# OPTIMASI MODEL BILSTM UNTUK ANALISIS SENTIMEN ULASAN FILM MENGGUNAKAN *HYPERPARAMETER TUNING* RANDOM SEARCH


Disusun Oleh:

N a m a : Muhammad Dzaki Arkaan Nasir

NIM : 19523212

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM INDONESIA**

**2023**

**HALAMAN PENGESAHAN DOSEN PEMBIMBING**

# OPTIMASI MODEL BILSTM UNTUK ANALISIS SENTIMEN ULASAN FILM MENGGUNAKAN *HYPERPARAMETER TUNING* RANDOM SEARCH

## TUGAS AKHIR

Disusun Oleh:

N a m a : Muhammad Dzaki Arkaan Nasir
NIM : 19523212

Yogyakarta, 10 Januari 2024

Pembimbing,

(Dr. Syarif Hidayat, S.Kom., M.I.T)

**HALAMAN PENGESAHAN DOSEN PENGUJI**

# OPTIMASI MODEL BILSTM UNTUK ANALISIS SENTIMEN ULASAN FILM MENGGUNAKAN *HYPERPARAMETER TUNING* RANDOM SEARCH

## TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta,

Tim Penguji

Dr. Syarif Hidayat, S.Kom., M.I.T

**Anggota 1**

Arrie Kurniawardhani, S.Si., M.Kom.

**Anggota 2**

Erika Ramadhani, S.T., M.Eng.

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

## HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Muhammad Dzaki Arkaan Nasir

NIM : 19523212

Tugas akhir dengan judul:

# OPTIMASI MODEL BILSTM UNTUK ANALISIS SENTIMEN ULASAN FILM MENGGUNAKAN *HYPERPARAMETER TUNING* RANDOM SEARCH

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 30 Januari 2024


(Muhammad Dzaki Arkaan Nasir)

# HALAMAN PERSEMBAHAN

Puji syukur Alhamdulillah atas kehadirat Allah SWT, berkat rahmat dan kesepatan yang diberikan kepada saya untuk menyelesaikan laporan tugas akhir ini. Laporan ini saya persembahkan sebagai tanda penghargaan yang istimewa kepada Orang tua saya, terutama Bapak Hasrullah dan Ibu Hiladawati Aziroh sebagai bentuk terima kasih atas dukungan serta doa yang tak henti-hentinya diberikan kepada saya, serta menjadi *support system* dalam setiap langkah perkembangan saya. Terima kasih atas semua doa, dukungan, dan kasih sayangnya.

Penghargaan dan terima kasih kepada Bapak Syarif Hidayat S.Kom., M.I.T., sebagai dosen pembimbing selama penyelesaian laporan tugas akhir ini. Terima kasih kepada Bapak Syarif Hidayat atas bimbingan, masukan, serta kesabaran yang diberikan selama proses pembelajaran dan penyelesaian laporan tugas akhir ini. Saya juga ingin memohon maaf atas segala kesalahan dan kekurngan saya selama masa bimbingan.

# HALAMAN MOTO

“Berdoalah agar dimampukan bukan agar dimudahkan”

**(Wisnu Manupraba)**

“Kita menjadi apa yang kita pikirkan”

**(Earl Nightingale)**

# KATA PENGANTAR

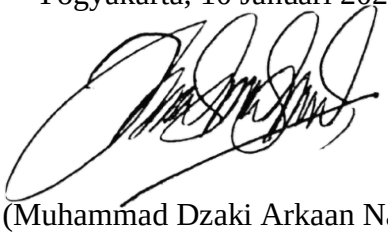
Puji Syukur Alhamdulillah penulis panjatkan atas segala limpahan rahmat Allah SWT sehingga penulis dapat menyelesaikan laporan tugas akhir yang berjudul "Optimasi Model BiLSTM untuk Analisis Sentimen Ulasan Film Menggunakan Hyperparameter Tuning Random Search". Laporan tugas akhir ini disusun sebagai salah satu syarat kelulusan pendidikan tingkat Strata-1 (S1) Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Selama proses penelitian dan penyusunan laporan ini, Penulis menghadapi berbagai kesulitan, baik terkait dengan tantangan dalam penerapan metodologi, maupun kendala waktu dan sumber daya. Namun, dengan bantuan dan dukungan dari berbagai pihak, kesulitan-kesulitan tersebut dapat Penulis atasi. Oleh karena itu, Penulis ingin menyampaikan terima kasih yang tulus kepada semua pihak yang telah memberikan bantuan, dukungan, serta masukan yang berharga terutama kepada:

1. Orang tua penulis, Bapak Hasrullah, S.STP., M.Si. dan Ibu Hiladawati Aziroh, S.STP atas doa dan dukungan yang diberikan kepada Penulis.
2. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.
3. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Ketua Program Studi Informatika Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia.
4. Bapak Dr. Syarif Hidayat, S.Kom., M.I.T. selaku dosen pembimbing yang telah sabar memberikan bimbingan dan masukan hingga penyelesaian laporan tugas akhir ini.
5. Teman-teman dari Cebelap Imoet Club yang selalu memberikan masukan, dukungan dan sudah berjuang bersama dari awal perkuliahan hingga sampai di tahap ini.
6. Teman-teman Business Growth PT. Javan Cipta Solusi yang juga memberikan dukungan dan pengertian selama penulis mengerjakan laporan tugas akhir.

Harapan Penulis atas selesainya Laporan Tugas Akhir ini dapat memberikan manfaat bagi semua pembaca. Penulis menyadari bahwa laporan tugas akhir ini memiliki banyak kekurangan. Terima Kasih

Yogyakarta, 10 Januari 2024



(Muhammad Dzaki Arkaan Nasir)

# SARI

Perkembangan pesat industri film dalam satu dekade terakhir ditambah dengan munculnya berbagai platform streaming saat ini. Membuat peningkatan jumlah penonton film yang meningkat pesat. Salah satu faktor penting dalam keputusan penonton untuk menonton film berdasarkan dari ulasan dari pengguna lain. Dengan ulasan yang kini lebih mudah diakses melalui berbagai platform, analisis sentimen menjadi alat yang sangat berharga untuk memahami persepsi penonton. Mengingat tantangan dalam pengembangan model machine learning, terutama dalam konteks analisis sentimen ulasan film, masalah akurasi menjadi fokus utama. Akurasi model sangat berpengaruh terhadap kinerja model dalam implementasi prediksi sentimen. Oleh karena itu, penting untuk melakukan optimalisasi hyperparameter guna meningkatkan akurasi tersebut. Penelitian ini berusaha mengeksplorasi kombinasi *hyperparameter* dengan menggunakan *Random Search.* Hasil terbaik yang diperoleh kemudian dibandingkan kinerjanya dengan model BiLSTM dengan parameter umum yang ditetapkan manual. Proses membandingkan kinerja kedua model tersebut dilakukan dengan proses *Stratified k-fold cross validation.* Hasilnya diperoleh bahwa model dengan parameter hasil *Random search* lebih unggul dibandingkan dengan model dengan parameter umum yang ditetapkan secara manual. Model dengan parameter umum yang ditetapkan secara manual memperoleh akurasi rata-rata pada *Stratified k-fold cross validation* yaitu 89,51%, 88,66%, 89,19%, dan 89,39% lebih rendah jika dibandingkan dengan model dengan parameter yang dihasilkan melalui proses *Random search* yang memperoleh akurasi rata-rata keseluruhan sebesar 96,99%. Model terbaik hasil *Random Search* kemudian dilakukan pengujian menggunakan sampel ulasan film. Hasil penelitian ini dapat memberikan kontribusi penting dalam bidang analisis sentimen, terkhusus dalam aplikasi NLP untuk ulasan film. Hal ini juga membuka peluang untuk penerapan model BiLSTM yang lebih luas dalam berbagai aspek analisis teks.

Kata kunci: Analisis sentimen, Ulasan film, BiLSTM, IMDb, Random search, Stratified k-fold cross validation.

# GLOSARIUM

| | |
|---|---|
| *Natural Language Processing (NLP)* | Cabang kecerdasan buatan yang berkaitan dengan interaksi antara komputer dan bahasa manusia, memungkinkan mesin untuk memahami, menerjemahkan, dan bereaksi terhadap bahasa manusia. |
| *Optimizer* | Algoritma yang digunakan dalam *machine learning* untuk mengurangi kesalahan dengan mengurangi bobot jaringan. |
| *Epochs* | Siklus lengkap ketika data set melewati jaringan dalam proses pembelajaran. |
| *Osilasi* | Fenomena ketika metrik evaluasi seperti loss atau akurasi tidak stabil selama fase pelatihan, tetapi sebaliknya naik dan turun dari iterasi ke iterasi. |
| *Vanishing Gradient* | Masalah dalam pelatihan jaringan dimana gradien yang digunakan menjadi sangat kecil, sehingga pembelajaran menjadi sangat lambat. |
| Fungsi aktivasi | Fungsi yang merubah input lapisan menjadi keluaran yang siap dikirim ke lapisan berikutnya. |
| *Loss* | Ukuran kesalahan antara prediksi model dan nilai sebenarnya. |
| *model* | Representasi matematis yang dibuat berdasarkan data. Model ini belajar dari data yang diberikan untuk membuat prediksi atau keputusan tanpa diprogram secara eksplisit |

# DAFTAR ISI

# DAFTAR TABEL

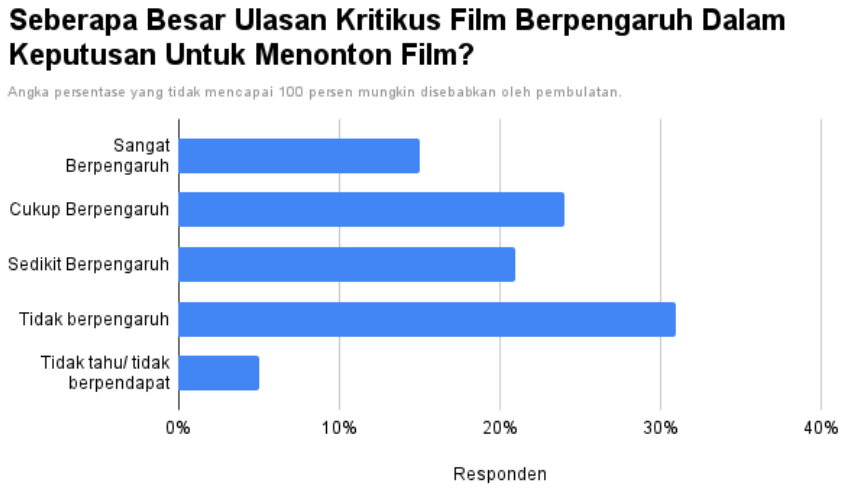
# DAFTAR GAMBAR

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Industri film dalam satu dekade terakhir telah mengalami transformasi yang signifikan, terutama dengan munculnya berbagai platform *streaming* di era digital saat ini. Munculnya berbagai platform *streaming* membuat apa yang dulunya dianggap sebagai media hiburan yang hanya bisa dinikmati di bioskop, kini telah berkembang menjadi suatu fenomena global berkat kemajuan teknologi yang dapat mempermudah penonton untuk melihat film dari mana saja. Di Indonesia saja, pertumbuhan penonton film dari tahun tahun 2010 hingga 2020 saja mengalami peningkatan sebanyak lima kali lipat (Iswara, 2020).

Sejalan dengan pertumbuhan industri film dan platform *streaming*, tak heran jika jumlah ulasan dan respons film yang tersedia online juga meningkat dengan cepat. Ulasan yang dulunya hanya terbatas pada kolom koran atau majalah, kini telah berkembang ke berbagai platform digital seperti situs web, media sosial, dan platform *streaming* itu sendiri. Hal ini juga menjadikan penonton yang sekarang memiliki akses lebih mudah dalam mengungkapkan pendapat dan tanggapan mereka tentang film menjadikan mereka sebagai bagian aktif dalam komunitas pecinta film.




Gambar 1.1 Pengaruh ulasan kritikus film di Amerika Serikat

Sumber: (Influence of movie critics reviews in the U.S. 2017)

Bagi banyak orang, ulasan film memegang peran penting dalam menentukan apakah mereka akan menonton sebuah film atau tidak. Survei yang dilakukan pada tahun 2017 oleh situs Statista dan dapat dilihat pada Gambar 1.1 memperlihatkan bahwa 15% dari responden menyatakan bahwa ulasan yang diberikan oleh kritikus film memiliki pengaruh kuat terhadap keputusan mereka untuk menonton film. (Eliashberg & Shugan, 1997) mengemukakan bahwa ulasan film dapat berfungsi sebagai prediktor apakah film tersebut akan sukses atau tidak. Selain itu, bagi pembuat film, ulasan dapat memberikan wawasan yang berharga mengenai apa yang berhasil dan apa yang kurang dari karya milik mereka, memungkinkan perbaikan dan inovasi dalam proyek-proyek film yang akan datang. Oleh karena itu, memahami dinamika ulasan film dan analisis sentimen yang terdapat di dalamnya menjadi krusial, bukan hanya bagi penonton tetapi bagi industri film itu sendiri.

Relevansi analisis sentimen dalam menilai ulasan film sangatlah tinggi. Analisis sentimen itu sendiri merupakan teknik pemrosesan bahasa alami manusia (*Natural Language Processing*/NLP) yang bertujuan untuk menentukan sentimen atau emosi yang terkandung dalam suatu ulasan, apakah itu positif, netral, atau negatif. Analisis sentimen dapat juga digunakan untuk mengidentifikasi bagaimana penonton merespon sebuah film berdasarkan ulasan yang mereka tulis. Sebuah ulasan yang mengandung kata seperti “mengesankan”, “luar biasa”, atau “hebat” cenderung menunjukan sentimen positif, sedangkan kata-kata seperti “membosankan”, “buruk”, atau “kecewa” menunjukkan sentimen yang negatif. Melakukan analisis sentimen secara manual terhadap volume data ulasan film yang besar, adalah tugas yang memakan waktu dan seringkali menghasilkan interpretasi yang kurang akurat (Wang & Manning, 2012). Oleh karena itu diperlukannya sebuah metode yang dapat memungkinkan kita untuk mengekstrak makna dari ulasan dengan skala besar dan mendapatkan wawasan yang lebih mendalam mengenai persepsi penonton film.

Selama beberapa dekade terakhir, metode berbasis *Machine Learning* seperti *Support Vector Machines* (SVM) dan *Naive Bayes* telah diterapkan dalam berbagai studi untuk analisis sentimen dengan hasil yang bervariasi (Wang & Manning, 2012). Namun, dengan meningkatnya kompleksitas data teks, metode tradisional sering kali menemui hambatan dalam menangkap nuansa dan konteks dalam teks. Kemudian digunakan pendekatan berbasis *Deep Learning*, khususnya metode *Bidirectional Long Short-Term Memory* (BiLSTM) dalam upaya mengatasi keterbatasan tersebut.

