu, varian seperti *Long Short-Term Memory* (LSTM) atau *Gated Recurrent Unit* (GRU) sering digunakan untuk mengatasi masalah ini.

Dalam penerapannya untuk klasifikasi emosi, model RNN biasanya dilatih menggunakan dataset teks yang telah diberi label emosi. Proses pelatihan melibatkan pembelajaran representasi pola dari data tersebut sehingga model dapat memprediksi

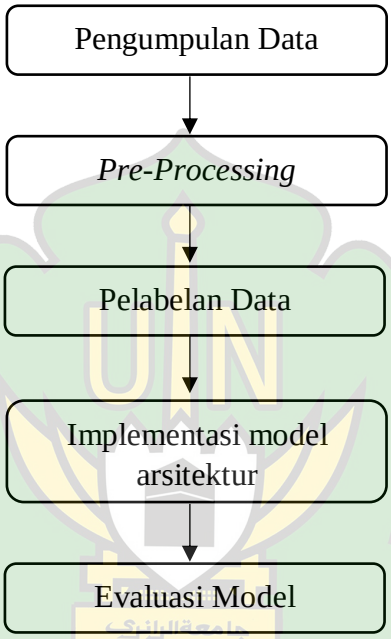
emosi berdasarkan teks baru. Dengan demikian, RNN menjadi alat yang sangat berguna dalam memahami dan menganalisis emosi, baik untuk kebutuhan aplikasi seperti *chatbot*, analisis sentimen, maupun sistem rekomendasi berbasis emosi.

BAB III

METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Tahapan penelitian merupakan serangkaian langkah dan proses yang perlu diikuti oleh peneliti dalam merancang, implementasi, serta mengevaluasi sebuah penelitian. Gambar 3.1 merupakan tahapan penelitian.



Gambar 3.1 Tahapan Penelitian

3.2 Pengumpulan Data

Penelitian ini melakukan pengumpulan data dengan menggunakan data ulasan konsumen yang berbahasa Indonesia terhadap aplikasi DANA di *website Google Play Store*. Pengumpulan data dilakukan dengan cara *scraping*, dimana informasi diambil

secara otomatis pada kolom komentar yang terdapat pada link https://play.google.com/store/apps/details?id=id.dana.

3.3 *Pre-Processing*

Tahap *pre-processing* adalah langkah untuk menghapus data yang tidak penting atau tidak diperlukan, sehingga data menjadi lebih sederhana untuk diolah dan mampu menghasilkan *output* yang sesuai dengan tujuan penelitian.Tahap ini melalui beberapa proses diantaranya sebagai berikut:

- *Case Folding*, yaitu proses mengubah kata-kata menjadi huruf kecil sehingga data yang diproses memiliki format yang seragam.
- *Filtering,* adalah metode yang digunakan untuk menghilangkan kata-kata yang kurang informatif atau tidak relevan dalam sebuah teks, seperti kata depan, kata gantu, atau kata sambung.
- Normalisasi teks, yaitu melakukan normalisasi teks untuk mengubah teks menjadi format yang konsisten seperti menghilangkan tanda baca atau karakter khusus, mengubah bahasa slang, dan mengganti bentuk singkatan atau akronim dengan bentuk lengkapnya.

3.4 Pelabelan Data

Tahap pelabelan data dilakukan secara manual dengan melakukan analisis mendalam terhadap setiap ulasan untuk menganali emosi yang dominan. Pelabelan data dilakukan untuk mendapatkan kualitas data yang baik. Terdapat lima label emosi yang digunakan dalam penelitian ini yaitu marah, kecewa, netral, puas, dan senang. Contoh kalimat yang sudah di label dapat dilihat pada tabel di bawah ini.

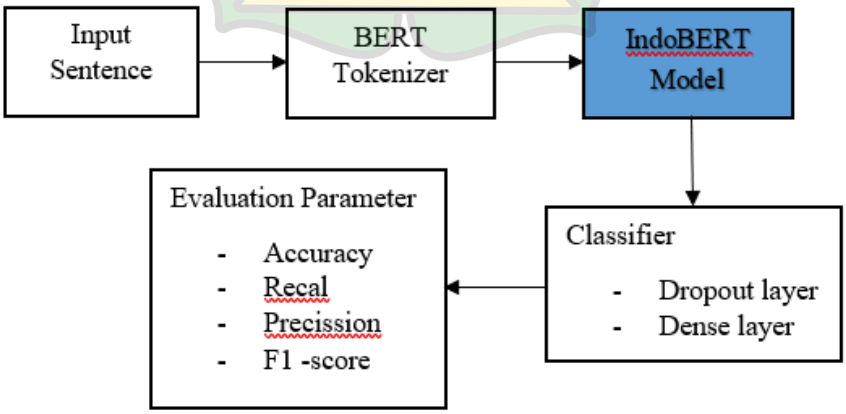
Tabel 3.1 Contoh Pelabelan Data

No	Contoh Data	Label
1	Senang sekali menggunakan aplikasi dana mudah, gampang, dan cepat	Senang
2	Aplikasi yang sangat membantu	Puas
3	Apakah dana lagi error?	Netral
4	Kecewa banget sama keamanan dana ini saldo saya berkurang tapi ngga ada riwayat transaksinya	Kecewa
5	Sudah saya coba semua ya saran nya percuma tetap tidak bisa duit saya nyangkut di sana bagaimana ini penipuan tolol	Marah

3.5 Implementasi Model dan Arsitektur

3.5.1 IndoBERT

Tahap ini bertujuan untuk mentransformasikan serta merealisasikan rancangan konsep penelitian ke dalam sistem yang telah dikembangkan. Implementasi mencakup alur kerja sistem yang dirancang dalam penelitian. Berikut ini adalah alur implementasi sistem dalam proses klasifikasi emosi dengan menggunakan model IndoBERT:

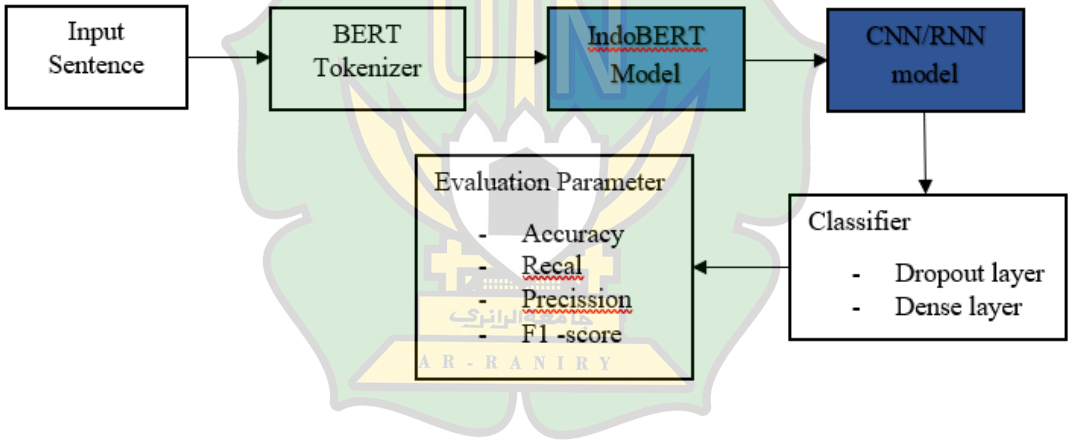


Gambar 3.2 Alur Implementasi Sistem

Rincian alur implementasi sistme adalah dataset yang digunakan dalam penelitian ini merupakan data ulasan konsumen yang berbahasa Indonesia terhadap aplikasi DANA. Selanjutnya dilakukan tahap *pre-processing* sebagai tahap membersihkan teks pada data lalu diberi label pada setiap teks.

Selanjutnya, dilakukan pembagian atau splitting data dengan proporsi 80% untuk data latih, 10% untuk data validasi, dan 10% untuk data testing. Data latih digunakan untuk melatih model IndoBERT agar mampu mengenali pola dalam data. Data validasi berfungsi untuk mengevaluasi kinerja model yang telah dilatih, sehingga dapat memantau sejauh mana model mampu memahami pola data. Sementara itu, data testing digunakan untuk mengukur performa akhir model.

3.5.2 Model Hybrid



Gambar 3.3 Model *Hybrid* IndoBERT dengan CNN/RNN

Model *hybrid* antara IndoBERT dan *Convolutional Neural Network* (CNN) menggabungkan kekuatan kedua arsitektur untuk meningkatkan performa dalam tugas-tugas pemrosesan Bahasa alami (NLP). IndoBERT, sebagai varian BERT yang dioptimalkan untuk bahasa Indonesia, sangat baik dalam memahami konteks dan makna kata dalam teks. Di sisi lain, CNN terkenal karena kemampuannya dalam mengekstraksi fitur-fitur penting dari data spasial atau sekuensial. Dalam model *hybrid* ini, IndoBERT digunakan untuk menangkap konteks semantic dan informasi bahasa

yang mendalam, Sedangkan CNN diterapkan untuk menangani dan menganalisis fitur lokal dalam teks, seperti pola atau struktur yang lebih halus. Dengan mengintegrasikan keduanya, model ini diharapkan dapat memanfaatkan keunggulan masing-masing arsitektur untuk memberikan hasil yang lebih akurat dan efektif dalam berbagai aplikasi NLP. Hal yang sama juga dilakukan untuk model IndoBERT dengan RNN.

3.6 Evaluasi Model

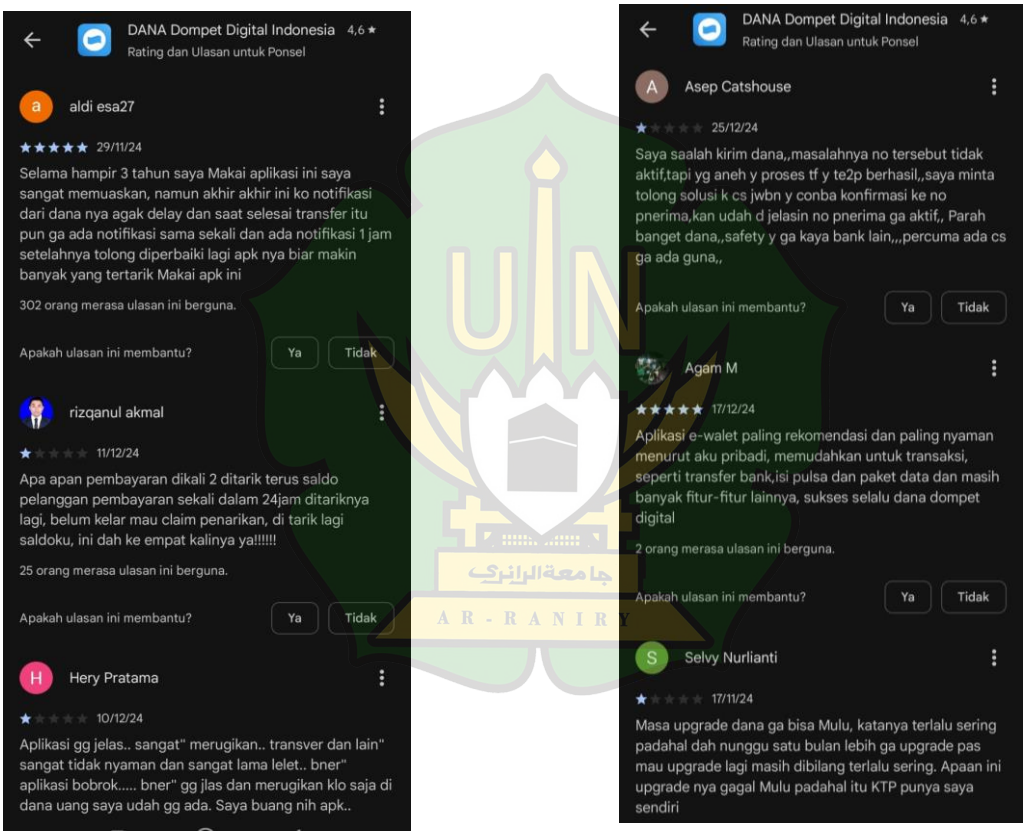
Evaluasi dilakukan untuk mengukur kinerja model yang dihasilkan dari proses *fine-tuning*. Evaluasi dilakukan dengan menggunakan *confusion matrix*. Metode *confusion matrix* ini terdiri dari 4 jenis nilai yaitu TP (*True Positive*), FP (*False Positive*), FN (*False Negative*), dan TN (*True Negative*). Nilai-nilai tersebut kemudian digunakan untuk menganalisis hasil kerja model dengan menghitung nilai *accuracy*, *precision*, *recall*, dan *f1- score*

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Data set yang digunakan dalam penelitian ini merupakan data ulasan konsumen yang berbahasa Indonesia terhadap aplikasi DANA di *website Google Play Store* sebanyak 2000 komentar yang disimpan dalam format csv. Berikut contoh salah satu komentar pada aplikasi DANA tersebut seperti pada gambar 4.1.



Gambar 4.1 Komentar Netizen Terhadap Aplikasi DANA

Pada pengumpulan data menggunakan *library google play scrapper* dengan atribut *rating* dan *comment.* Untuk mempermudah proses pengolahan data, dataset disimpan dalam format *file csv.* Untuk memaksimalkan pemanfaatan data,

maka langkah yang di ambil adalah mengurangi atribut dengan menghapus *rating* karena tidak relevan dalam klasifikasi emosi. Gambar 4.2 menunjukkan contoh hasil pengumpulan data.

Gambar 4.2 Hasil Pengumpulan Data

4.2 Pelabelan Data

Penelitian ini menggunakan lima jenis label emosi yaitu senang, puas, netral, kecewa dan marah. Berdasarkan 2000 komentar data terdapat 794 data senang, 171 data puas, 144 netral, 397 data kecewa, dan 493 data marah.

4.3 *Pre-Processing*

Tahap berikutnya adalah *pre-processing*. *Pre-processing* merupakan tahapan untuk membersihkan dataset yang akan digunakan sebagai bahan uji, yang bertujuan agar dataset dapat dibaca dengan baik oleh model sehingga akan memberikan hasil yang akurat.

Dataset yang diambil melalui beberapa tahap *pre-processing* antara lain sebagai berikut:

a. *Case folding*

Merupakan tahapan mengubah semua huruf dalam teks menjadi huruf kecil yang bertujuan untuk menghindari perbedaan makna yang disebabkan oleh variasi

penulisan huruf. Kode pemograman untuk proses *case folding* dapat dilihat pada gambar 4.3.

```
df['comment'] = df['comment'].apply(lambda x: " ".join(word.lower() for word in x.split()))
df.head()
```

	comment	label
0	baik dan mudah digunakan	senang
1	bagus sesuai kebutuhan,terima kasih.	senang
2	top lah pakai apk dana	senang
3	kak dana saya terproteksi sudah 3 hari tolong di bantu	marah
4	aplikasi e-wallet paling bagus se indonesia. mau top up banyak pilihan dan gampang aksesnya. saldo dana juga bisa di tarik uang tunai juga. transfer semua bank bisa. bayar tagihan per bulan, seperti listrik, pdam, dan lain-lain bisa. bahkan sudah bisa bayar untuk negara, seperti: pbb, e-tilang, dan lain-lain.	senang

Gambar 4.3 Kode Pemrograman *Case Folding*

b. *Filtering*

Merupakan teknik yang bertujuan untuk menghapus kata-kata yang tidak informatif atau tidak relevan dalam sebuah teks misalnya kata depan, kata ganti, atau kata sambung. Teknik ini bertujuan untuk mengurangi ukuran data teks sehingga dapat meningkatkan fokus dalam analisis kata yang penting. Kode pemrograman yang digunakan dalam proses *filtering* dapat dilihat pada gambar 4.4.

```
def clean_review(text):
    # Ubah enter menjadi spasi
    text = re.sub(r'\n', ' ', text)
    # Hapus emoji
    text = emoji.demojize(text)
    text = re.sub(r':[a-zA-Z_]+:', ' ', text)  # Hapus nama emoji yang diconvert
    # Hapus emoticon
    text = re.sub(r"([xX;:]'?[dDpPvVoO3\)])", ' ', text)
    # Hapus link
    text = re.sub(r"https?:\/\/[^\s]+", "", text)
    # Hapus username
    text = re.sub(r"@[^\s]+[\s]?", ' ', text)
    # Hapus hashtag
    text = re.sub(r'#(\S+)', r'\1', text)
    # Hapus angka dan beberapa simbol
    text = re.sub(r'[^a-zA-Z,.?!~]+', ' ', text)
    # Hapus karakter berulang
    text = repeatcharClean(text)
    # Bersihkan spasi berlebih
    text = re.sub(r'[ ]+', ' ', text)
    return text
df.head(10)
```

Gambar 4.4 Kode Pemrograman *Filtering*

c. Normalisasi Teks

Merupakan melakukan normalisasi teks untuk mengubah teks menjadi format yang konsisten seperti menghilangkan tanda baca atau karakter khusus, mengubah bahasa slang, dan mengganti bentuk singkatan atau akronim dengan bentuk lengkapnya. Kode pemrograman yang digunakan dalam proses normalisasi teks dapat dilihat pada gambar 4.5.

```
character = ['.',',',';',':','-','...','?','!','(',')','[',']', '{','}','<',
             '>','"','/','\'','#','-','@','a', 'b', 'c', 'd','e', 'f','g', 'h', 'i',
             'j', 'k', 'l', 'm', 'n', 'o', 'p','q', 'r', 's', 't', 'u', 'v', 'w', 'x','y',
             'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I','J', 'K', 'L', 'M', 'N', '0',
             'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'w', 'x', 'Y', 'z']

# hapus karakter yang berulang
def repeatcharClean(text):
  for i in range(len(character)):
    charac_long = 5
    while charac_long > 2:
      char = character[i]*charac_long
      text = text.replace(char,character[i])
      charac_long -= 1
  return text
df.head()
```

Gambar 4.5 Kode Pemrograman Normalisasi Teks

	A
1	comment
2	mantap, minus nya cuma ga bisa jadi metode pembayaran di apk lain
3	Mau melakukan penarikan dana bisnis, tapi gak bisa pindai wajah, katanya wajah berbeda, padahal aku akun bikin sendiri pakai wajah sendiri, ko bisa kata
4	sangat membantu
5	Harusnya aplikasi ini ditutup saja. Banyak disalahgunakan jadi fasilitas untuk topup judi online bahaya sekali
6	aplikasi yg bagus dan berguna
7	Gk bisa daptar aplikasi dana
8	Sudah saya coba semua ya saran nya percuma tetap tidak bisa duit saya nyangkut di sana bagaimana ini penipuan Kntl
9	Kok dana cicil saya belum tersedia ya
10	dana cili blum muncul juga
11	Aku Udah Biasa Pake Yang Ini
12	lebih mudah untuk transaksi
13	Bagaimana Mau Transfer Antar Bank, Harus Upgrade. Di Upgrade Harus Pakai E-KTP. Pakai E-KTP Enggak Bisa. Enggak Bisa Lagi Enggak Bisa. Enggak Bisa La
14	Mudah digunakan untuk swmua transaksi
15	Nyata saya sudah chat diana hasil nol,,saya jelasan kronologi kalau akun saya kebobolan,,saya tidak melakukan transaksi pembayaran Qris sama pembaya
16	lumayan
17	Payah kenisi malah makin parah masa iya TF ke reqening berhasil tapi saldo ngga masuk ke reqening gimana tuh penjelasannya saldo nya kemana @dana

Gambar 4.6 Teks setelah melewati Pre-Processing

4.4 Split Data

Dataset yang telah melalui tahap *preprocessing* akan dibagi menjadi tiga bagian, yaitu data training, data valid, dan data testing. Data training merupakan data

yang digunakan untuk melatih model IndoBERT agar mampu mengenali pola dalam data. Data valid berfungsi untuk mengevaluasi kinerja model yang telah dilatih, sehingga dapat memantau sejauh mana model mampu memahami pola data. Sementara itu, data testing digunakan untuk mengukur performa akhir model.

Penelitian ini membagi dataset menjadi 80% data training, 10% data valid, dan 10% data testing. Jumlah setiap data yang telah dibagi dapat dilihat pada tabel 4.1.

Tabel 4.1 Jumlah Hasil Dataset

Dataset	Jumlah Dataset
Dataset Train	1600
Dataset Validation	200
Dataset Test	200

4.5 Perancangan Model IndoBERT

Tahap berikutnya yang dilakukan adalah memasukkan hasil yang telah didapatkan pada tahap sebelumnya ke jaringan IndoBERT. Penelitian ini menggunakan perancangan model IndoBERT base p1. Kode untuk penerapan model dapat dilihat pada gambar 4.7.

```
# Tokenizer IndoBERT
tokenizer = BertTokenizer.from_pretrained('indobenchmark/indobert-base-p1')
```

```
# ==== 2. Model IndoBERT ====
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = BertForSequenceClassification.from_pretrained('indobenchmark/indobert-base-p1', num_labels=5)
model = model.to(device)
```

Gambar 4.7 Kode Penerapan model IndoBERT Base P1

Kode ini digunakan untuk memuat tokenizer yang sesuai dengan model IndoBERT-base-p1, yaitu model BERT yang dirancang untuk bahasa Indonesia. Fungsi utama tokenizer adalah mengonversi teks menjadi token yang dapat dimengerti

oleh model, termasuk memisahkan kata atau frasa, menambahkan token spesial seperti [CLS] dan [SEP], serta mengubah token menjadi angka. Langkah ini penting agar data teks dapat diproses secara optimal oleh model IndoBERT.