BiLSTM ini sendiri merupakan pengembangan lebih lanjut dari model *Long Short-Term Memory* (LSTM), di mana informasi tidak hanya dipelajari dari depan ke belakang,

tetapi juga sebaliknya, yaitu dari belakang ke depan. BiLSTM ini sendiri sebenarnya merupakan dua buah jaringan LSTM yang terpisah, jaringan LSTM pertama mempelajari dari depan ke belakang dan LSTM yang lainnya mempelajari dari belakang ke depan atau secara terbalik. Sehingga BiLSTM dapat menangkap konteks lebih baik daripada LSTM (Graves & Schmidhuber, 2005). Selain memahami informasi dari kedua arah dalam teks, BiLSTM sendiri juga menawarkan kemampuan untuk mengingat informasi baik informasi jangka panjang maupun informasi jangka pendek yang membuat BiLSTM menjadi kandidat kuat untuk melakukan analisis sentimen (Schuster & Paliwal, 1997).

Namun, meskipun BiLSTM telah menunjukkan potensi besar dengan beberapa penelitian seperti yang dilakukan oleh Rolangon dkk., (2023) BiLSTM memiliki akurasi sebesar 86% dalam analisis sentimen. Kemudian juga BiLSTM memperoleh akurasi sebesar 80,5%, 90,59%, dan 85,79% dalam berbagai percobaan dataset (Hameed & Garcia-Zapirain, 2020). Dengan akurasi BiLSTM tersebut masih dapat dilakukan peningkatan dengan pemilihan *hyperparameter* yang tepat. Seperti yang dijelaskan oleh Bergstra dkk., (2012), tanpa *hyperparameter tuning* yang tepat, model mungkin tidak mencapai potensinya yang sebenarnya.

Oleh karena itu dilakukan *hyperparameter tuning* untuk mencari kombinasi *hyperparameter* terbaik melalui metode seperti *Random Search*. Metode pencarian *hyperparameter* tersebut telah menjadi topik utama dalam penelitian yang dilakukan oleh Bergstra dkk., (2012). Mengingat potensi BiLSTM dan pentingnya *hyperparameter tuning* dalam mengoptimalkannya, penelitian ini bertujuan untuk mengeksplorasi kombinasi terbaik dari *hyperparameter* untuk meningkatkan akurasi analisis sentimen ulasan film menggunakan metode *Random Search.* Random Search menawarkan pendekatan yang lebih luas dan acak dalam mengeksplorasi ruang *hyperparameter*, dengan potensi menemukan kombinasi yang lebih efektif dibandingkan dengan metode tradisional.

Diperlukan eksplorasi penggunaan Random Search dalam *hyperparameter tuning* pada model BiLSTM untuk meningkatkan akurasi analisis sentimen ulasan film. Dikarenakan dalam konteks analisis sentimen, akurasi menjadi aspek penting. Akurasi yang tinggi dalam menganalisis sentimen akan memastikan bahwa penilaian terhadap respon penonton adalah akurat dan dapat diandalkan.

## 1.2 Rumusan Masalah

Mengingat tantangan yang dihadapi dalam pengembangan model machine learning terutama dalam konteks analisis sentimen ulasan film adalah masalah akurasi, dimana akurasi model berpengaruh terhadap kinerja model dalam implementasi prediksi sentimen. Penting untuk dilakukan optimalisasi *hyperparameter* untuk meningkatkan akurasi tersebut. Oleh karena itu, penelitian ini berfokus pada penerapan metode Random Search untuk menentukan parameter optimal yang dapat meningkatkan akurasi model.

## 1.3 Batasan Masalah

Dalam rangka memberikan fokus yang jelas dalam penelitian ini, beberapa batasan yang yang diterapkan sebagai berikut:

a. Penelitian ini menggunakan data ulasan film dari penelitian yang dilakukan oleh Maas dkk., (2011).
b. Data ulasan film terbatas hanya menggunakan Bahasa Inggris.
c. Metode analisis sentimen yang digunakan pada penelitian ini berfokus kepada metode BiLSTM
d. *Hyperparameter* yang akan dilakukan *tuning* terbatas hanya kepada unit LSTM, besaran *dropout*, dan unit *dense*.
e. Penentuan kombinasi *hyperparameter* menggunakan *Random Search*.
f. Klasifikasi sentimen hanya terbatas kepada sentimen positif dan sentimen negatif.
g. Evaluasi performa model menggunakan *Confusion Matrix* seperti menghitung *accuracy*, *precision*, *recall*, dan *f1-score*.

Dengan menentukan batasan-batasan tersebut, diharapkan penelitian ini dapat lebih fokus dan hasil yang diperoleh lebih relevan dan spesifik sesuai dengan tujuan penelitian.

## 1.4 Tujuan Penelitian

Tujuan dilakukan penelitian ini adalah mengoptimalkan analisis sentimen ulasan film dengan menggunakan model BiLSTM melalui proses *hyperparameter tuning* untuk meningkatkan akurasi dalam mengklasifikasikan sentimen.

## 1.5 Manfaat Penelitian

Manfaat yang diharapkan didapat pada penelitian ini adalah:

a. Penelitian ini akan memberikan pemahaman tentang bagaimana model BiLSTM bekerja dalam konteks analisis sentimen dan bagaimana *hyperparameter* mempengaruhi kinerjanya.
b. Hasil penelitian ini dapat dijadikan sebagai dasar atau tambahan literatur bagi penelitian selanjutnya yang berkaitan dengan analisis sentimen atau penerapan model BiLSTM.
c. Teknik *hyperparameter tuning* yang digunakan dalam penelitian ini dapat diadopsi atau disesuaikan untuk penelitian lain yang serupa.

## 1.6 Sistematika Penulisan

Sistematika penulisan dalam tugas akhir ini disusun menggunakan sistematika sebagai berikut:

a. BAB I PENDAHULUAN

   Bab I terdiri dari latar belakang permasalahan, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.
b. BAB II LANDASAN TEORI

   Bab II berisi tentang penjelasan teori-teori yang digunakan pada penelitian ini dan juga berisi tentang penelitian-penelitian sebelumnya.
c. BAB III METODOLOGI PENELITIAN

   Bab III menjelaskan tentang tahapan-tahapan yang dilakukan dalam penelitian ini.
d. BAB IV HASIL DAN PEMBAHASAN

   Bab IV berisi mengenai penjelasan dari hasil yang didapat dari sentimen analisis ulasan film menggunakan metode BiLSTM.
e. BAB V KESIMPULAN DAN SARAN

   Bab V memberikan kesimpulan yang didapatkan selama penelitian dan saran yang diberikan untuk penelitian sebelumnya.

# BAB II
# LANDASAN TEORI

## 2.1 Penelitian Terdahulu

Penelitian terdahulu merupakan hal yang sangat penting bagi penulis. Penelitian tersebut dapat menjadi dasar dalam menyusun maupun melakukan penelitian. Penelitian terdahulu juga dapat diketahui persamaan maupun perbedaan antara penelitian satu dengan penelitian yang lainnya. Baik dari metode yang digunakan dan hasil yang dihasilkan. Hal tersebut juga dapat mengurangi duplikasi penelitian yang dilakukan sekarang dengan penelitian terdahulu.

Penelitian mengenai analisis sentimen sudah pernah dilakukan oleh Lakshmi dkk., (2020) dengan menggunakan pendekatan model *Convolutional Neural Network* (CNN) dan LSTM untuk menganalisis ulasan film yang berasal dari situs *Internet Movie Database* (IMDb). Hasil akurasi yang didapat saat mengolah data sebanyak 50000 ulasan film menunjukkan bahwa LSTM memiliki akurasi sebesar 85,88% dan CNN memiliki akurasi 85.97%. Dalam penelitian ini juga ditemukan bahwa *deep learning* memiliki tingkat akurasi lebih baik jika dibandingkan dengan tradisional *Machine Learning* seperti *Naïve Bayes* dan *K-Nearest Neighbors* (KNN). Pada tahun yang sama juga, Hameed & Garcia-Zapirain (2020) melakukan penelitian dengan menggunakan *single-layered* BiLSTM untuk analisis sentimen ulasan film dan didapati bahwa BiLSTM mampu mencapai akurasi tertinggi sebesar 90,59% dengan menggunakan tiga dataset yang diujikan.

Pada tahun 2022, penelitian selanjutnya yang dilakukan oleh Gifari dkk. (2022), melakukan analisis sentimen dengan menggunakan *Term Frequency-Inverse Document Frequency* (TF-IDF) dan *Support Vector Machine* (SVM) diperoleh akurasi 85% dan ditemukan bahwa banyak sedikitnya data latih akan mempengaruhi hasil yang didapatkan. Pada tahun yang sama, Witanto dkk., (2022) melakukan eksplorasi terhadap model LSTM dengan membandingkan performa dari dua *optimizer* yaitu Adam dan RMSprop untuk analisis sentimen ulasan film. Akurasi dari *RMSprop Optimizer* diperoleh sebesar 80,07% dan *Adam optimizer* sebesar 77,11%. Eksplorasi lebih dalam juga dilakukan oleh Handayani dkk., (2022) dengan menggunakan LSTM dengan *Word2Vec* dan *Hyperparameter tuning* dalam melakukan analisis tweet meskipun akurasi yang didapatkan hanya sebesar 57,35%.

Penelitian yang dilakukan oleh Rahman dkk., (2022) pada saat melakukan analisis sentimen terhadap kebijakan *New Normal* menemukan bahwa model dengan LSTM memiliki

akurasi lebih baik dengan akurasi 83,33% dibandingkan dengan model Naïve Bayes yang memiliki akurasi sebesar 82%. Hal ini juga selaras dengan temuan Lakshmi dkk., (2020). Studi oleh Alghifari dkk., (2022) menunjukkan jika BiLSTM dalam analisis sentimen terhadap layanan Grab Indonesia ditemukan bahwa model BiLSTM memiliki kekurangan yaitu memerlukan data yang besar untuk mengurangi *overfitting* dan komputasi model yang lama. Meskipun demikian, model BiLSTM menunjukkan akurasi yang lebih tinggi sebesar 91% dibandingkan model LSTM biasa yang hanya sebesar 76%. Penelitian lebih lanjut yang dilakukan oleh Rolangon dkk., (2023) dengan melakukan komparasi antara beberapa algoritma seperti LSTM, BiLSTM, GRU, dan SimpleRNN untuk analisis pengguna Twitter terhadap layanan rumah sakit selama pandemi Covid-19. Penelitian tersebut menghasilkan bahwa BiLSTM sedikit lebih unggul dalam hal akurasi dibandingkan dengan model lainnya dengan akurasi sebesar 86% dan unggul dalam beberapa aspek matriks evaluasi seperti *accuracy*, *recall*, dan MCC.

Rangkuman penelitian sebelumnya yang menjadi acuan pada penelitian ini yang dapat dilihat pada Tabel 2.1 untuk memberikan pemahaman terkait berbagai teknik dan pendekatan yang telah diuji dan diterapkan.

Tabel 2.1 Penelitian Terdahulu

| **Judul Penelitian** | **Penulis** | **Metode** | **Akurasi** | **Data set** | **Bahasa dalam Data set** | **Temuan Lain** |
|---|---|---|---|---|---|---|
| Perbandingan Algoritma LSTM Untuk Analisis Sentimen Pengguna Twitter Terhadap Layanan Rumah Sakit Saat Pandemi Covid-19 | (Rolangon et al., 2023) | LSTM | 85% | 13.321 tweet | Indonesia | Model BiLSTM lebih unggul dari GRU dalam beberapa aspek matriks yaitu *Accuracy*, *Recall*, dan MCC. |
| | | BiLSTM | 86% | | | |
| | | GRU | 86% | | | |
| | | SimpleRNN | 75% | | | |
| Analisis Sentimen Review Film Menggunakan TF-IDF dan Support Vector Machine | (Gifari et al., 2022) | SVM, TF-IDF | 85% | tweet | Indonesia | Semakin Banyak *data training* yang digunakan maka akan mempengaruhi kinerja sistem. |

| Judul Penelitian | Penulis | Metode | Akurasi | Data set | Bahasa dalam Data set | Temuan Lain |
|---|---|---|---|---|---|---|
| Implementasi LSTM pada Analisis Sentimen Review Film Menggunakan Adam dan RMSprop Optimizer | (Witanto et al., 2022) | LSTM dengan Adam Optimizer, Batch Size 64, Epoch 5 | 77,11% | 14.000 data ulasan film | - | - |
| | | LSTM dengan RMSprop Optimizer, Batch Size 128, Epoch 5 | 80,07% | | | |
| Analisis Sentimen pada Data Ulasan Twitter dengan LSTM | (Handayani et al., 2022) | LSTM, *hyperparameter tuning*, CBOW | 57,35% | 10.806 tweet. | Indonesia. | Hasil terbaik *Hyperparameter Tuning* adalah jumlah neuron sebesar 150, jumlah epoch sebanyak 30, serta menggunakan softmax sebagai fungsi aktivasi. |
| Analisis Perbandingan Algoritma LSTM dan Naïve Bayes untuk Analisis Sentimen | (Rahman et al., 2022) | LSTM | 83,33% | 1.590 tweet | Indonesia | - |
| | | Naïve Bayes | 82% | | | |
| Implementasi Bidirectional LSTM untuk Analisis Sentimen Terhadap Layanan Grab Indonesia | (Alghifari et al., 2022) | LSTM | 76% | 5000 ulasan dari PlayStore | Indonesia | BiLSTM memiliki akurasi lebih baik dikarenakan dapat mempelajari makna dalam teks dari dua arah sekaligus. |

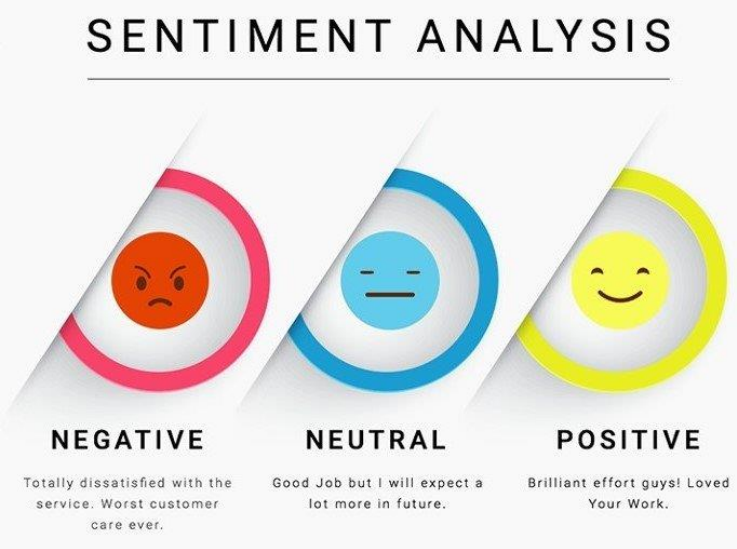
| Judul Penelitian | Penulis | Metode | Akurasi | Data set | Bahasa dalam Data set | Temuan Lain |
|---|---|---|---|---|---|---|
| | | BiLSTM | 91% | | | Tetapi, memiliki kelemahan yaitu memerlukan data yang besar untuk menghindari *overfitting* dan waktu komputasi yang lama. |
| *Sentiment Classification Using a Single-Layered* BiLSTM Model | (Hameed & Garcia-Zapirain, 2020) | BiLSTM | 80,50% | *Movie Review*, 10.662 ulasan (Pang & Lee, 2005) | Inggris | - |
| | | | 90,59% | IMDb dataset, 50.000 ulasan (Maas et al., 2011) | | |
| | | | 85,79% | Standford Sentiment Treebank, 9.613 ulasan (Socher et al., 2013) | | |
| *A Deep Learning Approach for Sentiment Analysis of Movie Review* | (Lakshmi et al., 2020) | LSTM | 85,88% | 50.000 ulasan film | Inggris | Model LSTM dan CNN memiliki akurasi lebih baik dari pada tradisional Machine Learning seperti Naïve Bayes dan K-Nearest Neighboars. |
| | | CNN | 85,97% | | | |

Berdasarkan hasil kajian penelitian terdahulu, penelitian ini akan melakukan analisis sentimen dengan menggunakan model BiLSTM dengan parameter ditentukan dan model BiLSTM yang akan dilakukan *hyperparameter tuning* untuk melihat seberapa besar atau signifikan perbedaan akurasi keduanya. Analisis ini juga akan menggunakan data ulasan film tetapi menggunakan bahasa Inggris untuk analisis sentimen.

## 2.2 Dasar Teori

### 2.2.1 Analisis Sentimen

Analisis sentimen atau juga dikenal sebagai *opinion mining* adalah bidang penelitian yang memiliki fokus terhadap identifikasi, ekstraksi, dan pemahaman informasi berupa teks terhadap suatu entitas. Entitas tersebut dapat mewakili individu, peristiwa, atau suatu topik yang kemungkinan tercantum dalam suatu ulasan (Medhat et al., 2014). Analisis sentimen bertujuan untuk mengelompokkan suatu kalimat berdasarkan polaritas teks kedalam sentimen positif maupun negatif (Zuhdi et al., 2019). Berikut ilustrasi dari analisis sentimen yang dapat dilihat pada Gambar 2.1.

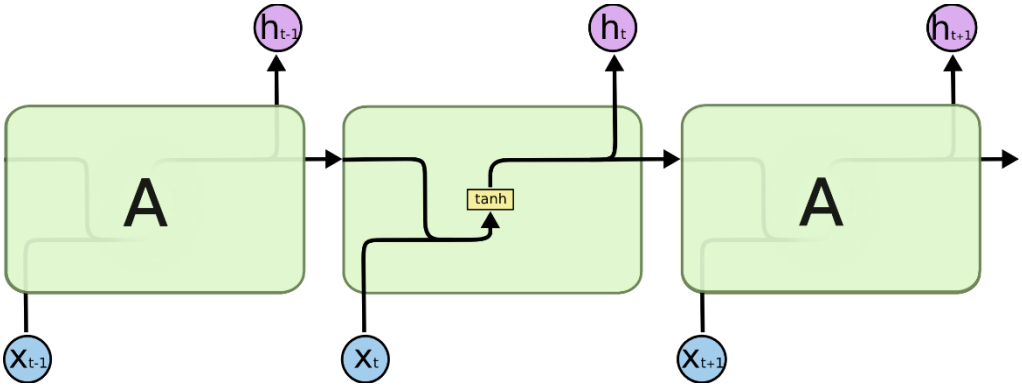



Gambar 2.1 Ilustrasi Analisis Sentimen

### 2.2.2 Long Short-Term Memory (LSTM)

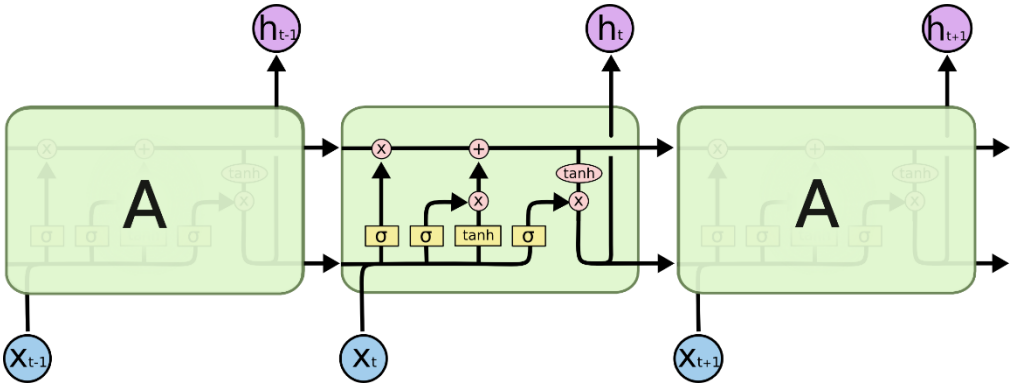
*Long Short-Term Memory* (LSTM) merupakan pengembangan model dari *Recurrent Neural Network* (RNN) yang dirancang untuk mengatasi masalah *vanishing gradient* pada RNN biasa (Hochreiter & Schmidhuber, 1997). Pada dasarnya, semua RNN memiliki bentuk rantai modul jaringan syaraf tiruan yang berulang dan pada RNN standar, modul pengulangan

tersebut memiliki struktur yang sederhana, seperti lapisan tanh tunggal dan dapat dilihat pada Gambar 2.2.




Gambar 2.2 Arsitektur RNN Standar

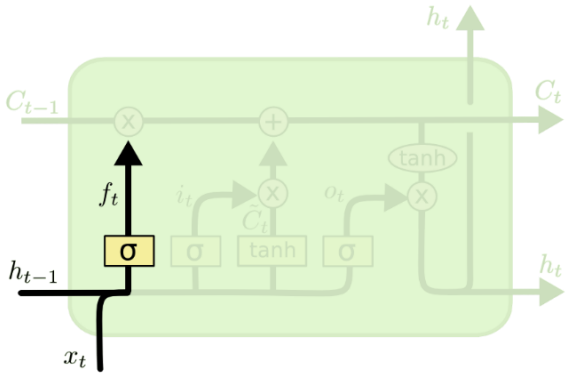
Pada model LSTM juga terdapat struktur rantai tersebut, tetapi modul pengulangan memiliki struktur yang berbeda jika dibandingkan dengan arsitektur RNN standar yang dapat dilihat pada Gambar 2.2. LSTM bukan hanya memiliki satu lapisan saja, tetapi memiliki empat lapisan berbeda yang memiliki fungsi masing-masing diantaranya adalah *forget gate*, *input gate*, *update gate*, dan *output gate*.




Gambar 2.3 Arsitektur LSTM

Dalam arsitektur LSTM pada Gambar 2.3, komponen yang dikenal sebagai *cell gates* terletak pada bagian bawah. Fungsi utama dari *cell gates* adalah untuk mengatur informasi yang akan ditransfer ke unit berikutnya atau ke dalam *cell state*. Sementara, cel state yang merupakan bagian atas dari arsitektur LSTM bertugas untuk mengalirkan informasi ke unit-unit selanjutnya (Pasaribu et al., 2020).

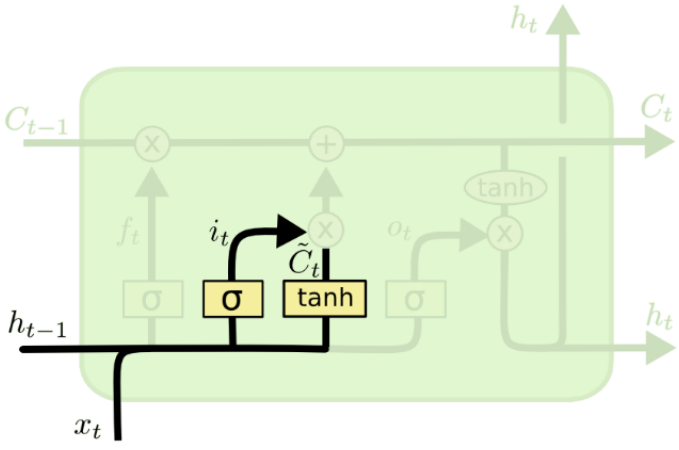
Langkah pertama dalam LSTM adalah menentukan informasi apa yang akan dibuang dari *cell state*. Dalam tahap ini keputusan diambil oleh lapisan sigmoid yang disebut sebagai *forget gate*. Lapisan ini memeriksa $h_{t-1}$ dan $x_t$, dan memberikan keluaran berupa angka 1 atau 0 dalam *cell state* $C_{t-1}$. Angka 1 berarti akan disimpan sepenuhnya dan 0 berarti akan dibuang/ dilupakan sepenuhnya.




Gambar 2.4 *Forget gate*

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

Dalam persamaan 2.1, *forget gate* ($f_t$) menentukan informasi mana dari *cell state* sebelumnya yang harus dilupakan atau dibuang, σ merupakan fungsi aktivasi sigmoid, $W_f$ merupakan bobot untuk *forget gate*, $b_f$ adalah bias untuk *forget gate*, $h_{t-1}$ adalah keluaran tersembunyi dari langkah sebelumnya, dan $x_t$ adalah masukan pada langkah saat ini.

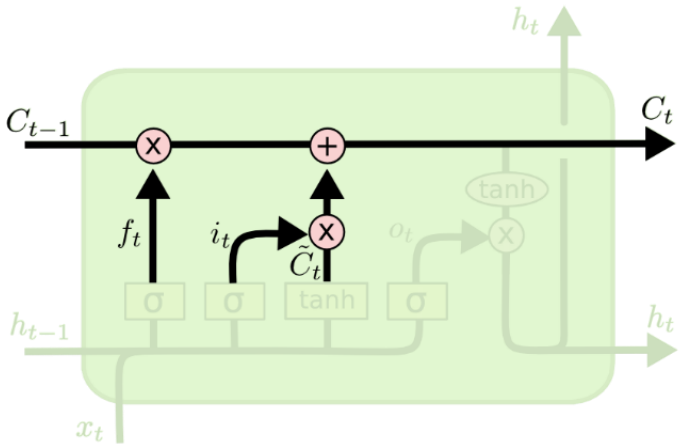



Gambar 2.5 Input *Gate*

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

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad 2.3$$

Input *gate* dalam LSTM bertanggung jawab untuk memutuskan informasi baru mana yang akan ditambahkan ke *cell state*. Input *gate* terdiri dari dua bagian utama, input *gate layer* yang merupakan lapisan untuk memutuskan nilai yang akan diperbarui. Rumus dari input *gate layer* ($i_t$) dapat dilihat pada persamaan 2.2. Di mana $W_i$ adalah bobot untuk input *gate layer*, $b_i$ adalah bias untuk input *gate layer*. Sementara *candidate layer* memiliki rumus yang dapat dilihat pada persamaan 2.3. *Candidate layer* merupakan bagian krusial dari input *gate*. Lapisan ini menciptakan vektor kandidat *cell state* ($\tilde{C}_t$) yang berisi informasi kandidat yang mungkin ditambahkan ke *cell state*. Di mana fungsi tanh memberikan kemampuan untuk memiliki nilai positif (untuk menambahkan informasi) atau nilai negatif (untuk mengurangi informasi), $W_C$ adalah bobot untuk *candidate layer*, $b_C$ adalah bias untuk *candidate layer*.




Gambar 2.6 *Update Cell state*

$$C_t' = f_t * C_{t-1} \quad 2.4$$

$$C_t'' = i_t * \tilde{C}_t \quad 2.5$$

$$C_t = C_t' + C_t'' \quad 2.6$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad 2.7$$

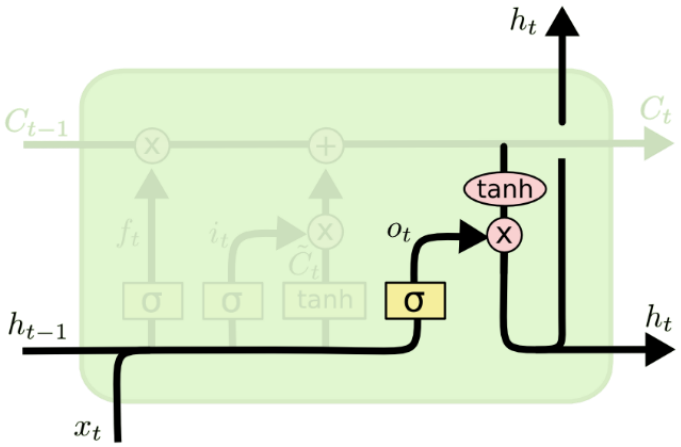
*Layer* selanjutnya dalam LSTM yaitu *update cell state*. Pada proses ini, *cell state* diperbarui berdasarkan informasi dari *forget gate* dan input *gate*. Pertama, *forget gate* telah

menentukan bagian mana dari *cell state* lama yang harus diingat atau dilupakan. Hal ini dilakukan melalui operasi perkalian *element-wise* $(*)$ antara *cell state* sebelumnya $(C_{t-1})$ dan keluaran dari *forget gate* $(f_t)$. Jika nilai *forget gate* dekat dengan 0 maka informasi dari *cell state* lama akan dihapus dan jika dekat dengan 1 maka informasi tersebut akan dipertahankan. Rumus dari persamaan tersebut dapat dilihat pada persamaan 2.3.

Kedua, Input *Gate* telah menentukan informasi baru mana yang harus ditambahkan ke *cell state*. Hal ini dilakukan melalui operasi perkalian *element-wise* $(*)$ antara keluaran dari input *gate* $(i_t)$ dan keluaran dari *candidate layer* $(\tilde{C}_t)$ yang merupakan informasi baru yang diusulkan untuk ditambahkan. Ini menunjukkan informasi apa yang harus disimpan di *cell state*. Untuk persamaan dapat dilihat pada persamaan 2.5.

Akhirnya, *cell state* diperbarui dengan menggabungkan hasil dari dua operasi sebelumnya. Ini dilakukan dengan menambahkan *output* dari kedua operasi tersebut, sehingga menghasilkan *cell state* baru $(C_t)$ dengan rumus pada persamaan 2.6 atau lebih detail pada persamaan 2.7.

*Update cell state* ini merupakan inti dari kemampuan LSTM untuk mempertahankan informasi jangka panjang dengan memutuskan informasi mana saja yang perlu diabaikan dan ditambahkan. Proses ini juga memungkinkan untuk terhindar dari masalah *vanishing gradient* yang sering terjadi dalam RNN tradisional.