Selanjutnya, memilih perangkat eksekusi dan mengonfigurasi model IndoBERT-base-p1 untuk klasifikasi teks. Perangkat yang digunakan dipilih secara otomatis berdasarkan ketersediaan GPU; jika tidak ada GPU, model akan dijalankan di CPU. Kemudian, model *BERT for Sequence Classification* dimuat menggunakan model dasar IndoBERT-base-p1 dengan konfigurasi untuk klasifikasi lima label yaitu senang, puas, netral, kecewa, dan marah. Model tersebut kemudian dipindahkan ke perangkat yang dipilih untuk memastikan proses pelatihan atau inferensi dapat berjalan dengan optimal.

4.6 Perancangan Model IndoBERT dan Model CNN

Pada perancangan model hybrid yang menggabungkan IndoBERT base p2 dengan jaringan saraf konvolusi (CNN) adalah pendekatan yang dirancang untuk meningkatkan performa tugas klasifikasi texs, seperti analisis emosi. Model ini memanfaatkan kemampuan BERT dalam memahami konteks semantic secara global dan CNN yang unggul dalam mengenali pola local pada data. Kode untuk penerapan model tersebut dapat dilihat pada gambar 4.8.

```
# Pastikan Anda memiliki IndoBERT pretrained model
MODEL_NAME = "indobenchmark/indobert-base-p2"
tokenizer = BertTokenizer.from_pretrained(MODEL_NAME)
bert_model = TFBertModel.from_pretrained(MODEL_NAME)
```

```
# Model definition
input_ids = tf.keras.layers.Input(shape=(max_len,), dtype=tf.int32, name="input_ids")
attention_mask = tf.keras.layers.Input(shape=(max_len,), dtype=tf.int32, name="attention_mask")

bert_output = bert_model(
    input_ids,
    attention_mask=attention_mask
)
```

```
# Combine BERT with CNN
sequence_output = bert_output.last_hidden_state
conv1 = tf.keras.layers.Conv1D(128, kernel_size=3, activation='relu')(sequence_output)
conv1 = tf.keras.layers.GlobalMaxPooling1D()(conv1)

output = tf.keras.layers.Dense(num_classes, activation='softmax')(conv1)
model = tf.keras.models.Model(inputs=[input_ids, attention_mask], outputs=output)

# Compile model
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=5e-5),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Gambar 4.8 Kode Penerapan model IndoBERT dengan CNN

Kode di atas mendefinisikan arsitektur model berbasis IndoBERT yang dikombinasikan dengan lapisan CNN untuk tugas klasifikasi emosi. Model menggunakan tokenizer dan arsitektur IndoBERT dari model pra-terlatih indobenchmark/indobert-base-p2, yang diimpor menggunakan *BertTokenizer* dan TFBertModel. Dua input utama model adalah input_ids dan *attention_mask*, yang masing-masing merepresentasikan token ID dan mask perhatian untuk teks yang diproses. Kedua input ini memiliki dimensi *max_len.*

Keluaran dari IndoBERT, yaitu *last_hidden_state,* digunakan sebagai representasi fitur sekuensial. Kemudian, representasi ini diproses lebih lanjut menggunakan lapisan *Conv1D* dengan 128 filter dan ukuran kernel 3 untuk menangkap pola lokal dalam data teks. Hasil dari lapisan konvolusi ini diringkas menggunakan lapisan *GlobalMaxPooling1D,* yang mengambil nilai maksimum di setiap fitur untuk menghasilkan vektor fitur tetap.

Vektor fitur yang dihasilkan dari pooling kemudian dimasukkan ke dalam lapisan *Dense* dengan aktivasi softmax, yang digunakan untuk memprediksi

probabilitas kelas dari sejumlah label yang telah ditentukan (num_classes). Model ini dikompilasi dengan pengoptimal *Adam* dengan laju pembelajaran sebesar $5e^{-5}$, menggunakan fungsi kerugian *sparse_categorical_crossentropy* untuk klasifikasi multi-kelas, dan metrik evaluasi berupa akurasi.

Singkatnya, model ini memadukan kekuatan pemahaman teks mendalam dari IndoBERT dengan kemampuan CNN dalam menangkap pola lokal, sehingga dirancang untuk memberikan performa yang baik pada tugas klasifikasi emosi berbasis teks.

4.7 Perancangan Model IndoBERT dan Model RNN

Pada perancangan model hybrid yang menggabungkan IndoBERT Base dengan jaringan saraf berulang (*Recurrent Neural Network*/RNN) adalah pendekatan inovatif dalam pemrosesan bahasa alami (NLP). Model ini memanfaatkan kemampuan BERT untuk memahami konteks semantik secara mendalam, yang kemudian diperkuat dengan RNN untuk menangkap pola sekuensial dalam data teks. Kombinasi ini sangat efektif untuk tugas seperti klasifikasi teks berbasis emosi, di mana hubungan antar kata dalam urutan teks memainkan peran penting. Kode untuk penerapan model tersebut dapat dilihat pada gambar 4.9.

```
pretrained_model_name = 'indobenchmark/indobert-base-p2'
tokenizer = BertTokenizer.from_pretrained(pretrained_model_name)
```

```
# Step 3: Build Hybrid Model
class BertRNNModel(tf.keras.Model):
    def __init__(self, bert_model, rnn_units, num_classes):
        super(BertRNNModel, self).__init__()
        self.bert = bert_model
        self.lstm = tf.keras.layers.LSTM(rnn_units, return_sequences=False)
        self.dropout = tf.keras.layers.Dropout(0.3)
        self.dense = tf.keras.layers.Dense(num_classes, activation='softmax')

    def call(self, inputs):
        # Ambil hidden states terakhir dari BERT
        bert_output = self.bert(inputs)[0]  # Shape: (batch_size, sequence_length, hidden_size)
        lstm_output = self.lstm(bert_output)  # LSTM mengolah sequence
        dropout_output = self.dropout(lstm_output)
        return self.dense(dropout_output)
```

```
# Load pretrained IndoBERT
bert_model = TFBertModel.from_pretrained(pretrained_model_name)
```

```
# Define the hybrid model
rnn_units = 128
model = BertRNNModel(bert_model, rnn_units, num_labels)
```

```
# Compile the model
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=5e-5),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Gambar 4.9 Kode Penerapan Model IndoBERT dan Model RNN

Langkah awal adalah memuat tokenizer dan model dasar BERT. Tokenizer bertugas mengonversi teks mentah menjadi token yang dapat dipahami oleh model BERT, sementara model pra-terlatih IndoBERT digunakan untuk menghasilkan representasi kontekstual dari masukan teks. Selanjutnya, arsitektur hybrid dibangun dengan menambahkan lapisan RNN ke keluaran BERT. RNN yang digunakan memiliki 128 unit untuk memproses informasi sekuensial, sehingga memungkinkan model untuk memahami hubungan antar token yang mungkin tidak sepenuhnya ditangkap oleh BERT.

Setelah model hybrid dirancang, tahap berikutnya adalah kompilasi model. Optimizer Adam digunakan dengan *learning rate* $5e^{-5}$ untuk memastikan konvergensi yang stabil selama pelatihan. Fungsi kerugian categorical_crossentropy diterapkan karena model menangani tugas klasifikasi multi-kelas, sementara metrik akurasi digunakan untuk mengevaluasi performa model.

Pendekatan ini menggabungkan keunggulan BERT dalam memahami konteks global dengan kemampuan RNN dalam mengolah urutan teks secara mendalam. Dengan arsitektur ini, model dapat lebih efektif dalam menyelesaikan tugas-tugas yang memerlukan analisis mendalam terhadap urutan kata, seperti klasifikasi emosi atau analisis sentimen, dibandingkan dengan penggunaan BERT atau RNN secara terpisah.

4.8 Hyperparameter

Untuk seluruh model dalam penelitian ini menggunakan *hyperparameter* yang sama, *hyperparameter* yang digunakan yaitu menggunakan *batch size* 32, *learning rate* $5e^{-5}$, dan dengan epoch 10. Untuk lebih jelasnya dapat dilihat pada tabel 4.2.

Tabel 4.2 Hyperparameter

Hyperparameter	**Skala**
Batch size	32
Learning rate	$5e^{-5}$
Epoch	10

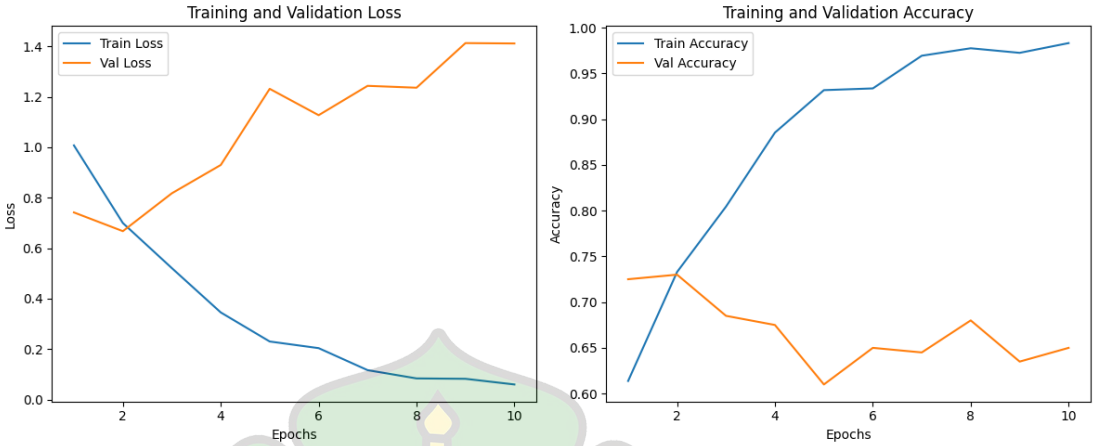
Dari tabel hyperparameter di atas menjelaskan pengaturan utama yang digunakan dalam pelatihan model. *Batch size* diatur sebesar 32, yang berarti model akan memproses 32 sampel data sekaligus dalam satu iterasi pelatihan. Hal ini bertujuan untuk mencapai keseimbangan antara kecepatan pelatihan dan konsumsi memori. Selanjutnya, *learning rate* ditetapkan sebesar 2×10^{-5} menunjukkan seberapa besar langkah pembaruan bobot model dilakukan selama pelatihan. Nilai *learning rate* ini cukup kecil untuk memastikan model belajar secara bertahap dan menghindari perubahan parameter yang terlalu drastis. Terakhir, pelatihan model dilakukan selama 10 *epoch*, yang berarti seluruh data pelatihan akan dilalui sebanyak sepuluh kali untuk memastikan model memiliki cukup waktu untuk belajar pola-pola dari data. Pengaturan ini bertujuan untuk mengoptimalkan kinerja model tanpa menyebabkan *overfitting.*

4.9 Hasil Pelatihan

Dalam melakukan klasifikasi emosi pada teks dalam penelitian ini, semua model yang digunakan membagi dataset yang sama menjadi tiga bagian yaitu data train, data valid, dan data test. Dengan porsi 80% untuk data train, 10% untuk data valid, dan 10% untuk data test.