Gambar 2.7 *Output Gate*

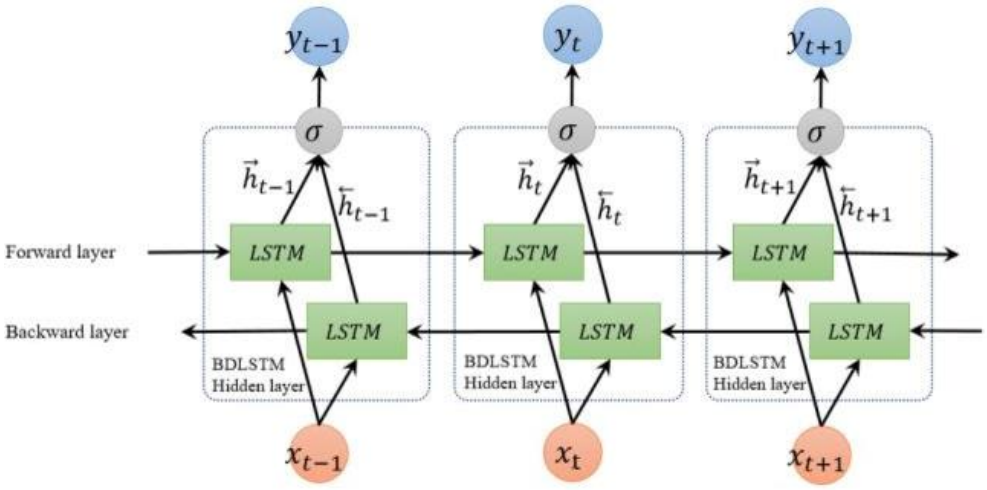
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad 2.8$$

$$h_t = o_t * \tanh(C_t) \quad 2.9$$

*Gate* terakhir dalam LSTM adalah *Output gate*. Fungsi utama dari *output gate* adalah untuk menentukan bagian mana dari *cell state* yang akan digunakan dalam keluaran. *Output gate* terdiri dari dua bagian yang pertama merupakan *output gate layer* yang menggunakan fungsi aktivasi sigmoid untuk menentukan bagian mana dari cell yang harus diteruskan ke *output*. Untuk rumus *output gate layer* dapat dilihat pada persamaan 2.8 dimana $W_o$merupakan bobot untuk *Output Gate* dan $b_o$ merupakan bias dari *output gate*. Kedua, keluaran akhir dari unit LSTM dihitung dengan mengalikan keluaran *output gate* dengan versi tanh dari *cell state* yang dapat dilihat pada persamaan 2.9. Ini memungkinkan jaringan untuk mengontrol dan menyesuaikan informasi yang diteruskan melalui jaringan yang pada akhirnya menghasilkan informasi yang tidak hanya relevan tetapi juga berguna untuk membuat suatu prediksi.

### 2.2.3 Bidirectional Long Short-Term Memory (BiLSTM)

*Bidirectional Long Short-Term Memory* atau disingkat menjadi BiLSTM, merupakan variasi dari LSTM tradisional yang dirancang untuk meningkatkan kemampuan model dalam menangani informasi sekuensial. Arsitektur BiLSTM terdiri dari dua lapisan LSTM yang saling terintegrasi satu lapisan bergerak maju (*Forward pass*) dan satu lapisan lain bergerak mundur (*backward pass*)(Rhanoui et al., 2019). Hal ini memungkinkan model untuk memperoleh informasi baik sebelum maupun sesudah titik data saat ini dan memberikan perspektif yang lebih lengkap (Graves & Schmidhuber, 2005). Kedua lapisan tersebut beroperasi secara independen. Arsitektur BiLSTM dapat dilihat pada Gambar 2.8 (Cui et al., 2018).




Gambar 2.8 Arsitektur BiLSTM

$$\overrightarrow{h_t} = \overrightarrow{o_t} * \tanh\left(\overrightarrow{C_t}\right) \quad 2.10$$

$$\overleftarrow{h_t} = \overleftarrow{o_t} * \tanh\left(\overleftarrow{C_t}\right) \quad 2.11$$

$$h_t = \overrightarrow{h_t}\,;\, \overleftarrow{h_t} \quad 2.12$$

Secara garis besar rumus dari BiLSTM sama dengan rumus LSTM, yang membedakan BiLSTM dengan LSTM adalah dalam BiLSTM memiliki dua buah keluaran yaitu keluaran LSTM *forward* yang dapat dilihat pada persamaan 2.10 dan keluaran LSTM *backward* yang dapat dilihat pada persamaan 2.11. Keluaran dari dua buah LSTM tersebut kemudian digabungkan menjadi keluaran BiLSTM. Penggabungan biasanya dilakukan dengan meng

konkatenasi keluaran *forward* dan *backward* LSTM. Keluaran BiLSTM dapat dilihat pada persamaan 2.12.

### 2.2.4 Distilbert

Distilbert merupakan model versi ringkas dan efisien dari model BERT (*Bidirectional Encoder Representations from Transformers*). Dikembangkan oleh Sanh dkk., (2019), Distilbert mengadopsi teknik '*knowledge distillation*' untuk mengurangi ukuran model BERT asli tanpa mengorbankan kemampuan BERT dalam dalam memahami bahasa secara signifikan. Dalam penelitian tersebut model Distilbert berhasil mengurangi ukuran model BERT sebesar 40% namun tetap mempertahankan 97% dari kemampuan pemahaman bahasa model asli dan 60% lebih cepat.

Perbedaan utama antara DistilBERT dan BERT terletak pada arsitektur modelnya. DistilBERT menghilangkan beberapa *layers* dari BERT serta menghapuskan beberapa komponen seperti *token-type embeddings* dan *pooler*. Meskipun lebih kecil, DistilBERT dapat di-*tune* dengan baik dan memberikan performa yang kompetitif pada berbagai tugas pemrosesan bahasa alami dan menjadikannya pilihan yang cocok untuk aplikasi yang memiliki keterbatasan sumber daya komputasi. Hal ini juga memungkinkan penggunaan model yang lebih canggih dalam situasi dimana sumber daya komputasi yang terbatas. DistilBERT membuktikan bahwa efisiensi komputasi dapat dicapai tanpa mengorbankan kemampuan pemrosesan bahasa secara signifikan.

### 2.2.5 Stratified K-Fold Cross Validation

*Stratified K-Fold Cross Validation* merupakan pengembangan dari metode *Cross Validation* (CV) biasa, yang dirancang untuk menangani permasalahan distribusi kelas yang tidak seimbang dalam dataset. Dalam k-fold CV standar, data dibagi menjadi lipatan atau fold, dan model dilatih sebanyak k kali, dengan setiap fold digunakan sebagai set validasi sekali dan sisa data sebagai set pelatihan. Tetapi dalam Stratifies K-Fold CV, pembagian data ke dalam fold dilakukan sedemikian rupa sehingga mempertahankan distribusi kelas asli di semua fold. Artinya, setiap Fold mengandung proporsi masing-masing kelas yang kurang lebih sama dengan seluruh dataset, memastikan bahwa tidak ada kelas yang kurang maupun lebih.

Metode ini sangat penting untuk memvalidasi keakuratan sebuah model, terlebih untuk data dengan proporsi kelas yang tidak seimbang (Widodo dkk., 2022).. Penelitian lain juga

menekankan bahwa pentingnya distribusi kelas yang merata di setiap fold dalam Stratified K-Fold Cross Validation untuk memastikan tidak ada gangguan pada rasio distribusi sampel antar kelas, yang merupakan aspek kritis dalam menjaga kualitas validasi model, terutama dalam situasi ketidakseimbangan kelas dapat mempengaruhi performa atau hasil evaluasi model (Prusty et al., 2022).

### 2.2.6 RandomSearch

Random search merupakan salah satu metode optimasi *hyperparameter* untuk machine learning dengan mencoba beberapa kombinasi *hyperparameter* secara acak dari ruang pencarian yang ditentukan. Perbedaan di antara *Random search* dan *Grid search* adalah *Random search* memilih potensial parameter secara acak, sedangkan *Grid search* mengeksplorasi semua kombinasi secara sistematis. *Random search* mampu menemukan model yang sama baik atau lebih baik dari *Grid search* dengan waktu komputasi yang lebih singkat atau dapat dikatakan bahwa random search memiliki efisiensi lebih baik jika dibandingkan dengan *Grid search* (Bergstra et al., 2012). Hal ini juga didukung pada penelitian lagi yang juga membandingkan *Random search* dengan *Grid search*. Hasil mengatakan bahwa *Random search* menemukan model yang lebih baik dalam sebagian kasus dan memerlukan waktu komputasi yang lebih singkat (Larochelle et al., 2007). *Random search* juga lebih mudah dilakukan karena setiap percobaan dalam random search bersifat independen (Bergstra et al., 2012).

### 2.2.7 Evaluation Matrix

Dalam konteks penelitian yang terkait dengan Machine learning, perlu dilakukan tahapan evaluasi. Tahap evaluasi sangatlah penting dengan tujuan untuk mengukur dan mengevaluasi kinerja model (Fawcett, 2006). Salah satu evaluasi yang dapat dilakukan adalah menggunakan evaluation matrix. Evaluation matrix yang sering digunakan yaitu menggunakan Confusion Matrix.

Evaluasi ini penting dilakukan dalam Machine learning untuk menentukan sejauh mana model yang dibuat dapat diandalkan dan apakah perlu dilakukan penyesuaian maupun perbaikan terhadap model. Pemilihan matriks evaluasi yang tepat bergantung pada sifat karakteristik data dan model. Seperti contoh, dalam data yang memiliki persebaran data yang seimbang pada tiap label, maka dapat digunakan nilai accuracy, tetapi untuk data yang

persebaran datanya tidak seimbang pada tiap label, maka dapat menggunakan *F1-score* sebagai acuan.


Gambar 2.9 *Confusion Matrix*

Dalam Confusion matrix 2 kali 2, terdapat empat kemungkinan hasil yang dapat terjadi. Jika nilai yang diprediksi dan nilai aktualnya bernilai positif maka diklasifikasikan ke dalam *true positive* (TP), jika nilai yang diprediksi bernilai positif dan nilai aktualnya adalah negatif maka diklasifikasikan ke dalam *false positive* (FP), jika nilai yang diprediksi bernilai negatif dan nilai aktualnya bernilai positif maka dapat diklasifikasikan ke dalam *false negative* (FN), dan jika nilai prediksi dan actual sama-sama bernilai negatif maka dapat diklasifikasikan kedalam *True negative* (TN) (Tripathy et al., 2015).

*Confusion matrix* tersebut dapat digunakan untuk mengukur seberapa baik model memprediksi data dan biasanya meliputi matriks seperti *accuracy*, *precision*, *recall*, dan *F1-score*. Dengan masing-masing matriks dapat digunakan dalam kondisi tertentu.

*Accuracy* mengukur seberapa sering model membuat prediksi yang benar. Dihitung berdasarkan persentase kasus di mana prediksi yang dilakukan model cocok dengan label sebenarnya dari data. *Accuracy* tidak dapat digunakan sebagai acuan ketika kelas tidak seimbang. Persamaan *accuracy* dapat dilihat pada persamaan 3.1.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \quad 3.1$$

*Precision* mengukur seberapa akurat prediksi positif yang dibuat oleh model. Dihitung berdasarkan proporsi prediksi positif yang benar-benar positif atau dapat diartikan sebagai ukuran seberapa baik model menghindari pemberian label positif pada sampel negatif.

*Precision* sangat penting digunakan dalam situasi dimana konsekuensi dari *false positive* (FP) cukup besar. *Precision* dapat dihitung pada persamaan 3.2.

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.2}$$

*Recall* mengukur seberapa baik model mengidentifikasi semua kasus positif. Dihitung sebagai proporsi dari semua kasus positif yang sebenarnya yang berhasil diidentifikasi oleh model. *Recall* sangat penting digunakan untuk mengidentifikasi semua kasus positif dalam kasus medis. *Recall* dapat dihitung pada persamaan 3.3.

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.3}$$

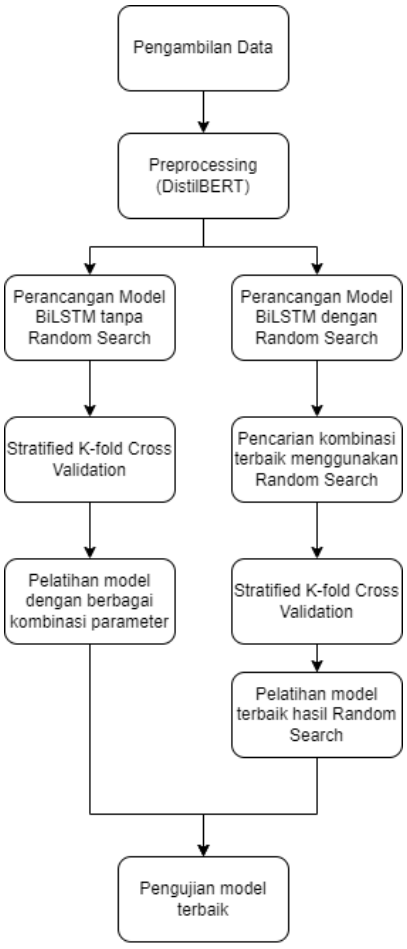
*F1-score* merupakan rata-rata harmonik dari *precision* dan *recall* dan memberikan keseimbangan antara kedua metrik tersebut. *F1-score* sangat berguna ketika kelas tidak seimbang. Persamaan *F1-score* dapat dilihat pada persamaan 3.4.

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.4}$$

# BAB III
# METODOLOGI PENELITIAN

## 3.1 Alur Penelitian

Langkah-langkah penelitian dalam analisis sentimen ulasan film menggunakan metode BiLSTM secara garis besar adalah pengambilan data, preprocessing merubah teks mentah menjadi vektor menggunakan DistilBert, perancangan model BiLSTM yang dibagi menjadi dua model, model dengan parameter yang sudah ditentukan atau tanpa hyper*tuning* parameter menggunakan Random search dan model dengan parameter yang dicari menggunakan Random Search, kemudian dilakukan Stratified K-fold Cross Validation untuk masing-masing model dan juga dilakukan evaluasi menggunakan confusion matrix serta pengujian yang dilakukan dengan menggunakan model paling baik dari segi akurasi. Diagram alur penelitian dapat dilihat pada Gambar 3.1.




Gambar 3.1 Alur Penelitian

## 3.2 Pengambilan Data

Penelitian ini tidak melakukan proses pengambilan data (*scraping*) secara mandiri. Penelitian ini menggunakan dataset yang telah dikumpulkan dan disusun berdasarkan penelitian sebelumnya yang dilakukan oleh Maas dkk., (2011) yang tersedia pada platform Hugging Face. Dataset ini dikenal sebagai "IMDB dataset" merupakan kumpulan ulasan film dari Internet Movie Database (IMDB). Keputusan ini diambil dengan pertimbangan penggunaan data set yang telah ada dapat memungkinkan efisiensi waktu yang signifikan, karena proses scraping dan pengolahan data awal dapat memakan banyak waktu, serta penelitian ini memiliki fokus utama bukan pada proses pengambilan data itu sendiri, tetapi lebih kepada penerapan dan analisis data tersebut.

Data set yang dipilih merupakan kumpulan ulasan film yang mencakup 50.000 ulasan film yang telah terstruktur dan seimbang. Lebih detail, data set ini memiliki batas maksimum 30 ulasan per film yang diperbolehkan. Hal tersebut dilakukan karena ulasan untuk film yang sama cenderung memiliki penilaian yang berkorelasi. Penentuan sentimen ditentukan berdasarkan rating yang diberikan pada ulasan di situs IMDB dengan rentang ulasan 1 sampai 10 dengan rating terbaik adalah rating 10. Sentimen negatif ditentukan dengan ulasan yang memiliki rating 1 sampai 4, sedangkan sentimen positif memiliki ulasan dengan rating 7 sampai 10. Kriteria ini digunakan oleh peneliti sebelumnya agar sentimen netral tidak tergabung dalam data set, sehingga meningkatkan kejelasan terhadap perbedaan sentimen negatif dan positif dan untuk menghindari bias. Data set ini telah dibagi menjadi dua bagian yaitu train dataset dan test dataset dengan masing-masing bagian memiliki 25000 ulasan dan proporsi sebaran sentimen yang merata.

Tabel 3.1 Variabel Data Set

| Variabel | Definisi |
|---|---|
| text | Data ulasan film |
| label | Sentimen yang telah diklasifikasikan berdasarkan kriteria nilai |

## 3.3 Preprocessing Text

Dalam penelitian ini, langkah-langkah preprocessing text biasa seperti case folding, penghapusan tag HTML, penghapusan tanda baca, penghapusan angka, penghapusan stopword, dan lemmatization tidak dilakukan karena menggunakan Distilbert untuk mengolah data teks hingga menjadi vektor. Alasannya terletak pada kemampuan yang dimiliki oleh

DistilBERT sebagai model yang berbasis pada BERT. Distilbert telah dilatih pada data set yang luas dan bervariasi yang sudah termasuk banyak aspek linguistik.

Distilbert memiliki efisiensi untuk memproses teks dengan semua elemennya, termasuk kapitalisasi huruf, tanda baca, dan angka, yang sering kali membawa makna penting. Penghapusan elemen-elemen tersebut dapat menghilangkan informasi penting yang terkandung yang berkontribusi pada pemahaman konteks kalimat. Selain itu, kata yang termasuk kedalam stopword juga dapat berperan penting dalam pemahaman konteks kalimat, terutama analisis sentimen. Proses Lemmatization juga tidak terlalu kritis karena DistilBERT mampu memahami berbagai bentuk kata.

Proses yang dilakukan pada tahap ini pertama, teks mentah diubah menjadi token menggunakan tokenizer DistilBERT. Tokenizer ini bertugas untuk memecah teks menjadi potongan yang lebih kecil, seperti kata atau sub-kata. Setelah teks dilakukan tokenize, setiap token kemudian dikonversi menjadi vektor numerik menggunakan model DistilBERT. Distilbert akan mengambil setiap token dan menghasilkan embedding kata, yang merupakan representasi vektor dari setiap token tersebut. Perubahan ini dilakukan agar teks juga dapat diolah oleh model BiLSTM untuk memproses dan memahami teks dengan lebih efektif.

### 3.4 Model BiLSTM

Setelah data set ulasan film melalui tahap preprocessing teks menggunakan DistilBERT, maka dilakukan proses pemodelan menggunakan BiLSTM (Bidirectional Long Short-Term Memory) yang dilakukan dengan dua pendekatan berbeda untuk menentukan parameter model. Kedua pendekatan memiliki struktur yang sama namun berbeda dalam menentukan parameter model.

1. Model BiLSTM Tanpa Menggunakan Random Search

   Dalam pendekatan pertama, parameter model BiLSTM ditetapkan secara manual dengan memilih beberapa kombinasi yang mungkin muncul untuk parameter seperti unit dalam lapisan LSTM, tingkat Dropout, dan unit dalam lapisan Dense berdasarkan asumsi kombinasi parameter yang mungkin bekerja paling baik untuk data set. Rancangan arsitektur model dapat dilihat pada Tabel 3.2.

Tabel 3.2 Rancangan arsitektur model tanpa menggunakan Random Search

| **No** | ***Layer*** | **Model BiLSTM tanpa *Hyperparameter Tuning*** |
|---|---|---|
| 1 | Embedding | Embedding( input_dim=tokenizer.vocab_size, *output*_dim=embedding_*layer*.dim, trainable=False |
| 2 | BiLSTM | Bidirectional(LSTM()) |
| 3 | Dropout | Dropout() |
| 4 | Dense | Dense() |
| 5 | Dropout | Dropout() |
| 6 | Dense (*Output*) | Dense(1, activation='sigmoid') |

Kombinasi parameter yang dilakukan pengujian pada tiap lapisan LSTM, tingkat Dropout, dan unit dalam lapisan Dense. Model tersebut selanjutnya akan disebut sebagai model_nhpt1, model_nhpt2, model_nhpt3, dan model_nhpt4. Kombinasi parameter dapat dilihat pada Tabel 3.3.

Tabel 3.3 Kombinasi parameter yang ditentukan sendiri

| ***Layer*** | Model_nhpt1 | Model_nhpt2 | Model_nhpt3 | Model_nhpt4 |
|---|---|---|---|---|
| LSTM | 64 | 32 | 96 | 128 |
| Dropout_1 | 0.5 | 0.4 | 0.3 | 0.4 |
| Dense | 64 | 128 | 96 | 32 |
| Dropout_2 | 0.5 | 0.2 | 0.4 | 0.3 |

2. Model BiLSTM dengan Pencarian Parameter Menggunakan Random Search

   Pendekatan kedua, parameter model BiLSTM ditetapkan menggunakan metode Random Search untuk menentukan parameter terbaik. Random Search memilih nilai acak parameter berdasarkan rentang yang sudah ditentukan dan mengevaluasi kinerja model dengan setiap kombinasi parameter. Tujuan dari pendekatan ini untuk mengeksplorasi parameter secara lebih luas dan potensial menemukan kombinasi parameter yang memberikan kinerja terbaik, tanpa dapat mempengaruhi hasil model. Pendekatan tidak memerlukan pengetahuan mendalam mengenai parameter. Rancangan arsitektur dan perbedaan kedua pendekatan dapat dilihat pada Tabel 3.4.

Tabel 3.4 Rancangan arsitektur model dengan menggunakan Random Search

| **No** | ***Layer*** | **Model BiLSTM dengan *Hyperparameter Tuning*** |
|---|---|---|
| 1 | Embedding | Embedding( input_dim=tokenizer.vocab_size, *output*_dim=embedding_*layer*.dim, trainable=False |
| 2 | BiLSTM | Bidirectional(LSTM(hp.Int('lstm_units', min_value=32, max_value=128, step=32))) |
| 3 | Dropout | Dropout(hp.Float('dropout_1', min_value=0.2, max_value=0.5, step=0.1)) |
| 4 | Dense | Dense(hp.Int('dense_units', min_value=32, max_value=128, step=32)) |
| 5 | Dropout | Dropout(hp.Float('dropout_2', min_value=0.2, max_value=0.5, step=0.1)) |
| 6 | Dense (*Output*) | Dense(1, activation='sigmoid') |

Deskripsi *layer* :

1. Embedding *Layer*: Membuat representasi vektor untuk setiap kata. Pada arsitektur diatas kedua model menggunakan bobot dari DistilBERT, yang tidak dilatih (trainable=False), untuk memanfaatkan pemahaman bahasa yang sudah dipelajari oleh DistilBERT.
2. Bidirectional LSTM: Memproses teks dengan LSTM yang beroperasi dalam dua arah untuk menangkap konteks.
3. Dropout: Mencegah *overfitting* dengan secara acak ’mematikan‘ beberapa neuron.
4. Dense *layer*: Lapisan yang terhubung penuh (fully connected) untuk ekstraksi fitur lanjutan atau klasifikasi.
5. Dense *layer* (*output*): Menghasilkan *output* akhir. Klasifikasi biner menggunakan aktivasi ‘Sigmoid‘

Untuk konfigurasi kompilasi model yang digunakan dalam model ini adalah memilih optimizer Adam dengan learning rate sebesar $1 \times 10^{-4}$ atau 0.0001, fungsi loss menggunakan binary_crossentropy . Setiap model juga akan dilatih sebanyak 15 epocs.

### 3.5 Stratified K-Fold Cross Validation

Stratified K-fold cross validation (SKCV) digunakan dalam penelitian ini untuk memvalidasi dan mengevaluasi model BiLSTM, baik model tanpa *hyperparameter tuning* maupun model dengan *hyperparameter tuning* menggunakan Random Search. SKCV digunakan untuk memastikan bahwa model yang dikembangkan mampu berperforma secara konsisten dan akurat di berbagai sampel data.

Pada model dengan pendekatan model dengan parameter yang ditentukan sendiri, SKCV diterapkan langsung. Ini berarti setelah model dibangun dengan parameter yang telah ditentukkan, SKCV digunakan untuk membagi dataset menjadi beberapa lipatan, dimana setiap lipatan memiliki distribusi kelas yang seimbang. Model tersebut kemudian dilatih dan diuji pada setiap lipatan.

Sementara model dengan pendekatan menggunakan Random Search proses SKCV dilakukan setelah Random Search menemukan konfigurasi terbaik dari model. Setelah model terbaik ditemukan, SKCV kemudian diterapkan sama seperti penerapan pada pendekatan model dengan parameter yang ditentukan dengan mengkloning model dengan konfigurasi terbaik.

Pada tahap ini, digunakan keseluruhan dari data set yaitu sebesar 50.000 ulasan film yang akan dibagi menjadi 5 *fold* atau lipatan. Setiap *fold* akan mendapatkan distribusi sentimen yang sama dengan data set asli, sehingga tiap *fold* akan memiliki 5000 ulasan positif dan 5000 ulasan negatif. Selama tahap ini, satu dari lima *fold* digunakan sebagai *data test* dan empat fold lainnya menjadi *data train*, sehingga perbandingan *data train* dan *data test* adalah 40000 : 10000 ulasan.

Tahap evaluasi kemudian dilakukan dengan menggunakan confusion matrix yang penjelasan lebih lengkap dapat dilihat pada sub bab 2.2.7. Dari matrix tersebut dapat diperoleh nilai accuracy, *recall*, *precision*, dan *F1-score* dengan penjelasan dan persamaan dapat dilihat dalam sub bab 2.2.7. Pengujian model juga dilakukan dengan memberikan kalimat baru yang kemudian dilakukan pengujian menggunakan model yang sudah disimpan.

### 3.6 Pengujian

Setelah proses validasi menggunakan Stratified K-fold Cross Validation (SKCV) pada kedua pendekatan model, pengujian dilakukan dengan model yang memiliki akurasi terbaik dari kedua pendekatan. Model terpilih kemudian digunakan untuk melakukan prediksi sentimen pada kata-kata atau frasa yang diinputkan.

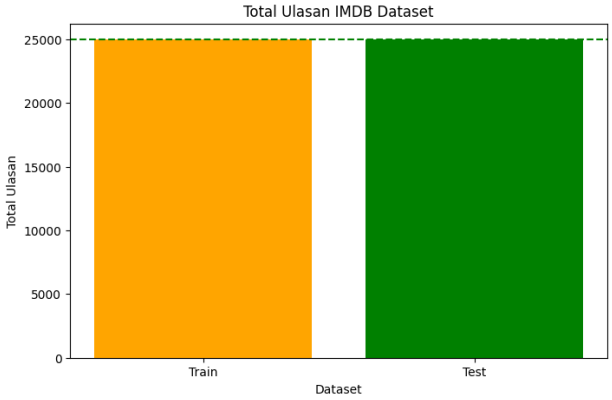
Dalam tahap pengujian, model akan menganalisis teks input yang dapat berupa kalimat dari ulasan film dan kemudian memprediksi sentimennya, apakah positif atau negatif. Proses ini diawali dengan mengubah teks input menjadi vektor dengan DistilBERT. Vektor tersebut kemudian diproses oleh model BiLSTM untuk diprediksi sentimennya.

Tujuan utama dari pengujian ini adalah untuk mengevaluasi seberapa baik model dapat memahami dan menafsirkan nuansa dalam teks ulasan film serta mengklasifikasikannya ke dalam kategori sentimen yang relevan. Hasil pengujian ini memberikan wawasan penting mengenai pengaplikasian model dalam skenario dunia nyata dan efektivitasnya dalam memahami sentimen yang terkandung dalam teks.

# BAB IV
# HASIL DAN PEMBAHASAN

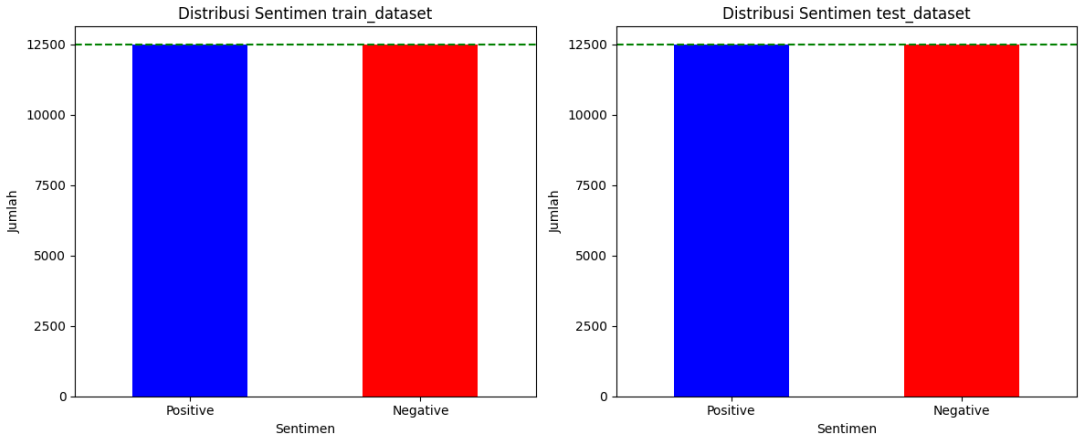
## 4.1 Data set

Data set yang digunakan dalam penelitian ini terdiri dari ulasan film yang telah dikumpulkan pada penelitian sebelumnya dengan detail yang sudah dijelaskan pada sub bab 3.2. Total ulasan secara keseluruhan dalam data set sebanyak 50.000 ulasan yang dibagi dalam data train dan data test. Grafik persebaran data dapat dilihat pada Gambar 4.1.




Gambar 4.1 Persebaran dataset

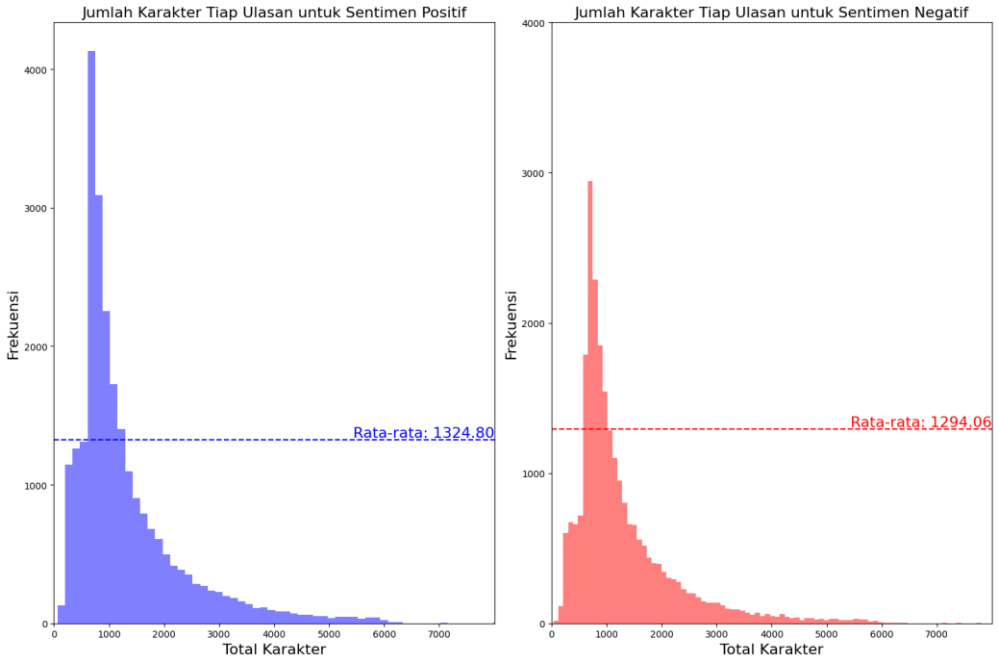
Dalam tiap data train maupun data test, dapat dikategorikan kembali berdasarkan label sentimen. Pada Gambar 4.2 dapat dilihat bahwa baik train_dataset dan test_dataset memiliki sebaran tiap sentimen dengan jumlah yang sama sebanyak 12.500 ulasan tiap sentimen tiap datasetnya.