4.9.1 Hasil Train Validation Accuracy dan Loss Model IndoBERT

Hasil train validation accuracy dan loss model IndoBERT dengan menggunakan *epoch* 10 pada dataset aplikasi dana dapat dilihat pada gambar 4.10.



Gambar 4.10 Grafik Akurasi dan Loss Model IndoBERT

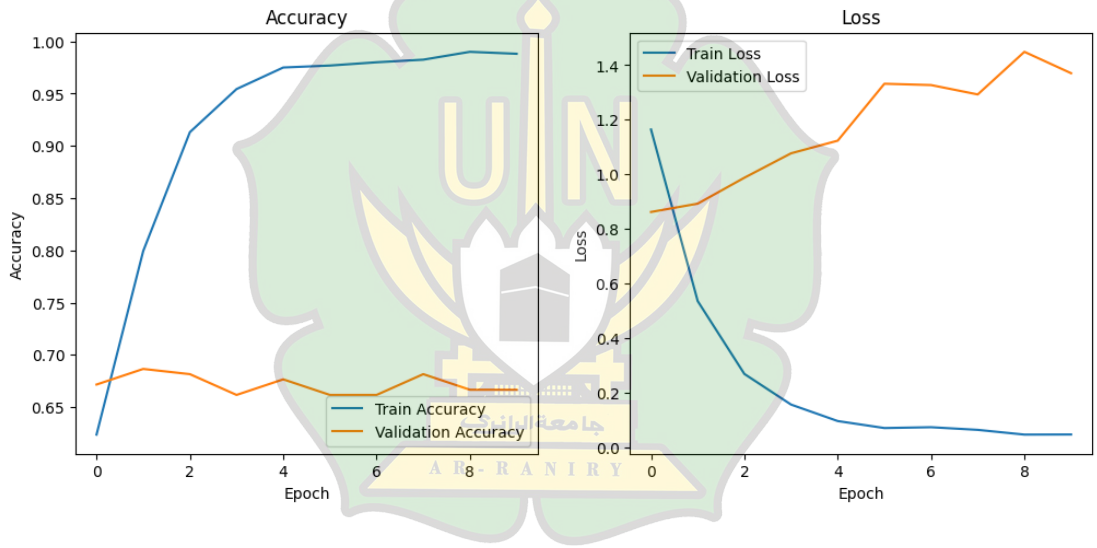
Grafik di atas menunjukkan tren akurasi dan *loss* selama pelatihan dan validasi untuk model yang diuji dalam 10 *epoch*. Pada grafik *training and validation loss* (kiri), terlihat bahwa *training loss* terus menurun secara konsisten hingga mencapai nilai mendekati nol, yang menunjukkan bahwa model mampu belajar dari data pelatihan dengan baik. Namun, *validation loss* justru meningkat setelah beberapa *epoch* awal, menandakan bahwa model mengalami *overfitting*. Model lebih fokus pada data pelatihan tetapi kehilangan kemampuan generalisasi pada data validasi.

Pada grafik *training and validation accuracy* (kanan), akurasi pelatihan (*train accuracy*) meningkat pesat hingga mendekati 100% yaitu 98% pada epoch terakhir, yang menguatkan indikasi bahwa model sangat cocok terhadap data pelatihan. Sebaliknya, akurasi validasi (*validation accuracy*) cenderung stagnan dan bahkan menurun di pertengahan pelatihan sebelum sedikit meningkat di akhir. Hal ini menunjukkan bahwa meskipun model berhasil mengenali pola dalam data pelatihan, performanya kurang stabil ketika diaplikasikan pada data validasi.

Secara keseluruhan, grafik ini mengindikasikan bahwa model mengalami *overfitting*. Oleh karena itu, diperlukan langkah-langkah seperti penggunaan teknik regulasi (*regularization*), *dropout*, atau penambahan data untuk meningkatkan kemampuan generalisasi model terhadap data validasi.

4.9.2 Hasil Train Validation Accuracy dan Loss Model IndoBERT dengan Model CNN

Penelitian ini menggabungkan antara model IndoBERT dengan model CNN. Hasil train validation accuracy dan loss model *hybrid* tersebut dengan menggunakan *epoch* 10 pada dataset aplikasi dana, yang dimana dapat dilihat pada tabel 4.2 dan tampilan dalam bentuk grafik pada gambar 4.11.



Gambar 4.11 Grafik *Accuracy* dan Loss Model IndoBERT dengan Model CNN

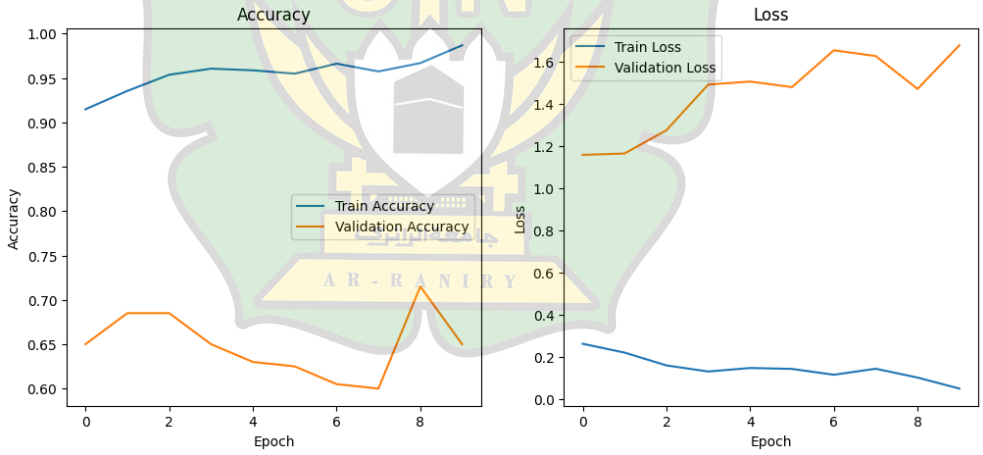
Diketahui bahwa grafik di atas menunjukkan tren akurasi dan loss pada model selama proses pelatihan dan validasi selama beberapa *epoch.* Pada grafik akurasi, akurasi pelatihan meningkat tajam pada awal epoch dan mencapai hampir 100% sekitar epoch ke-4 hingga ke-5. Setelah itu, akurasi pelatihan tetap stabil mendekati 100%. Namun, akurasi validasi justru tetap stagnan di sekitar angka 65%-70% sepanjang proses pelatihan, tanpa peningkatan yang signifikan.

Di sisi lain, grafik loss menunjukkan bahwa loss pelatihan berkurang secara konsisten seiring bertambahnya epoch, menandakan bahwa model berhasil mempelajari pola pada data pelatihan. Namun, loss validasi justru menunjukkan tren peningkatan setelah beberapa epoch awal, yang menjadi indikasi bahwa model mengalami *overfitting*.

Secara keseluruhan, grafik ini menunjukkan bahwa meskipun model sangat akurat pada data pelatihan, performanya pada data validasi kurang optimal.

4.9.3 Hasil Train-validation Accuracy dan Loss Model IndoBERT dengan Model RNN

Penelitian ini menggabungkan antara model IndoBERT dengan model RNN. Hasil train validation accuracy dan loss model hybrid tersebut dengan menggunakan epoch 10 pada dataset aplikasi dana. Hasil tersebut dapat dilihat pada tampilan gambar 4.12.



Gambar 4.12 Grafik *Accuracy* dan *Loss* Model IndoBERT dengan Model RNN

Gambar grafik di atas menunjukkan perubahan akurasi dan loss untuk data pelatihan dan validasi selama beberapa *epoch*. Pada grafik akurasi, terlihat bahwa akurasi pelatihan secara konsisten meningkat hingga mendekati 100% yaitu 98%, menunjukkan bahwa model mampu mempelajari pola dalam data pelatihan dengan sangat baik. Sebaliknya, akurasi validasi cenderung berfluktuasi dan tidak

menunjukkan tren peningkatan yang konsisten. Akurasi validasi berada di sekitar 60%-70%, dengan penurunan signifikan pada beberapa epoch, yang mengindikasikan bahwa model tidak mampu mempertahankan performa yang baik pada data validasi.

Sementara itu, grafik loss memperlihatkan penurunan yang stabil pada loss pelatihan, yang menunjukkan bahwa model terus meminimalkan kesalahan selama pelatihan. Namun, loss validasi justru meningkat seiring bertambahnya epoch, menunjukkan tanda-tanda *overfitting*. Peningkatan loss validasi ini mengindikasikan bahwa model lebih fokus pada data pelatihan dan kehilangan kemampuan untuk menggeneralisasi pada data baru.

Secara keseluruhan, grafik ini menunjukkan bahwa model mengalami overfitting. Meskipun performa pada data pelatihan sangat baik, hasil pada data validasi menunjukkan bahwa model tidak generalisasi dengan baik. Berikut merupakan kondisi tambahan yang biasa terjadi selama proses pelatihan.

1. Overfitting

   Overfitting adalah situasi di mana sebuah model machine learning terlalu terfokus pada data pelatihan, sehingga mampu memberikan hasil yang sangat baik pada data tersebut, namun memiliki performa yang buruk ketika diterapkan pada data baru atau data uji. Hal ini disebabkan oleh model yang mempelajari detail serta noise dari data pelatihan, termasuk pola-pola yang sebenarnya tidak signifikan, sehingga mengurangi kemampuannya untuk melakukan generalisasi dengan baik

2. Underfitting

   Underfittingterjadi ketika model machine learning tidak mampu mengenali pola atau hubungan penting dalam data pelatihan. Hal ini menyebabkan kinerja model menjadi rendah, baik pada data pelatihan maupun data uji. Penyebabnya biasanya adalah model yang terlalu sederhana atau kurang kompleks sehingga tidak dapat mempelajari data dengan maksimal.

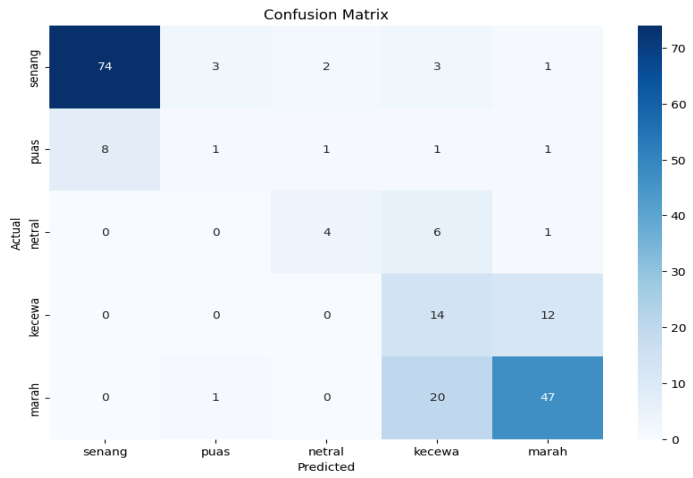
3. Goodfitting menggambarkan keadaan optimal dalam machine learning, di mana model berhasil mengenali pola signifikan dari data pelatihan tanpa terpengaruh oleh noise. Model dengan karakteristik ini mampu menghasilkan performa yang stabil dan akurat pada data pelatihan serta data baru (data uji). Hal ini menunjukkan bahwa model dapat belajar secara efektif dan seimbang antara bias dan variansi.

4.10 Hasil Confussion Matrix dan Evaluasi Model

Setelah melewati tahap *fine-tuning* dari setiap model yang telah di uji, selanjutnya akan dilakukan perbandingan terhadap performa setiap model yang telah di latih. Perbandingan dilakukan antara model IndoBERT dengan model *hybrid*. Evaluasi model akan dinilai dengan menghitung nilai *accuracy*, *precision*, *recall*, dan *f1- score*. Confusion matrix ini merupakan sebuah tabel yang digunakan untuk mengevaluasi kinerja suatu model klasifikasi, dengan membandingkan hasil prediksi model terhadap data yang sebenarnya. Hasil dari confusion matrix pada setiap model dapat dilihat sebagai berikut ini.

4.10.1 Confusion Matrix Model IndoBERT

Untuk melihat hasil confusion matrix dari model IndoBERT dapat di lihat pada gambar 4.13.



Gambar 4.13 Confusion Matrix Model IndoBERT

Pada gambar *confusion matrix* di atas menggambarkan performa model dalam memprediksi lima kategori emosi. Baris pada matriks menunjukkan label sebenarnya, sedangkan kolom merepresentasikan prediksi model. Secara keseluruhan, model mampu mengenali kelas "senang" dengan sangat baik, menghasilkan 74 prediksi yang benar dari total 83 sampel dalam kategori ini, meskipun masih terdapat kesalahan prediksi pada beberapa sampel yang diklasifikasikan sebagai kelas lain.

Kinerja pada kelas "puas" cukup lemah. Dari 12 sampel, hanya 1 yang berhasil diklasifikasikan dengan benar, sedangkan sisanya salah dikategorikan ke dalam kelas lain, seperti "senang" atau "kecewa." Hal ini menunjukkan bahwa model kesulitan dalam mengidentifikasi emosi "puas," kemungkinan karena keterbatasan jumlah data atau tumpang tindih karakteristiknya dengan kelas lain.

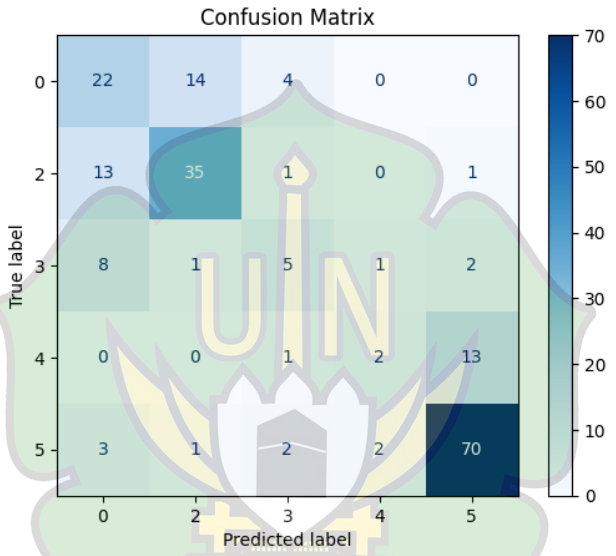
Pada kelas "kecewa," model memprediksi 14 dari 26 sampel dengan benar, tetapi masih terjadi kesalahan yang signifikan, terutama ketika emosi ini dikategorikan sebagai "marah." Hal serupa terjadi pada kelas "marah," di mana dari 68 sampel, 47 di antaranya diklasifikasikan dengan benar, tetapi 20 lainnya salah dikategorikan sebagai "kecewa."

Secara keseluruhan, hasil ini menunjukkan bahwa model lebih baik dalam mengenali kelas mayoritas seperti "senang" dan "marah," namun kesulitan dalam

memprediksi kelas minoritas, sepertipuas dan netral. Hal ini mengindikasikan perlunya peningkatan pada representasi data dan teknik pelatihan agar akurasi model dapat lebih merata di semua kategori emosi.

4.10.2 Confusion Matrix Model IndoBERT dengan Model CNN

Untuk melihat hasil confusion matrix dari model IndoBERT dengan model CNN terhadap dataset aplikasi dana dapat dilihat pada gambar 4.14.

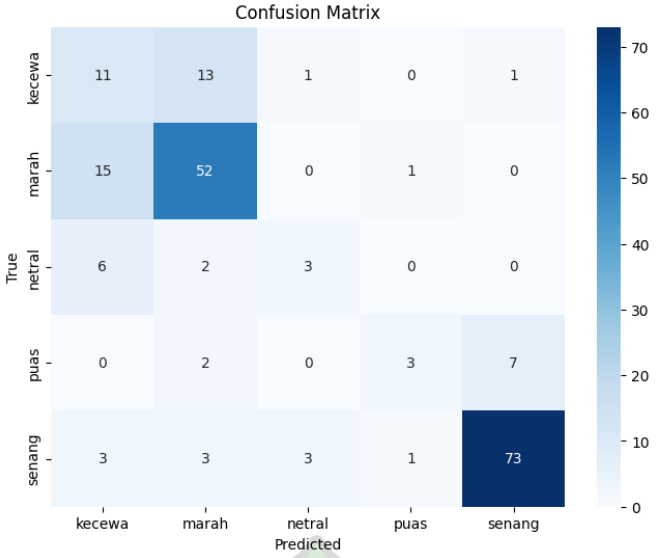


Gambar 4.14 Confusion Matrix Model IndoBERT dengan Model CNN

Pada gambar di atas grafik ini Secara keseluruhan, model lebih akurat dalam mengidentifikasi label senang dibandingkan label lainnya, sedangkan label seperti puas dan netral tampaknya menjadi tantangan bagi model untuk diklasifikasikan dengan benar. Hal ini menunjukkan bahwa ada ruang untuk perbaikan, terutama dalam membedakan label yang mungkin memiliki pola atau emosi serupa.

4.10.3 Confusion Matrix Model IndoBERT dengan Model RNN

Untuk melihat hasil confusion matrix dari model IndoBERT dengan model RNN terhadap dataset aplikasi dana dapat dilihat pada gambar 4.15.



Gambar 4.15 Confusion Matrix Model IndoBERT dengan Model RNN

Pada Confussion Matrix ini kategori untuk kelas *kecewa*, precision sebesar 0.31 menunjukkan bahwa hanya 31% dari prediksi kelas ini yang benar, sementara *recall* 0.42 mengindikasikan bahwa model berhasil mengidentifikasi 42% dari seluruh sampel sebenarnya di kelas ini. *F1-score* 0.36 mengindikasikan keseimbangan antara *precision* dan *recall*.

Kelas *marah* memiliki kinerja yang cukup baik dengan precision 0.72 dan *recall* 0.76, menghasilkan *f1-score* 0.74, yang menunjukkan performa prediksi yang konsisten. Kelas *netral* memiliki precision 0.43 dan *recall* 0.27, menghasilkan *f1-score* 0.33, yang menunjukkan bahwa kelas ini cukup sulit untuk diklasifikasikan dengan baik.

Kinerja kelas *puas* cukup rendah dengan *recall* 0.25, sehingga hanya 25% sampel yang benar-benar terdeteksi, meskipun *precision* nya adalah 0.60. Akhirnya, kelas *senang* menunjukkan performa terbaik dengan *precision* 0.90, *recall* 0.88, dan *f1-score* 0.89, yang mencerminkan kemampuan model yang sangat baik untuk mengenali kelas ini.

4.11.1 Hasil Pengujian Model IndoBERT

Pada tahap pertama ini dilakukan evaluasi performa model terhadap dataset yang telah diberikan yaitu dataset aplikasi dana dengan *epoch* 10. Hasil performa model dapat dilihat pada gambar 4.16.

	precision	recall	f1-score	support
senang	0.90	0.89	0.90	83
puas	0.20	0.08	0.12	12
netral	0.57	0.36	0.44	11
kecewa	0.32	0.54	0.40	26
marah	0.76	0.69	0.72	68
accuracy			0.70	200
macro avg	0.55	0.51	0.52	200
weighted avg	0.72	0.70	0.70	200

Gambar 4.16 Performa Model IndoBERT

Hasil performa model yang berdasarkan evaluasi model menunjukkan bahwa akurasi keseluruhan mencapai 70%, dengan nilai rata-rata *macro* untuk *precision*, *recall*, dan *f1-score* masing-masing sebesar 55%, 51%, dan 52%. Sementara itu, nilai rata-rata berbobot (*weighted average*) untuk metrik yang sama masing-masing adalah 72%, 70%, dan 70%, yang menunjukkan bahwa kelas-kelas dengan jumlah data lebih besar, seperti "senang" dan "marah," memiliki pengaruh dominan terhadap keseluruhan hasil.

Dari setiap kelas, kelas "senang" memiliki performa terbaik, dengan *precision*, *recall*, dan *f1-score* yang tinggi, yaitu sekitar 90%. Hal ini menunjukkan bahwa model dapat mengidentifikasi emosi "senang" dengan baik. Kelas "marah" juga menunjukkan hasil yang memuaskan, dengan *f1-score* mencapai 72%, yang mencerminkan keseimbangan antara *precision* dan *recall*. Namun, performa pada kelas lainnya, seperti "puas," "netral," dan "kecewa," masih jauh dari memadai. Untuk mengubah hasil dari koma tersebut menjadi bentuk persen maka setiap hasil yang terdapat pada hasil evaluasi tersebut dikalikan dengan 100.

4.11.2 Hasil Pengujian Model IndoBERT dengan Model CNN

Sama seperti pada tahap sebelumnya, pada tahap kedua ini akan dilakukan evaluasi performa model dengan epoch 10, namun pada tahap ini hasil evaluasi berasal dari model *hybrid*, yaitu antara model indoBERT dengan model CNN. Berikut hasil performa model dengan epoch 10, hasil tersebut dapat dilihat pada gambar 4.17.

	precision	recall	f1-score	support
senang	0.81	0.90	0.85	78
puas	0.40	0.12	0.19	16
netral	0.38	0.29	0.33	17
kecewa	0.48	0.55	0.51	40
marah	0.69	0.70	0.69	50
accuracy			0.67	201
macro avg	0.55	0.51	0.52	201
weighted avg	0.65	0.67	0.65	201

Gambar 4.17 Performa Model IndoBERT dengan Model CNN

Dapat dilihat bahwa hasil evaluasi model menggunakan pendekatan hybrid antara model IndoBERT dengan model CNN menunjukkan tingkat akurasi keseluruhan sebesar 67%. Berdasarkan metrik evaluasi per kelas, model memiliki performa yang sangat baik dalam mendeteksi label *senang*, dengan nilai *precision* sebesar 0.81, *recall* 0.90, dan *f1-score* 0.85. Hal ini menunjukkan bahwa model mampu mengidentifikasi emosi *senang* dengan akurat dan konsisten. Untuk label *marah*, performa juga cukup baik, dengan *precision* 0.69, *recall* 0.70, dan *f1-score* 0.69.