Gambar 4.2 Distribusi sentimen tiap data set

Setelah dilakukan eksplorasi lanjutan, ditemukan bahwa terdapat beberapa ulasan yang terduplikasi dalam data set. Duplikasi ini dapat diinterpretasikan sebagai bagian dari proses *balancing* atau penyeimbangan data, dimana beberapa ulasan disertakan lebih dari sekali untuk memastikan distribusi merata antara sentimen positif dan sentimen negatif. Keseimbangan data antara kedua sentimen juga memberikan keuntungan dalam hal analisis dan pemodelan, karena memastikan bahwa model tidak akan bias terhadap satu sentimen tertentu karena ketidakseimbangan dalam data.

Kemudian dilakukan analisis panjang karakter setiap sentimennya. Panjang rata-rata karakter untuk ulasan positif adalah sekitar 1324.80 karakter, sedangkan untuk ulasan negatif adalah sekitar 1294.06 karakter. Ini menunjukkan bahwa individu cenderung menulis ulasan positif dengan panjang sedikit lebih banyak dibandingkan dengan ulasan negatif. Walaupun perbedaan tidak terlalu signifikan, hal ini bisa mengindikasikan bahwa ketika orang memiliki pengalaman positif, mereka cenderung lebih termotivasi untuk berbagi detail lebih banyak atau mengekspresikan lebih luas. Sebaliknya, ulasan negatif meskipun tetap informatif, tetapi cenderung lebih dituliskan secara langsung dan singkat. Untuk grafik dapat dilihat pada Gambar 4.3




Gambar 4.3 Frekuensi Panjang Karakter Tiap Ulasan Sebelum *Preprocessing*

## 4.2 Preprocessing Text

Pada tahap ini dilakukan perubahan teks mentah menjadi vektor dengan menggunakan DistilBERT. Proses ini diawali dengan tokenisasi menggunakan pre-trained ‘distilbert-base-uncased’. Tokenizer ini bertugas mengubah teks mentah menjadi token yang kemudian diubah menjadi ID token berdasarkan kamus internal model DistilBERT yang merupakan representasi numerik dari setiap token atau kata dalam teks. Pseudocode dapat dilihat pada Gambar 4.4.

```
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")
```

Gambar 4.4 Pseudocode tokenizer

Dalam fungsi ‘preprocess’, diterapkan padding hingga panjang maksimum 512 token dan melakukan truncation jika panjang teks melebihi batas maksimum. Padding memastikan bahwa semua input memiliki panjang yang sama, sementara truncation mencegah kelebihan informasi yang tidak diperlukan.

```
def preprocess(data):
   return tokenizer(data['text'], padding="max_length",
                    truncation=True, max_length=512)
```

Gambar 4.5 Pseudocode fungsi preprocess

Kemudian fungsi ‘preprocess’ tersebut diimplementasikan dalam kolom baru bernama ‘processed’ yang berisi struktur data yang dihasilkan oleh tokenizer. Struktur tersebut termasuk ‘input_ids’ yang merupakan representasi numerik dari teks asli. Kemudian dibuat array ‘combined_input_ids’ yang merupakan kumpulan dari semua ‘input_ids’ yang dihasilkan dari setiap baris dalam dataset.

```
combined_df['processed'] = combined_df.apply(lambda x: preprocess(x),
axis=1)

combined_input_ids = np.array([x['input_ids'] for x in
combined_df['processed']])
```

Gambar 4.6 Pseudocode implementasi fungsi preprocess

## 4.3 Pembuatan Model BiLSTM

### 4.3.1 Model tanpa *hyperparameter tuning*

Pada tahap ini, model memiliki parameter yang sudah ditentukan sebelumnya, seperti yang sudah dijelaskan pada sub bab 3.4. Untuk implementasi arsitektur BiLSTM dengan parameter yang ditentukan dapat dilihat pada Gambar 4.7.

```
model = Sequential()
model.add(Embedding(input_dim=tokenizer.vocab_size,
                    output_dim=embedding_layer.dim,
                    weights=[embedding_layer.weights[0]],
                    trainable=False))
model.add(Bidirectional(LSTM()))
model.add(Dropout())
model.add(Dense())
model.add(Dropout())
model.add(Dense(1, activation='sigmoid'))
```

Gambar 4.7 Pseudocode implementasi model tanpa *hyperparameter tuning*

Kemudian setelah arsitektur model terbentuk dilakukan ‘model.compile’ dengan ketentuan optimizer menggunakan ‘Adam’ atau adaptive moment estimation, loss function menggunakan ‘binary_crossentropy’ karena model melakukan klasifikasi biner (positif (1) dan negatif (0)) dan metrik yang digunakan untuk mengevaluasi model adalah metrik ‘Accuracy’.

```
model.compile(optimizer='adam', loss='binary_crossentropy',
              metrics=['accuracy'])
```

Gambar 4.8 Pseudocode compile model tanpa *hyperparameter tuning*

### 4.3.2 Model dengan *hyperparameter tuning* menggunakan random search

Model dengan pendekatan *hyperparameter* pada penelitian ini menggunakan random search untuk menentukan set parameter terbaik. Untuk implementasi model yang digunakan dapat dilihat pada Gambar 4.9.

```
def BiLSTM_model(hp):
    model = Sequential()
```

```
model.add(Embedding(input_dim=tokenizer.vocab_size,
                    output_dim=embedding_layer.dim,
                     weights=[embedding_layer.weights[0]],
                     trainable=False))
model.add(Bidirectional(LSTM(hp.Int('lstm_units', min_value=32,
                                    max_value=128, step=32))))
model.add(Dropout(hp.Float('dropout_1', min_value=0.2,
                           max_value=0.5, step=0.1)))
model.add(Dense(hp.Int('dense_units', min_value=32,
                       max_value=128, step=32)))
model.add(Dropout(hp.Float('dropout_2', min_value=0.2,
                           max_value=0.5, step=0.1)))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy',
           metrics=['accuracy'])
return model
```

Gambar 4.9 Pseudocode implementasi model dengan menggunakan *hyperparameter tuning*

Pada pendekatan ini secara garis besar model memiliki arsitektur *layer* yang sama dengan model tanpa *hyperparameter tuning* dengan compile model yang sama juga. Perbedaan diantara keduanya terletak pada model arsitektur model yang dibuat menjadi fungsi dengan nama 'BiLSTM_model' yang membuat model dapat dipanggil kembali dengan menggunakan parameter yang berbeda. Perbedaan lainnya yaitu beberapa parameter pada 'BiLSTM_model' ditetapkan secara dinamis dan dilakukan pencarian *hyperparameter* menggunakan Random Search.

Untuk inisialisasi tuner, digunakan Random Search dari library keras_tuner. Implementasi dari inisialisasi tuner dapat dilihat pada Gambar 4.10.

```
tuner = kt.RandomSearch(BiLSTM_model,
            objective='val_accuracy',
            max_trials=10,
                executions_per_trial=1,
                directory= tuner_directory,
                project_name='bilstm')
```

Gambar 4.10 Pseudocode implementasi *hyperparameter tuning* menggunakan *Random search*

Penjelasan dari pseudocode, ‘BiLSTM_model’ memanggil fungsi model yang akan dilakukan *hyperparameter tuning*. ‘Objective=val_accuracy’ mendefinisikan bahwa tujuan dari pencarian ini dimaksimalkan pada akurasi pada set validasi dengan mencari akurasi validasi tertinggi. ‘Max_trials=10’ mendefinisikan bahwa *Random search* akan mencari kombinasi sebanyak sepuluh kali percobaan, dengan setiap kombinasi *hyperparameter* unik diuji. Setiap percobaan akan diuji sebanyak maksimal satu kali yang diindikasikan dengan ‘executions_per_trials=1’. Untuk penyimpanan log dan hasil pencarian dilakukan dengan penggunaan parameter ‘directory’ dan ‘project_name’.

Kemudian tuner akan melakukan pencarian *hyperparameter* dengan menjalankan fungsi ‘tuner.search’. Fungsi tersebut dijalankan dengan parameter ‘train_input_ids’ dan ‘train_labels’ sebagai data pelatihan, serta ‘test_input_ids’ dan ‘test_labels’ digunakan sebagai data validasi. Kemudian model akan dijalankan sebanyak 15 *epoch* untuk setiap kombinasi parameter.

```
tuner.search(train_input_ids, train_labels, epochs=15,
        validation_data=(test_input_ids, test_labels))
```

Gambar 4.11 Pseudocode inisiasi tuner.search

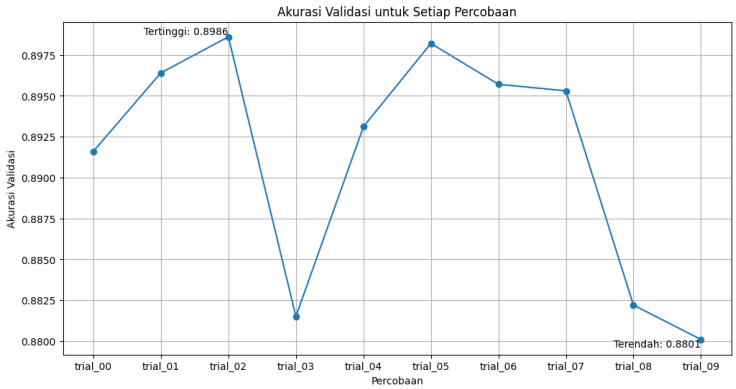
Setelah proses pencarian selesai, kerastuner akan menyimpan seluruh percobaan yang dilakukan ke dalam directory yang ditetapkan di awal. Hasil dari *hyperparameter tuning* menggunakan Random Search dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil *hyperparameter tuning* menggunakan Random search

| No | Trial | LSTM_units | Dropout_1 | Dense_units | Dropout_2 | Val_accuracy |
|---|---|---|---|---|---|---|
| 1 | trial_00 | 64 | 0.2 | 64 | 0.2 | 0.8916400074958801 |
| 2 | trial_01 | 64 | 0.3 | 64 | 0.3 | 0.8964400291442871 |
| 3 | trial_02 | 128 | 0.2 | 128 | 0.2 | 0.898639976978302 |
| 4 | trial_03 | 96 | 0.3 | 96 | 0.4 | 0.8814799785614014 |
| 5 | trial_04 | 32 | 0.3 | 96 | 0.2 | 0.8930799961090088 |
| 6 | trial_05 | 96 | 0.4 | 128 | 0.2 | 0.8982399702072144 |
| 7 | trial_06 | 128 | 0.3 | 128 | 0.4 | 0.8957200050354004 |
| 8 | trial_07 | 96 | 0.2 | 128 | 0.3 | 0.8953199982643127 |
| 9 | trial_08 | 32 | 0.4 | 128 | 0.2 | 0.8821600079536438 |
| 10 | trial_09 | 32 | 0.2 | 32 | 0.4 | 0.8801199793815613 |

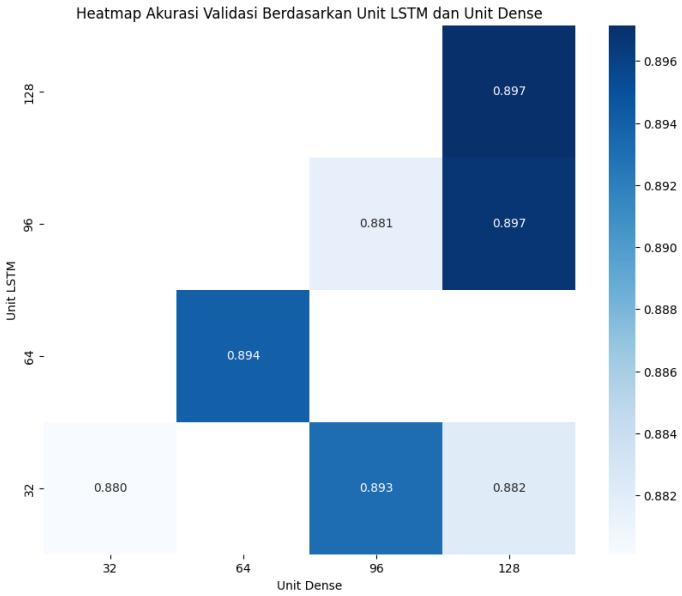
Pada Tabel 4.1, dapat dilihat bahwa tabel menunjukkan hasil dari 10 kali percobaan *tuning* dengan berbagai kombinasi *hyperparameter* pada model BiLSTM dengan setiap

percobaan yang dilakukan terdapat variasi yang luas dalam jumlah unit LSTM, tingkat dropout, dan jumlah unit pada lapisan dense.




Gambar 4.12 Akurasi validasi tiap percobaan *hyperparameter tuning*

Analisis lebih lanjut tampak bahwa model dengan jumlah unit LSTM dan unit dense yang lebih tinggi cenderung menghasilkan akurasi validasi yang lebih baik. Terlihat pada percobaan dengan 128-unit LSTM dan 128-unit dense seperti pada trial_02 dan trial_06 memiliki akurasi validasi yang cenderung mencapai nilai tertinggi. Berbanding terbalik dengan percobaan dengan 32-unit LSTM dan 32-unit dense pada trial_09 yang memiliki akurasi paling rendah. Ini mengindikasikan bahwa jumlah unit LSTM dan unit dense yang lebih besar pada model mungkin memberikan kapasitas yang lebih besar bagi model untuk mempelajari pola yang kompleks dari data, sehingga meningkatkan akurasi.




Gambar 4.13 Heatmap akurasi validasi berdasarkan unit LSTM dan unit Dense

Sementara pada dropout, tidak menunjukkan pola yang konsisten terkait peningkatan validasi akurasi. Ini menunjukkan bahwa efek dari tingkat dropout dapat bervariasi tergantung pada konfigurasi lain dari model, dan tidak selalu berkontribusi langsung terhadap peningkatan akurasi validasi.

Secara keseluruhan, hasil *hyperparameter tuning* menunjukkan akurasi antara 88% hingga mendekati angka 89%. Ini juga menunjukkan bahwa pencarian *hyperparameter* telah berhasil dalam mencari konfigurasi yang menghasilkan kinerja yang baik. Temuan ini menunjukkan bahwa pentingnya memilih dan menyesuaikan *hyperparameter* yang tepat dalam pengembangan model BiLSTM. Hasil tersebut juga didapatkan model pada trial_02 memiliki akurasi validasi terbaik dengan nilai 89,86% dengan kombinasi 128 unit LSTM, 0,2 pada tingkatan dropout_1, 128 unit dense, dan 0,2 pada tingkatan dropout_2.

## 4.4 Stratified K-Fold Cross Validation (SKCV)

Proses Stratified K-Fold Cross Validation untuk kedua pendekatan baik model tanpa menggunakan *hyperparameter tuning* maupun model dengan *hyperparameter tuning* memiliki arsitektur SKCV yang sama untuk memastikan evaluasi yang konsisten. Arsitektur utama pada Stratified K-fold cross validation dapat dilihat pada Gambar 4.14.