Namun, performa model pada beberapa label lainnya, seperti *puas* dan *netral*, masih kurang memuaskan. Pada label *puas*, *precision* tercatat sebesar 0.40, tetapi *recall*-nya rendah di angka 0.12, menghasilkan *f1-score* sebesar 0.19. Ini menunjukkan bahwa model sering gagal mendeteksi emosi *puas* dalam data validasi. Serupa dengan itu, label *netral* juga menunjukkan kelemahan, dengan *f1-score* hanya sebesar 0.33.

Sementara itu, label *kecewa* menunjukkan performa moderat, dengan *precision* 0.48, *recall* 0.55, dan *f1-score* 0.51. Secara keseluruhan, nilai rata-rata tertimbang

(*weighted average*) dari *precision* dan *recall* masing-masing adalah 0.65 dan 0.67, yang mendukung akurasi keseluruhan model.

4.11.3 Hasil Pengujian Model IndoBERT dengan Model RNN

Masih sama seperti tahap sebelumnya, pada tahap ketiga ini juga dilakukan evaluasi performa model dengan *epoch* 10, namun pada tahap ini hasil evaluasi berasal dari model *hybrid*, yaitu antara model indoBERT dengan model RNN. Berikut hasil performa model dengan *epoch* 10, hasil tersebut dapat dilihat pada gambar 4.18.

	precision	recall	f1-score	support
kecewa	0.31	0.42	0.36	26
marah	0.72	0.76	0.74	68
netral	0.43	0.27	0.33	11
puas	0.60	0.25	0.35	12
senang	0.90	0.88	0.89	83
accuracy			0.71	200
macro avg	0.59	0.52	0.54	200
weighted avg	0.72	0.71	0.71	200

Gambar 4.18 Performa Model IndoBERT Dengan Model RNN

Dari gambar di atas dapat dilihat bahwa hasil evaluasi performa model *Hybrid* dengan RNN menunjukkan tingkat akurasi keseluruhan sebesar 71%, yang mencerminkan kemampuan model untuk mengklasifikasikan data secara cukup baik. Pada label *senang*, model mencapai performa yang sangat tinggi dengan *precision* sebesar 0.90, *recall* 0.88, dan *f1-score* 0.89, menunjukkan bahwa model dapat mengenali emosi ini secara akurat dan konsisten. Label *marah* juga menunjukkan hasil yang solid, dengan *precision* 0.72, *recall* 0.76, dan *f1-score* 0.74.

Sebaliknya, performa model pada label dengan jumlah data lebih kecil seperti *puas*, *kecewa*, dan *netral* relatif kurang memuaskan. Pada label *puas*, model mencatat *precision* 0.60, tetapi *recall*-nya hanya 0.25, menghasilkan *f1-score* sebesar 0.35, yang menunjukkan kesulitan model dalam mendeteksi data dengan label tersebut. Label *kecewa* memiliki *f1-score* sebesar 0.36, dengan *precision* dan *recall* masing-masing

sebesar 0.31 dan 0.42. Sementara itu, performa pada label *netral* cukup rendah, dengan *f1-score* sebesar 0.33.

Secara keseluruhan, rata-rata tertimbang (*weighted average*) dari *precision*, *recall*, dan *f1-score* masing-masing adalah 0.72, 0.71, dan 0.71. Hasil ini menunjukkan bahwa model memiliki kinerja yang baik secara umum, terutama pada label yang memiliki lebih banyak data seperti *senang* dan *marah*. Namun, performa yang kurang stabil pada label dengan data lebih sedikit seperti *puas* dan *netral* mengindikasikan perlunya perbaikan, misalnya dengan menambah jumlah data atau melakukan penyesuaian pada arsitektur model agar lebih sensitif terhadap kategori dengan representasi kecil.

Dari seluruh hasil akurasi yang di dapat, berikut perbandingan nilai *f1-score* dari setiap akurasi yang didapat pada setiap model yang digunakan, dapat dilihat dalam tabel 4.2.

Tabel 4.2 Hasil Akurasi Setiap Dataset Terhadap Model

No	Model	Nilai f1-score Dataset Aplikasi Dana
1	IndoBERT	70 %
2	IndoBERT dengan CNN	67%
3	IndoBERT dengan RNN	71%

Berdasarkan tabel di atas, performa setiap model dalam memproses dataset aplikasi DANA diukur menggunakan nilai *f1-score*. Model IndoBERT mencatat nilai *f1-score* sebesar 70%, menunjukkan kinerja yang stabil dalam menangani data tersebut. Model IndoBERT dengan model CNN sedikit lebih rendah, dengan *f1-score* sebesar 67%, yang mengindikasikan bahwa penambahan lapisan CNN tidak memberikan peningkatan performa secara signifikan. Sementara itu, model IndoBERT *dengan* RNN menunjukkan performa terbaik di antara ketiga pendekatan dengan nilai *f1-score* sebesar 71%. Hal ini menunjukkan bahwa integrasi RNN pada model IndoBERT lebih

efektif dalam menangkap pola sekuensial dari data, sehingga menghasilkan klasifikasi yang lebih akurat dibandingkan dengan model lainnya. Secara keseluruhan, meskipun selisih performa antar-model tidak terlalu besar, model IndoBERT dengan model RNN memberikan hasil yang paling menjanjikan untuk dataset ini.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada penelitian ini dilakukan klasifikasi emosi pada komentar aplikasi DANA dengan jumlah data sebanyak 2000 data yang terdiri dari label senang 794 data, puas 171 data, netral 144 data, kecewa 394 data, dan marah 493 data. Penelitian ini menggunakan model IndoBERT dan model *hybrid* yaitu model IndoBERT dengan model CNN dan model IndoBERT dengan model RNN. Adapun hasil yang diperoleh dari penelitian ini adalah:

1. Proses klasifikasi emosi komentar aplikasi DANA dengan model IndoBERT dan model *hybrid* yaitu model IndoBERT dengan model CNN dan IndoBERT dengan model RNN, semua model menggunakan jumlah parameter *learning rate* yang sama yaitu sebesar $5e^{-5}$ dan *epoch* 10 pada pelatihan.
2. Berdasarkan hasil pengujian dari implementasi model, model IndoBERT dan model *hybrid* yaitu model IndoBERT dengan model CNN dan model IndoBERT dengan model RNN dalam melakukan klasifikasi emosi, maka diperoleh kesimpulan sebagai berikut: Klasifikasi emosi menggunakan model IndoBERT menghasilkan akurasi sebesar 70%, model IndoBERT dengan model CNN memperoleh hasil akurasi 67 %, dan model IndoBERT dengan model RNN memperoleh nilai akurasi sebesar 71 %.
3. Dari hasil penelitian sebelumnya menunjukkan model *Hybrid* yaitu antara model IndoBERT dengan model RNN mendapatkan hasil terbaik dalam melakukan klasifikasi emosi dengan nilai akurasi yang di dapatkan sebesar 71 % dibandingkan dengan model IndoBERT dan model IndoBERT dengan model RNN.

5.2 Saran

1. Menggunakan jumlah data yang seimbang antara emosi senang, puas, netral, kecewa dan marah serta menggunakan metode pelabelan dataset lainnya untuk mengetahui pengaruh pada performa model.
2. Melakukan penelitian dengan memanfaatkan model IndoBERT lainnya, dan model *hybrid* lainnya seperti, untuk memajukan penelitian dalam pengolahan bahasa alami (NLP) berbahasa Indonesia.

DAFTAR PUSTAKA

Abrilia, N.D. (2020). Pengaruh Persepsi Kemudahan dan Fitur Layanan Terhadap Minat Menggunakan E-Wallet Pada Aplikasi Dana Di Surabaya. *Jurnal Pendidikan Tata Niaga (JPTN)*, 8(3), 1006–1012.

Andika, I. (2023). Analisis Sentimen Persepsi Konsumen Terhadap Kualitas Layanan Aplikasi BSI Mobile Menggunakan Model Indobert. Universitas Islam Negeri Ar-Raniry, Banda Aceh.

Aripin., Agastya, W., & haryanto, H. (2021). Ekstraksi Emosi Majemuk Kalimat Bahasa Indonesia Menggunakan *Convolutional Neural Network*. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*. 10(2).

Bagus, A.T., & Fudholi, D.H. (2021). Klasifikasi Emosi pada Teks dengan Menggunakan Metode Deep Learning. *Syntax Literate: Jurnal Ilmiah Indonesia*. 6(1).

Chandradev, V., Suarjaya, I.M.A.D., & Bayupati, I.P.A. (2023). Analisis Sentimen Review Hotel Menggunakan Metode Deep Learning BERT. *Urnal BUana Informatika*. 14(2).

Chatterjee, A., Narahari, K.N., Joshi, M., & Agrawal, P. (2019). Semeval-2019 Task 3: Emocontext Contextual Emotion Detection in Text. *Proceedings of the 13th International Workshop on Semantic Evaluation*, 39–48.

Chiorrini, A., Mircoli, A., Diamantini, C., & Potena, D. (2021). Emotion and Sentiment Analysis of Tweets Using BERT. *Workshop Proceedings of the EDBT/ICDT*.

Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Naacl-Hlt 2019*, *Mlm*, 4171–4186.

Gupta, N., 2021. A Pre-Trained Vs Fine-Tuning Methodology in Transfer Learning. *Journal of Physics: Conference Series*, 1947(1).

Huang, C., Trabelsi, A., & Zaïane, O. (2019). ANA at SemEval-2019 Task 3: Contextual Emotion detection in Conversations through hierarchical LSTMs and

BERT. *Proceedings of the 13th International Workshop on Semantic Evaluation.* 49–53. https://doi.org/10.18653/v1/s19-2006.

Idayanti, R., & Ulandari, P. (2023). Peran Aplikasi Dompet Digital Indonesia (DANA) dalam Memudahkan Masyarakat Melakukan Pembayaran Digital. *Islamic Banking and Finance Journal.* 3(2). Doi: 10.30863/ibf.v3i2.5438.

Imaduddin, H., A'la, F.Y. & Nugroho, y.s., (2023). Sentiment Analysis in Indonesian Healthcare Applications using IndoBERT Approach. *International Journal of Advanced Computer Science and Applications*. 14(8). 113–117.

Imron, S., Setiawan, E.I., & Santoso, J. (2023). Deteksi Aspek Review E-Commerce Menggunakan IndoBERT Embedding dan CNN. *Journal of Inteligent System and Computation*. 5(1). Doi: 10.52985/insyst.v5i1.267.

Jayadianti, H., Kaswidjanti, W., Utomo, A.T., Saifullah, S., Dwiyanto, F.A., & Drezewski, R. (2022). Sentiment Analysis of Indonesian Reviews Using Fine-Tuning IndoBERT and R-CNN. *ILKOM Jurnal Ilmiah.* 14(3). 348-354.