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

for train_index, test_index in skf.split(combined_input_ids,
                                         combined_labels):
    X_train_fold, X_test_fold = combined_input_ids[train_index],
                                combined_input_ids[test_index]
    y_train_fold, y_test_fold = combined_labels[train_index],
                                combined_labels[test_index]
```

Gambar 4.14 Potongan arsitektur stratified k-fold cross validation

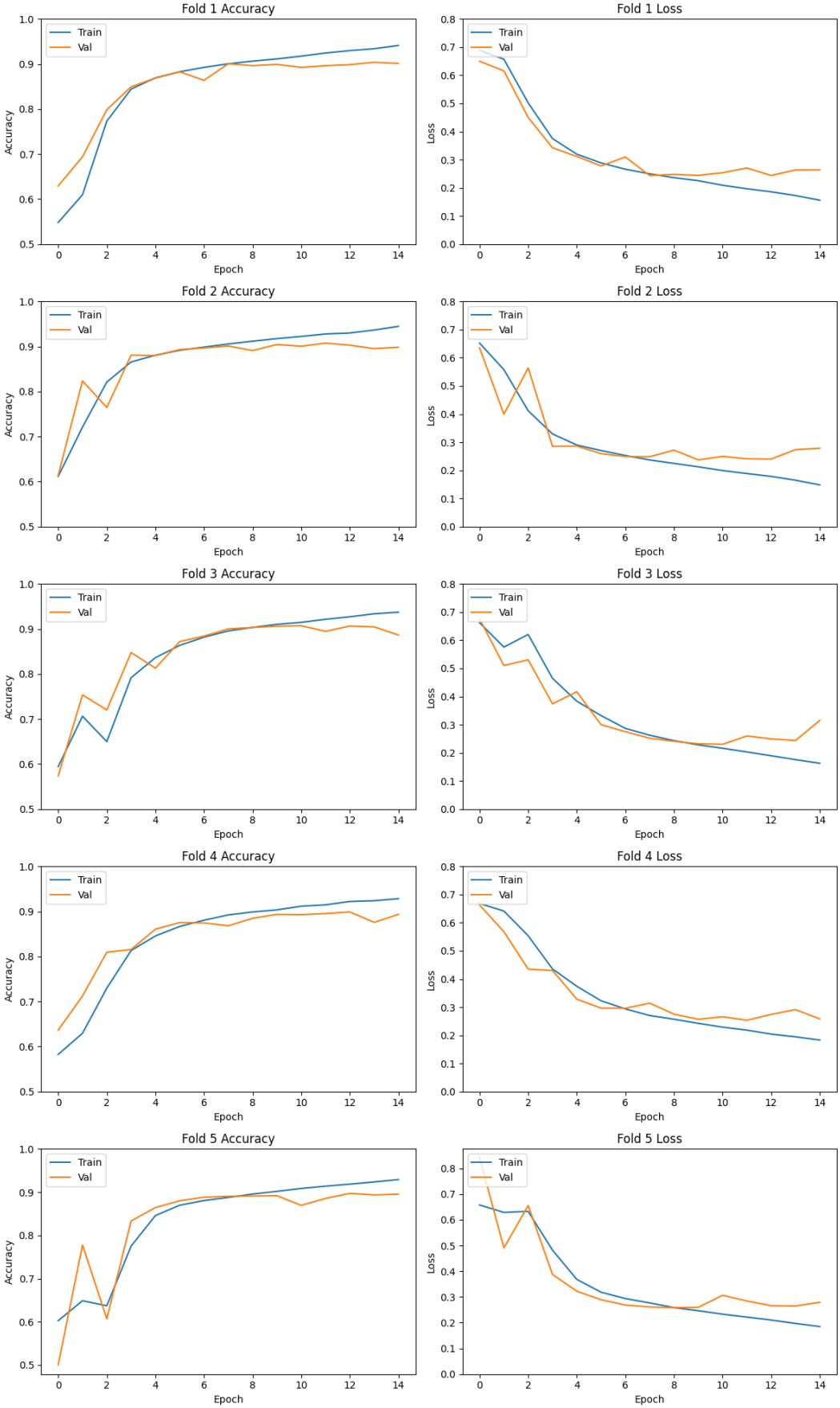
SKCV diinisialisasi dengan menggunakan ‘StratifiedKFold’ dengan parameter ‘n_splits=6’ untuk menentukan jumlah lipatan (folds) dalam SKCV sebanyak 6 bagian/lipatan terpisah. Ketika satu fold digunakan sebagai set pengujian, maka fold lain digunakan untuk set pelatihan. Pembagian ini memungkinkan model untuk dilakukan pelatihan dan pengujian pada berbagai kombinasi dataset. Kemudian ‘shuffle=True’ digunakan untuk memastikan bahwa data di ‘shuffle’ secara acak sebelum dibagi menjadi

lipatan. Parameter ‘random_state=42’ ini memastikan bahwa hasil pengacakan dapat direproduksi/dapat diulang kembali dalam berbagai eksperimen berulang. Dalam iterasi, dataset dibagi menjadi set pelatihan dan set pengujian untuk setiap lipatan (fold), menggunakan indeks dari ‘skf.split(combined_input_ids, combined_labels)’. ‘X_train_fold’ dan ‘X_test_fold’ mengandung input IDs, sedangkan ‘y_train_fold’ dan ‘y_test_fold’ berisi label yang sesuai.

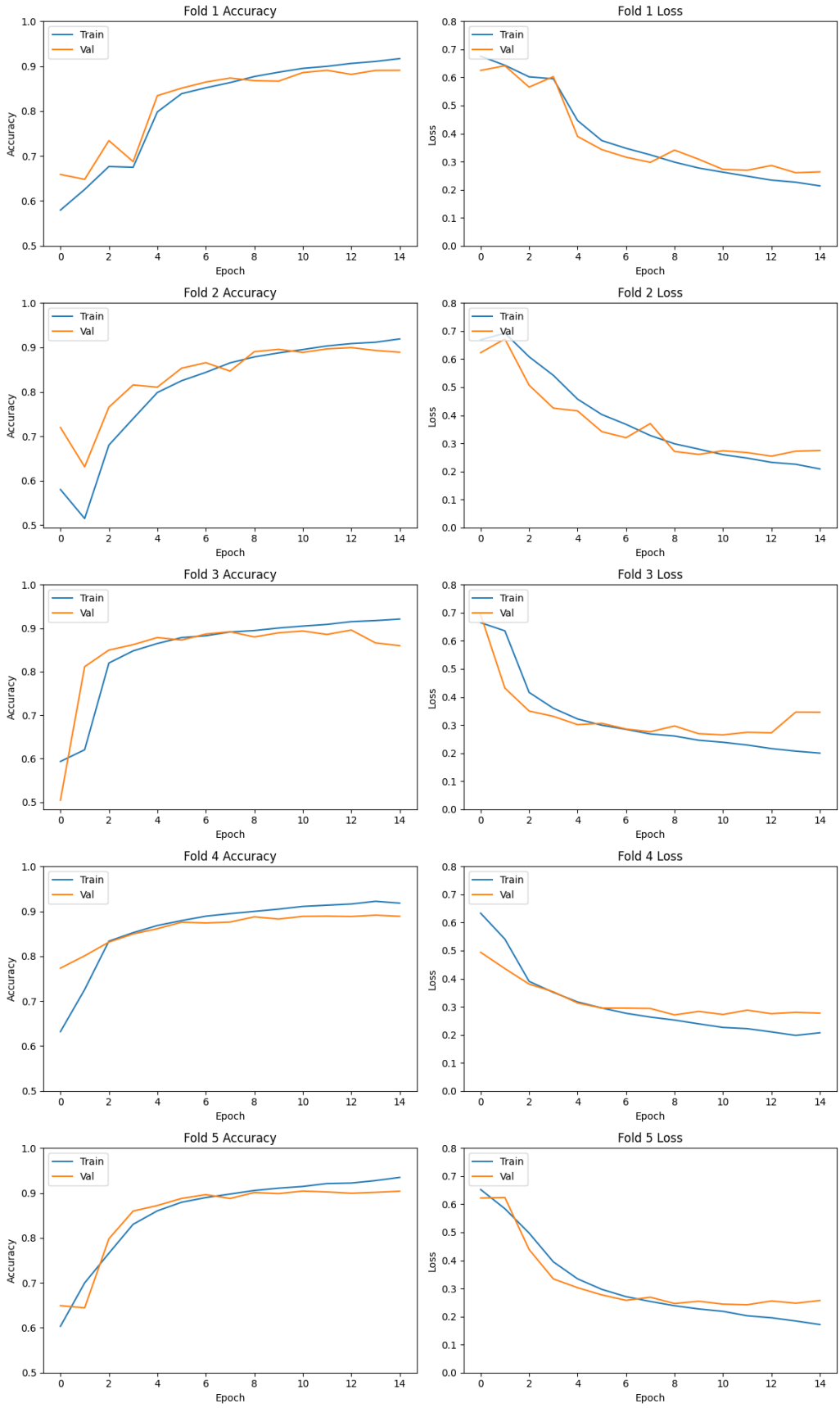
Dalam iterasi tersebut kedua model sama-sama dilatih pada set pelatihan (‘X_train_fold’, ‘y_train_fold’) dan validasi dilakukan pada set pengujian (‘X_test_fold’, ‘y_test_fold’) untuk 15 epochs, yang mana riwayat pelatihan disimpan dalam ‘history’ untuk analisis kinerja model. Setelah pelatihan, model dievaluasi menggunakan skor akurasi yang akan dihitung rata-rata akurasi keseluruhan fold.

### 4.4.1 Hasil SKCV untuk model tanpa menggunakan *hyperparameter tuning*

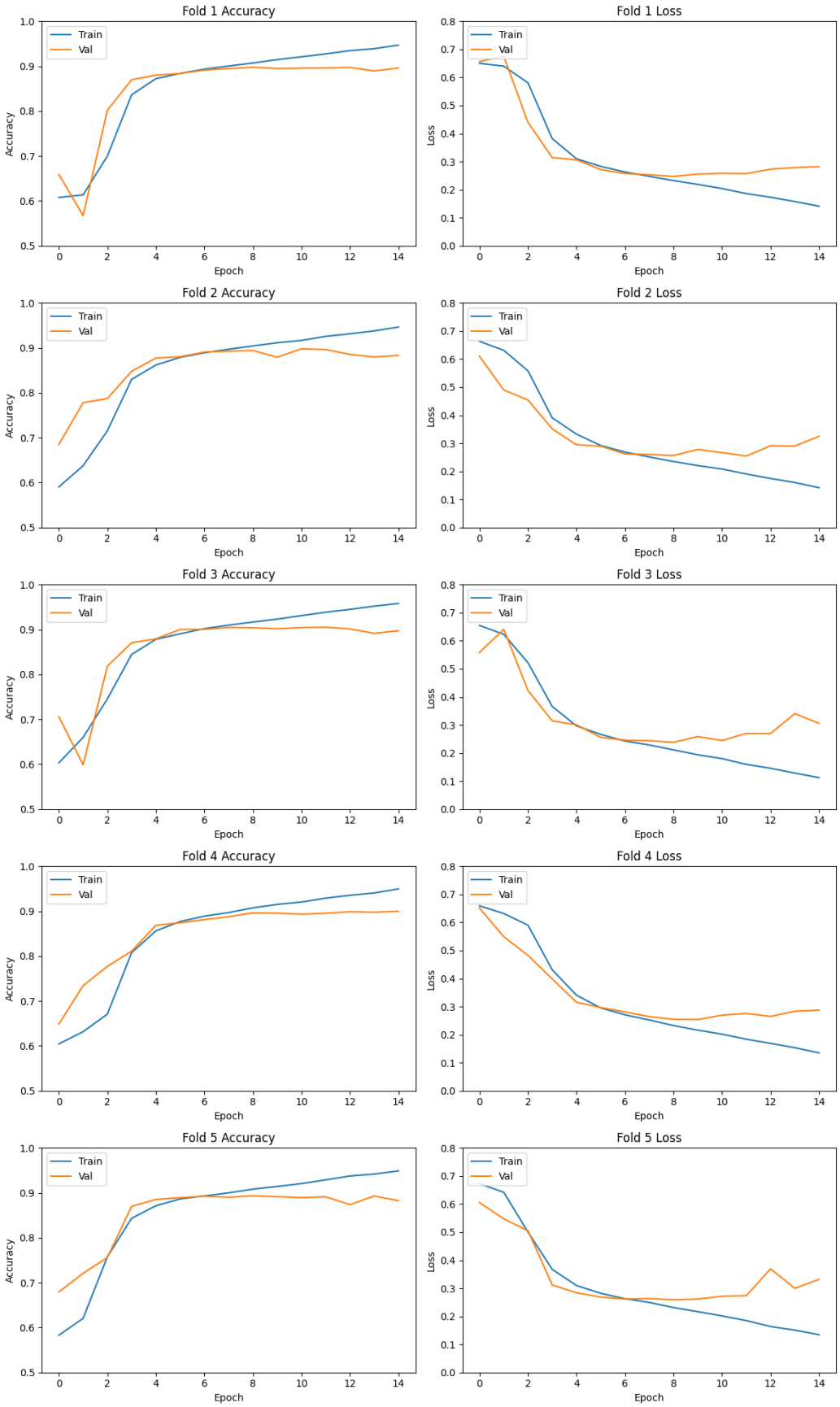
Setelah arsitektur model dibuat dengan parameter yang ditentukan dengan detail ada pada sub bab 4.3.1, model kemudian dilakukan SKCV. Hasil dari proses SKCV tersebut yang berupa grafik learning curve pada tiap foldnya dapat dilihat pada Gambar 4.15, Gambar 4.16, Gambar 4.17, dan Gambar 4.18.



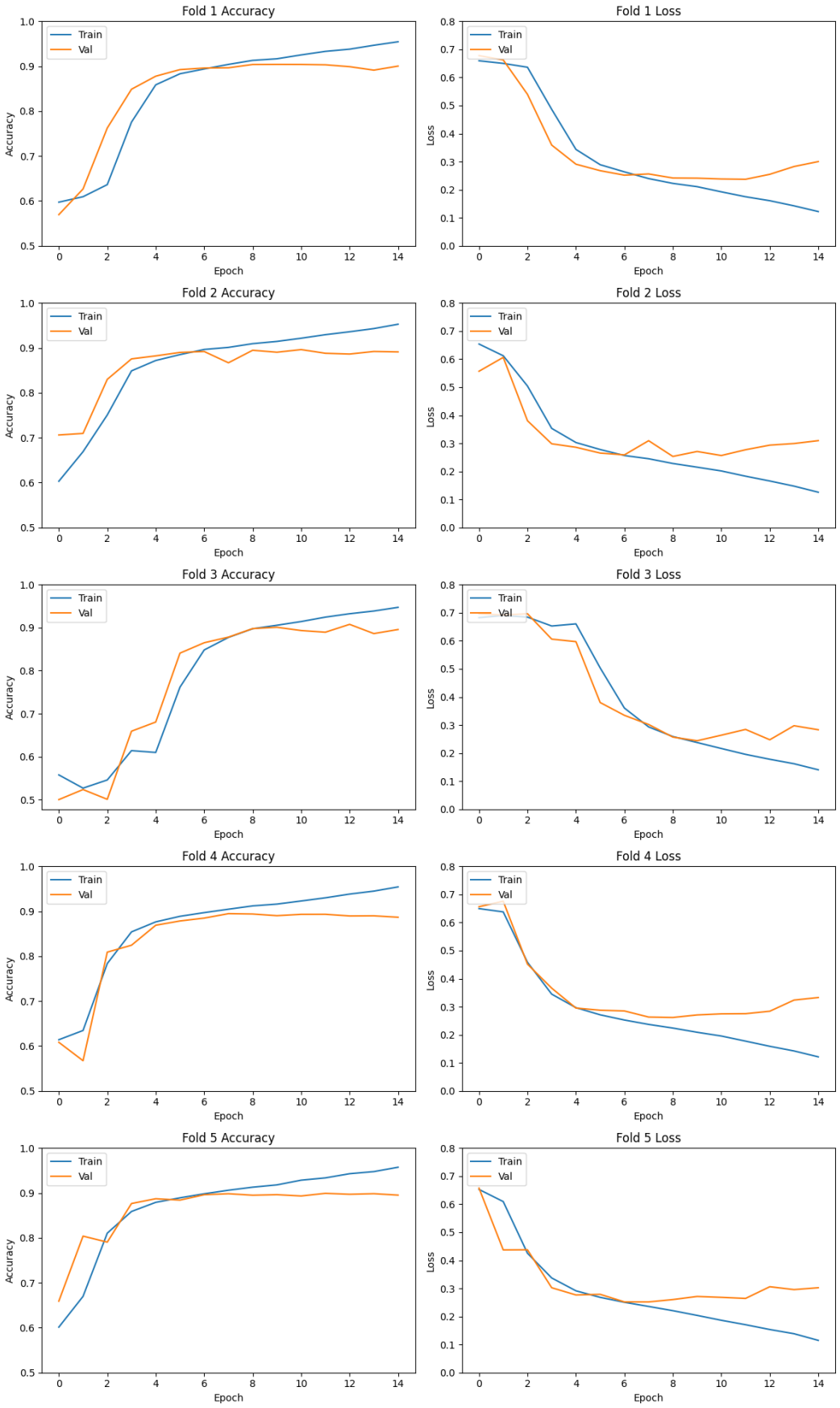

Gambar 4.15 Learning curve tiap fold untuk model_nhpt1




Gambar 4.16 Learning curve tiap fold untuk model_nhpt2




Gambar 4.17 Learning curve tiap fold untuk model_nhpt3




Gambar 4.18 Learning curve tiap fold untuk model_nhpt4

Berdasarkan *learning curve* diatas, secara umum *learning curve* dari ke empat model yang dilatih dengan stratified k-fold cross validation sebanyak 5-fold menunjukkan beberapa pola yang konsisten yang mengindikasikan bahwa terdapat *osilasi*, dimana model menunjukkan fluktuasi selama proses pelatihan.

Untuk menilai kinerja model, maka dilakukan evaluasi model menggunakan confusion matrix untuk setiap *fold*nya dan disajikan pada Tabel 4.2, Tabel 4.3, Tabel 4.4, dan Tabel 4.5.

Tabel 4.2 Hasil confusion matrix model_nhpt1

| ***Fold*** | ***True Positive*** | ***False Positive*** | ***True Negative*** | ***False Negative*** |
|---|---|---|---|---|
| Fold 1 | 4356 | 343 | 4657 | 644 |
| Fold 2 | 4694 | 710 | 4290 | 306 |
| Fold 3 | 4098 | 231 | 4769 | 902 |
| Fold 4 | 4374 | 437 | 4563 | 626 |
| Fold 5 | 4472 | 518 | 4482 | 528 |

Tabel 4.3 Hasil confusion matrix model_nhpt2

| ***Fold*** | ***True Positive*** | ***False Positive*** | ***True Negative*** | ***False Negative*** |
|---|---|---|---|---|
| Fold 1 | 4350 | 441 | 4559 | 650 |
| Fold 2 | 4178 | 286 | 4714 | 822 |
| Fold 3 | 3840 | 242 | 4758 | 1160 |
| Fold 4 | 4603 | 714 | 4286 | 397 |
| Fold 5 | 4493 | 452 | 4548 | 507 |

Tabel 4.4 Hasil confusion matrix model_nhpt3

| ***Fold*** | ***True Positive*** | ***False Positive*** | ***True Negative*** | ***False Negative*** |
|---|---|---|---|---|
| Fold 1 | 4536 | 574 | 4426 | 464 |
| Fold 2 | 4644 | 811 | 4189 | 356 |
| Fold 3 | 4644 | 671 | 4329 | 356 |
| Fold 4 | 4497 | 497 | 4503 | 503 |
| Fold 5 | 4380 | 553 | 4447 | 620 |

Tabel 4.5 Hasil confusion matrix model_nhpt4

| ***Fold*** | ***True Positive*** | ***False Positive*** | ***True Negative*** | ***False Negative*** |
|---|---|---|---|---|
| Fold 1 | 4503 | 500 | 4500 | 497 |
| Fold 2 | 4587 | 676 | 4324 | 413 |
| Fold 3 | 4683 | 725 | 4275 | 317 |