Kumala, I., & Mutia, I. (2020). Pemanfaatan Aplikasi Dompet Digital Terhadap Transaksi Retail Mahasiswa. *Seminar Nasional Riset Dan Inovasi Teknologi (Semnas Ristek)*, 4(1).

Nafisa, Z. (2024). Klasifikasi Emosi Tokoh dalam Novel The Coldest Boyfriend Karya Itsfiyawn: Kajian Psikologi Sastra David Krech serta Manfaatnya dalam Pembelajaran Sastra di SMA. *BAPALA.* 11(1). 49-61.

Nazar, R. (2024). Implementasi Pemrograman Python Menggunakan Google Colab. *Jurnal Infoormatika dan Komputer (JIK).* 15(1). 50-56.

Nongthombam, K., & Sharma, D. (2021). Data Analysis using Python. *International Journal of Engineering Reasearch & technology (IJERT).* 10(7).

Press Release. (2024). Capai pertumbuhan positif pada 2023 DANA sambut 2024 dengan optimis. *Press release.id.* Available: https://pressrelease.kontan.co.id/news/capai-pertumbuhan-positif-pada-2023-dana-sambut-2024-dengan-optimistis.

Rahman, S., sembiring, A., Sireger, D., …., Zen, M. (2023). Python: Dasar dan pemrograman Berorientasi Objek. *Tahta Media Grup.*

Rahmatullah, B. (2021). *Sentiment Analysis Pelaksanaan Work from Home di Indonesia pada Masa Pandemi COVID-19 Menggunakan IndoBERT*.

Saraswati, N.P.V., Yudistira, N., & Adikara, P.P. (2023). Analisis Sentimen terhadap Perundungan Siber pada Twitter menggunakan Algoritma Bidirectional Encoder Representations from Transformer (BERT). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*. 7(2). 909-916.

Septian, R, Saputra, D.I., & Sambasri, S. (2019). Klasifikasi Emosi Menggunakan Convolutional Neural Networks. *Seminar Nasional Teknik Elektro (SENTER)*. 53-62.

Singh, N., Srivastava, S., & Sinha, N. (2 Desember 2016). Consumer preference and satisfaction of M-wallets: a study on North Indian consumers. *International Journal of Bank Marketing, 35*, 944-965.

Syarief, Fauzi. (2017). Pemanfaatan Media Sosial Dalam Proses Pembentukan Opini Publik (Analisa Wacana Twitter Sby). *Jurnal Komunikasi*, 8(3).

Umutsalur, M., & Aydin, I. (2017). A Novel Hybrid Deep Learning Model for Sentiment Classification. *IEEE Access.* Doi: 10.1109/ACCESS.2020.2982538.

Viana, F.N. (2023). "Klasifikasi Emosi Opini Twitter Menggunakan Model Benchmark Indonlu". Universitas Islam Negeri Ar-Raniry, Banda Aceh.

Wang, H., Zhang, L., Yin, K., Luo, H., & Li, J., 2021. Landslide identification using machine learning. *Geoscience Frontiers*. 12(1). 351–364.

Widianto, A. T. B. (2022). "Klasifikasi Emosi pada teks menggunakan Deep Learning". Universitas Islam Indonesia, Yogyakarta.

Wilie, B., Vincentio, K., Winata, G. I., Cahyawijaya, S., Li, X., Lim, Z. Y., Soleman, S., Mahendra, R., Fung, P., Bahar, S., & Purwarianti, A. (2020). IndoNLU: Benchmark and Resources for Evaluating Indonesian Natural Language Understanding. *Proceedings of The 1st Conference of The Asia-Pacific Chapter of The Association for Computational Linguistics and The 10th International Joint Conference on Natural Language Processing.* 843-857.

Yuliastuti, D. (2017). 3 era perkembangan digital payment di Indonesia. *Digination.id.* Available: https://www.digination.id/read/01513/3-era-perkembangan-digital-payment-diindonesia.

LAMPIRAN

Berikut *Source Code* Untuk IndoBERT

```
import pandas as pd
import numpy as np
import torch
from torch.utils.data import DataLoader, Dataset
from transformers import BertTokenizer, BertForSequenceClassification, AdamW
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
```

```
[ ] # ==== 1. Persiapan Data ====
    # Pastikan Anda memiliki file CSV dengan kolom 'comment' dan 'label'
    # Data diharapkan memiliki label yang sudah dikodekan menjadi 0-4 (senang, puas, netral, kecewa, marah)

    class EmotionDataset(Dataset):
        def __init__(self, data, tokenizer, max_len):
            self.data = data
            self.tokenizer = tokenizer
            self.max_len = max_len

        def __len__(self):
            return len(self.data)

        def __getitem__(self, index):
            comment = str(self.data.iloc[index]['comment'])
            label = self.data.iloc[index]['label']
```

```
# Tokenisasi data
encoding = self.tokenizer.encode_plus(
    comment,
    add_special_tokens=True,
    max_length=self.max_len,
    padding='max_length',
    truncation=True,
    return_tensors='pt',
)
return {
    'input_ids': encoding['input_ids'].flatten(),
    'attention_mask': encoding['attention_mask'].flatten(),
    'label': torch.tensor(label, dtype=torch.long)
}
```

```
[ ] # Load data
    train_data = pd.read_csv('/content/DATA_TRAIN_bersih')
    val_data = pd.read_csv('/content/DATA_VALID_bersih')
    test_data = pd.read_csv('/content/DATA_TEST_bersih')
```

```
[ ] # Mapping label string ke angka
    label_mapping = {
        'senang': 0,
        'puas': 1,
        'netral': 2,
        'kecewa': 3,
        'marah': 4
    }
    train_data['label'] = train_data['label'].map(label_mapping)
    val_data['label'] = val_data['label'].map(label_mapping)
    test_data['label'] = test_data['label'].map(label_mapping)
```

```
[ ] # Validasi bahwa label sudah berupa angka
    assert train_data['label'].notnull().all(), "Label train_data ada yang tidak terkonversi!"
    assert val_data['label'].notnull().all(), "Label val_data ada yang tidak terkonversi!"
    assert test_data['label'].notnull().all(), "Label test_data ada yang tidak terkonversi!"
```

```
[ ] # Tokenizer IndoBERT
    tokenizer = BertTokenizer.from_pretrained('indobenchmark/indobert-base-p1')

    # Dataset
    max_len = 128
    batch_size = 32
    train_dataset = EmotionDataset(train_data, tokenizer, max_len)
    val_dataset = EmotionDataset(val_data, tokenizer, max_len)
    test_dataset = EmotionDataset(test_data, tokenizer, max_len)

    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
    val_loader = DataLoader(val_dataset, batch_size=batch_size)
    test_loader = DataLoader(test_dataset, batch_size=batch_size)
```

```
[ ]  # ==== 2. Model IndoBERT ====
     device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
     model = BertForSequenceClassification.from_pretrained('indobenchmark/indobert-base-p1', num_labels=5)
     model = model.to(device)
```

```
[ ]  # Optimizer dan scheduler
     optimizer = AdamW(model.parameters(), lr=2e-5)
```

```
[ ]  # ==== 3. Fungsi Pelatihan dan Evaluasi ====
     def train_epoch(model, data_loader, optimizer, device):
         model.train()
         total_loss = 0
         correct = 0
         total = 0

         for batch in data_loader:
             optimizer.zero_grad()

             input_ids = batch['input_ids'].to(device)
             attention_mask = batch['attention_mask'].to(device)
             labels = batch['label'].to(device)

             outputs = model(input_ids=input_ids, attention_mask=attention_mask, labels=labels)
             loss = outputs.loss
             logits = outputs.logits

             total_loss += loss.item()
             preds = torch.argmax(logits, dim=1)
             correct += (preds == labels).sum().item()
             total += labels.size(0)

             loss.backward()
             optimizer.step()
```

```
     return total_loss / len(data_loader), correct / total

 def eval_model(model, data_loader, device):
     model.eval()
     total_loss = 0
     correct = 0
     total = 0
     all_preds = []
     all_labels = []

     with torch.no_grad():
         for batch in data_loader:
             input_ids = batch['input_ids'].to(device)
             attention_mask = batch['attention_mask'].to(device)
             labels = batch['label'].to(device)

             outputs = model(input_ids=input_ids, attention_mask=attention_mask, labels=labels)
             loss = outputs.loss
             logits = outputs.logits
```

```
             total_loss += loss.item()
             preds = torch.argmax(logits, dim=1)
             all_preds.extend(preds.cpu().numpy())
             all_labels.extend(labels.cpu().numpy())
             correct += (preds == labels).sum().item()
             total += labels.size(0)

     return total_loss / len(data_loader), correct / total, all_preds, all_labels
```

```
[ ]  # ==== 4. Training Loop ====
     epochs = 10
     train_losses, val_losses = [], []
     train_accuracies, val_accuracies = [], []

     for epoch in range(epochs):
         train_loss, train_acc = train_epoch(model, train_loader, optimizer, device)
         val_loss, val_acc, _, _ = eval_model(model, val_loader, device)

         train_losses.append(train_loss)
         val_losses.append(val_loss)
         train_accuracies.append(train_acc)
         val_accuracies.append(val_acc)

         print(f"Epoch {epoch+1}/{epochs}")
         print(f"Train Loss: {train_loss:.4f}, Train Acc: {train_acc:.4f}")
         print(f"Val Loss: {val_loss:.4f}, Val Acc: {val_acc:.4f}")
```

```
[ ] # ==== 5. Evaluasi pada Data Test ====
    test_loss, test_acc, test_preds, test_labels = eval_model(model, test_loader, device)
    print(f"Test Loss: {test_loss:.4f}, Test Acc: {test_acc:.4f}")
```

```
Test Loss: 1.5699, Test Acc: 0.6800
```

```
[ ] # Confusion Matrix
    cm = confusion_matrix(test_labels, test_preds)
    class_names = ['senang', 'puas', 'netral', 'kecewa', 'marah']
    df_cm = pd.DataFrame(cm, index=class_names, columns=class_names)

    plt.figure(figsize=(10, 7))
    sns.heatmap(df_cm, annot=True, fmt="d", cmap="Blues")
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()
```

```
[ ] # Classification Report
    print("Classification Report:")
    print(classification_report(test_labels, test_preds, target_names=class_names))
```

```
# ==== 6. Plot Grafik Accuracy dan Loss ====
plt.figure(figsize=(12, 5))

# Plot Loss
plt.subplot(1, 2, 1)
plt.plot(range(1, epochs+1), train_losses, label='Train Loss')
plt.plot(range(1, epochs+1), val_losses, label='Val Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()

# Plot Accuracy
plt.subplot(1, 2, 2)
plt.plot(range(1, epochs+1), train_accuracies, label='Train Accuracy')
plt.plot(range(1, epochs+1), val_accuracies, label='Val Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

Berikut Source Code untuk IndoBERT dengan CNN

```
import pandas as pd
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
from transformers import BertTokenizer, TFBertModel
```

```
[ ] # Pastikan Anda memiliki IndoBERT pretrained model
    MODEL_NAME = "indobenchmark/indobert-base-p2"
    tokenizer = BertTokenizer.from_pretrained(MODEL_NAME)
    bert_model = TFBertModel.from_pretrained(MODEL_NAME)
```

```
[ ] # Load data
    data_path = "/content/DATA_2000_BERSIH.csv"  # Ganti dengan path file Anda
    data = pd.read_csv(data_path, sep=',', header=None, names=['comment', 'label'])
```

```
[ ] # Label encoding
    from sklearn.preprocessing import LabelEncoder
    label_encoder = LabelEncoder()
    data['label'] = label_encoder.fit_transform(data['label'])
    num_classes = len(label_encoder.classes_)
```

```
[ ] # Split data
    from sklearn.model_selection import train_test_split

    # Membagi data menjadi 80% data training dan 20% data sisanya (validasi + testing)
    train_texts, temp_texts, train_labels, temp_labels = train_test_split(
        data['comment'], data['label'], test_size=0.2, random_state=42
    )

    # Membagi 20% data sisanya menjadi 10% validasi dan 10% testing
    valid_texts, test_texts, valid_labels, test_labels = train_test_split(
        temp_texts, temp_labels, test_size=0.5, random_state=42
    )
```

```
[ ] # Tokenization
    def tokenize_texts(texts, tokenizer, max_len=128):
        return tokenizer(
            list(texts),
            max_length=max_len,
            padding='max_length',
            truncation=True,
            return_tensors="tf"
        )

    max_len = 128
    train_encodings = tokenize_texts(train_texts, tokenizer, max_len)
    test_encodings = tokenize_texts(test_texts, tokenizer, max_len)
```

```
[ ] # Prepare datasets
    train_dataset = tf.data.Dataset.from_tensor_slices((
        dict(train_encodings), train_labels
    )).shuffle(1000).batch(32)
```

```
[ ] test_dataset = tf.data.Dataset.from_tensor_slices((
        dict(test_encodings), test_labels
    )).batch(32)
```

```
[ ] # Model definition
    input_ids = tf.keras.layers.Input(shape=(max_len,), dtype=tf.int32, name="input_ids")
    attention_mask = tf.keras.layers.Input(shape=(max_len,), dtype=tf.int32, name="attention_mask")

    bert_output = bert_model(
        input_ids,
        attention_mask=attention_mask
    )
```

```
[ ] # Combine BERT with CNN
    sequence_output = bert_output.last_hidden_state
    conv1 = tf.keras.layers.Conv1D(128, kernel_size=3, activation='relu')(sequence_output)
    conv1 = tf.keras.layers.GlobalMaxPooling1D()(conv1)

    output = tf.keras.layers.Dense(num_classes, activation='softmax')(conv1)
    model = tf.keras.models.Model(inputs=[input_ids, attention_mask], outputs=output)

    # Compile model
    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=5e-5),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
    )
```

```
    # Train model
    epochs = 10
    history = model.fit(
        train_dataset,
        validation_data=test_dataset,
        epochs=epochs
    )
```

```
[ ] # Evaluate model
    test_loss, test_acc = model.evaluate(test_dataset)
    print(f"Test Accuracy: {test_acc}")
```

```
7/7 [==============================] - 2s 242ms/step - loss: 1.3708 - accuracy: 0.6667
Test Accuracy: 0.6666666865348816
```

```
[ ] # Predictions
    predictions = model.predict(test_dataset)
    predicted_labels = np.argmax(predictions, axis=1)
```

```
7/7 [==============================] - 4s 236ms/step
```

```
    print(label_encoder.classes_)
    print(set(test_labels))
    print(set(predicted_labels))
```

```
['kecewa' 'label' 'marah' 'netral' 'puas' 'senang']
{0, 2, 3, 4, 5}
{0, 2, 3, 4, 5}
```

```
[ ] # Definisikan label dan target_names secara manual jika ada perbedaan
    target_names = ['senang', 'puas', 'netral', 'kecewa', 'marah']  #

    # Gunakan parameter 'labels' untuk membatasi ke kelas yang ada
    labels = label_encoder.transform(target_names)  # Konversi nama kelas ke indeks
    report = classification_report(test_labels, predicted_labels, target_names=target_names, labels=labels)
    print(report)
```

```
[ ] #Confusion Matrix
    from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
    import matplotlib.pyplot as plt
    import numpy as np

    # Hitung confusion matrix
    cm = confusion_matrix(test_labels, predicted_labels)

    # Pastikan label yang sesuai
    unique_labels = np.unique(test_labels)

    # Tampilkan confusion matrix
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=unique_labels)
    disp.plot(cmap=plt.cm.Blues)
    plt.title('Confusion Matrix')
    plt.show()
```

```
[ ] # Plot training history
    def plot_training_history(history):
        plt.figure(figsize=(12, 5))

        # Accuracy plot
        plt.subplot(1, 2, 1)
        plt.plot(history.history['accuracy'], label='Train Accuracy')
        plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
        plt.title('Accuracy')
        plt.xlabel('Epoch')
        plt.ylabel('Accuracy')
        plt.legend()

        # Loss plot
        plt.subplot(1, 2, 2)
        plt.plot(history.history['loss'], label='Train Loss')
        plt.plot(history.history['val_loss'], label='Validation Loss')
        plt.title('Loss')
        plt.xlabel('Epoch')
        plt.ylabel('Loss')
        plt.legend()

        plt.show()
```

Berikut *Source Code* untuk IndoBERT dengan RNN

```
import pandas as pd
import numpy as np
import tensorflow as tf
from transformers import BertTokenizer, TFBertModel
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

```
[ ] # Step 1: Load Data
    # ============================
    # Upload your CSV files to Google Colab
    train_df = pd.read_csv('//content/DATA_TRAIN_bersih')
    valid_df = pd.read_csv('/content/DATA_VALID_bersih')
    test_df = pd.read_csv('/content/DATA_TEST_bersih')
```

```
[ ] # Step 2: Preprocessing
    # Encode labels
    label_encoder = LabelEncoder()
    train_df['label'] = label_encoder.fit_transform(train_df['label'])
    valid_df['label'] = label_encoder.transform(valid_df['label'])
    test_df['label'] = label_encoder.transform(test_df['label'])

    num_labels = len(label_encoder.classes_)
```

```
[ ] # Tokenizer and encoding
    pretrained_model_name = 'indobenchmark/indobert-base-p2'
    tokenizer = BertTokenizer.from_pretrained(pretrained_model_name)

    max_length = 128  # Maximum token length

    def encode_data(data):
        return tokenizer(
            list(data['comment']),
            max_length=max_length,
            padding=True,
            truncation=True,
            return_tensors='tf'
        )

    train_encodings = encode_data(train_df)
    valid_encodings = encode_data(valid_df)
    test_encodings = encode_data(test_df)
```

```
[ ] # Convert labels to tensors
    train_labels = tf.keras.utils.to_categorical(train_df['label'], num_classes=num_labels)
    valid_labels = tf.keras.utils.to_categorical(valid_df['label'], num_classes=num_labels)
    test_labels = tf.keras.utils.to_categorical(test_df['label'], num_classes=num_labels)
```

```
[ ] # Step 3: Build Hybrid Model
    class BertRNNModel(tf.keras.Model):
        def __init__(self, bert_model, rnn_units, num_classes):
            super(BertRNNModel, self).__init__()
            self.bert = bert_model
            self.lstm = tf.keras.layers.LSTM(rnn_units, return_sequences=False)
            self.dropout = tf.keras.layers.Dropout(0.3)
            self.dense = tf.keras.layers.Dense(num_classes, activation='softmax')

        def call(self, inputs):
            # Ambil hidden states terakhir dari BERT
            bert_output = self.bert(inputs)[0]  # Shape: (batch_size, sequence_length, hidden_size)
            lstm_output = self.lstm(bert_output)  # LSTM mengolah sequence
            dropout_output = self.dropout(lstm_output)
            return self.dense(dropout_output)
```

```
[ ] # Load pretrained IndoBERT
    bert_model = TFBertModel.from_pretrained(pretrained_model_name)
```

```
[ ] # Define the hybrid model
    rnn_units = 128
    model = BertRNNModel(bert_model, rnn_units, num_labels)
```

```
[ ] # Compile the model
    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=5e-5),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
```

```
    # Step 4: Train the Model
    batch_size = 32
    epochs = 10

    history = model.fit(
        x={'input_ids': train_encodings['input_ids'],
           'attention_mask': train_encodings['attention_mask'],
           'token_type_ids': train_encodings['token_type_ids']},
        y=train_labels,
        validation_data=(
            {'input_ids': valid_encodings['input_ids'],
             'attention_mask': valid_encodings['attention_mask'],
             'token_type_ids': valid_encodings['token_type_ids']},
            valid_labels
        ),
        batch_size=batch_size,
        epochs=epochs
    )
```

```
[ ]  # Step 5: Evaluate the Model
     from sklearn.metrics import classification_report

     # Fungsi evaluasi model
     def evaluate_model(model, encodings, labels):
         # Melakukan prediksi
         predictions = model.predict(
             {
                 'input_ids': encodings['input_ids'],
                 'attention_mask': encodings['attention_mask'],
                 'token_type_ids': encodings['token_type_ids']
             }
         )
         y_true = np.argmax(labels, axis=1)  # Label asli
         y_pred = np.argmax(predictions, axis=1)  # Prediksi
         return y_true, y_pred

     # Mendapatkan y_true dan y_pred
     y_true, y_pred = evaluate_model(model, test_encodings, test_labels)
```

```
[ ]  # Classification Report
     report = classification_report(y_true, y_pred, target_names=label_encoder.classes_)
     print(report)
```

```
     # Confusion Matrix
     conf_matrix = confusion_matrix(y_true, y_pred)
```

```
     # Step 6: Visualizations
     # Confusion Matrix Heatmap
     plt.figure(figsize=(8, 6))
     sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
     plt.xlabel('Predicted')
     plt.ylabel('True')
     plt.title('Confusion Matrix')
     plt.show()
```

```
[ ]  # Plot training history
     def plot_training_history(history):
         plt.figure(figsize=(12, 5))

         # Accuracy plot
         plt.subplot(1, 2, 1)
         plt.plot(history.history['accuracy'], label='Train Accuracy')
         plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
         plt.title('Accuracy')
         plt.xlabel('Epoch')
         plt.ylabel('Accuracy')
         plt.legend()

         # Loss plot
         plt.subplot(1, 2, 2)
         plt.plot(history.history['loss'], label='Train Loss')
         plt.plot(history.history['val_loss'], label='Validation Loss')
         plt.title('Loss')
         plt.xlabel('Epoch')
         plt.ylabel('Loss')
         plt.legend()

         plt.show()

     plot_training_history(history)
```