| Fold 4 | 4460 | 592 | 4408 | 540 |
| Fold 5 | 4441 | 488 | 4512 | 559 |

Selanjutnya berdasarkan confusion matrix tersebut maka dilakukan pengukuran kinerja model secara lebih detail, dengan melakukan perhitungan metrik-metrik penting seperti

accuracy, *precision*, *recall* (sensitivitas), dan F1-score yang perhitungannya sudah dibahas dalam sub bab 2.2.7. Hasil dari perhitungan setiap fold dapat dilihat pada Tabel 4.6, Tabel 4.7, Tabel 4.8, dan Tabel 4.9.

Tabel 4.6 Evaluasi tiap fold model_nhpt1

| ***Fold*** | ***Accuracy*** | ***Precision*** | ***Recall*** | ***F1-score*** |
|---|---|---|---|---|
| Fold 1 | 0,9013 | 0,9270 | 0,8712 | 0,8982 |
| Fold 2 | 0,8984 | 0,8686 | 0,9388 | 0,9023 |
| Fold 3 | 0,8867 | 0,9466 | 0,8196 | 0,8786 |
| Fold 4 | 0,8937 | 0,9092 | 0,8748 | 0,8917 |
| Fold 5 | 0,8954 | 0,8962 | 0,8944 | 0,8953 |

Tabel 4.7 Evaluasi tiap fold model_nhpt2

| ***Fold*** | ***Accuracy*** | ***Precision*** | ***Recall*** | ***F1-score*** |
|---|---|---|---|---|
| Fold 1 | 0.8909 | 0.9079 | 0.8700 | 0.8886 |
| Fold 2 | 0.8892 | 0.9359 | 0.8356 | 0.8829 |
| Fold 3 | 0.8598 | 0.9407 | 0.7680 | 0.8456 |
| Fold 4 | 0.8889 | 0.8657 | 0.9206 | 0.8923 |
| Fold 5 | 0.9041 | 0.9086 | 0.8986 | 0.9036 |

Tabel 4.8 Evaluasi tiap fold model_nhpt3

| ***Fold*** | ***Accuracy*** | ***Precision*** | ***Recall*** | ***F1-score*** |
|---|---|---|---|---|
| Fold 1 | 0.8962 | 0.8877 | 0.9072 | 0.8973 |
| Fold 2 | 0.8833 | 0.8513 | 0.9288 | 0.8884 |
| Fold 3 | 0.8973 | 0.8738 | 0.9288 | 0.9004 |
| Fold 4 | 0.9000 | 0.9005 | 0.8994 | 0.8999 |
| Fold 5 | 0.8827 | 0.8879 | 0.8760 | 0.8819 |

Tabel 4.9 Evaluasi tiap fold model_nhpt4

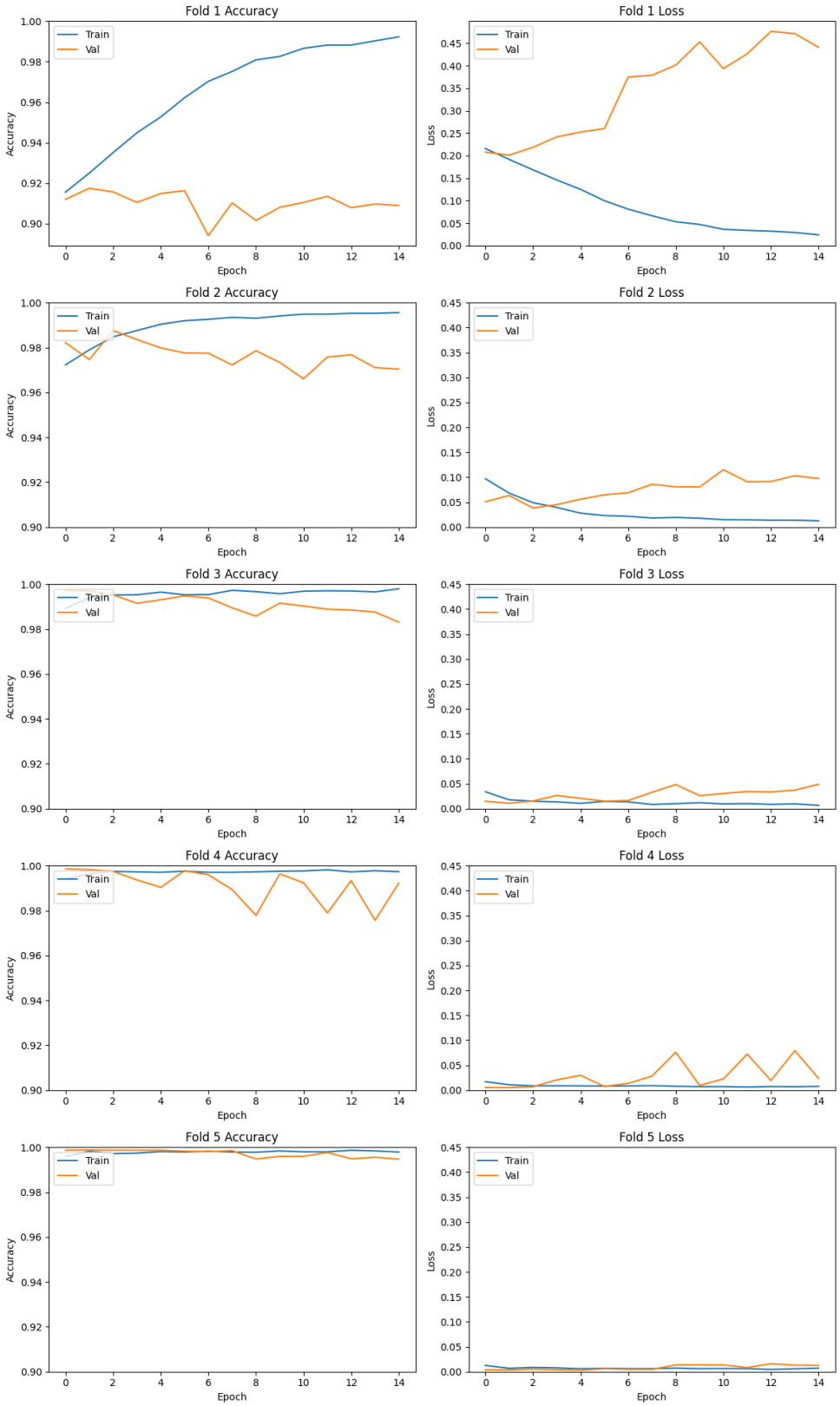
| ***Fold*** | ***Accruacy*** | ***Precision*** | ***Recall*** | ***F1-score*** |
|---|---|---|---|---|
| Fold 1 | 0.9003 | 0.9001 | 0.9006 | 0.9003 |
| Fold 2 | 0.8911 | 0.8716 | 0.9174 | 0.8939 |
| Fold 3 | 0.8958 | 0.8659 | 0.9366 | 0.8999 |
| Fold 4 | 0.8868 | 0.8828 | 0.8920 | 0.8874 |
| Fold 5 | 0.8953 | 0.9010 | 0.8882 | 0.8946 |

Berdasarkan tabel evaluasi diatas, secara keseluruhan tiap model berbeda (model_nhpt1, model_nhpt2, model_nhpt3, dan model_nhpt4) pada lima fold berbeda menunjukkan konsistensi dalam kinerja dan memiliki kemampuan yang baik dalam mengklasifikasikan data uji, dengan nilai akurasi berada diantara 86% hingga 90%. Hal ini mengindikasikan bahwa model tanpa *hyperparameter tuning* memiliki kemampuan yang baik untuk memprediksi dengan benar dalam berbagai data yang telah diujikan. Model dengan

akurasi tertinggi didapatkan model_nhpt2 pada fold 5 dengan nilai akurasi sebesar 90,41% yang memiliki kombinasi hyperparameter 32 LSTM unit, 0.4 Dropout_1, 128 Dense unit, dan 0.2 Droput_2. Rata-rata akurasi model_nhpt1, model_nhpt2, model_nhpt3, dan model_nhpt4 secara berurutan yaitu 89,51%, 88,66%, 89,19%, dan 89,39%.

### 4.4.2 Hasil SKCV untuk model dengan menggunakan *hyperparameter tuning*

Pada bagian ini, model yang dilakukan SKCV merupakan model terbaik yang dihasilkan dari proses *hyperparameter tuning* menggunakan Random Search yaitu model trial_02, dengan detail ada pada sub bab 4.3.2 yang setelah ini akan disebut sebagai model_hpt. Yang mana model tersebut dilakukan SKCV dengan hasil learning curve proses tersebut dapat dilihat pada Gambar 4.19.




Gambar 4.19 Learning curve tiap fold untuk model_hpt

Secara keseluruhan, model_hpt yang menjadi model terbaik hasil dari *hyperparameter tuning* random search menunjukkan tingkat akurasi yang sangat tinggi dan cenderung stabil mendekati di persentase 100%. Ini menunjukkan bahwa model sangat efektif mempelajari pola dari data pelatihan. Tetapi disisi lain, model mengindikasikan terjadi osilasi atau terdapat fluktuasi yang signifikan pada akurasi dan loss untuk data validasi, yang mungkin saja menunjukkan bahwa model tidak sepenuhnya stabil saat diterapkan dalam data baru yang tidak terlihat selama pelatihan.

Kemudian pada model ini juga dilakukan evaluasi terhadap kinerja model dengan menggunakan confusion matrix untuk setiap foldnya. Untuk hasil dari confusion matrix tiap foldnya dapat dilihat pada Tabel 4.10.

Tabel 4.10 Hasil confusion matrix model_hpt

| ***Fold*** | ***True Positive*** | ***False Negative*** | ***True Negative*** | ***False Positive*** |
|---|---|---|---|---|
| Fold 1 | 4440 | 350 | 4650 | 560 |
| Fold 2 | 4829 | 125 | 4875 | 171 |
| Fold 3 | 4867 | 36 | 4964 | 133 |
| Fold 4 | 4943 | 20 | 4980 | 57 |
| Fold 5 | 4972 | 25 | 4975 | 28 |

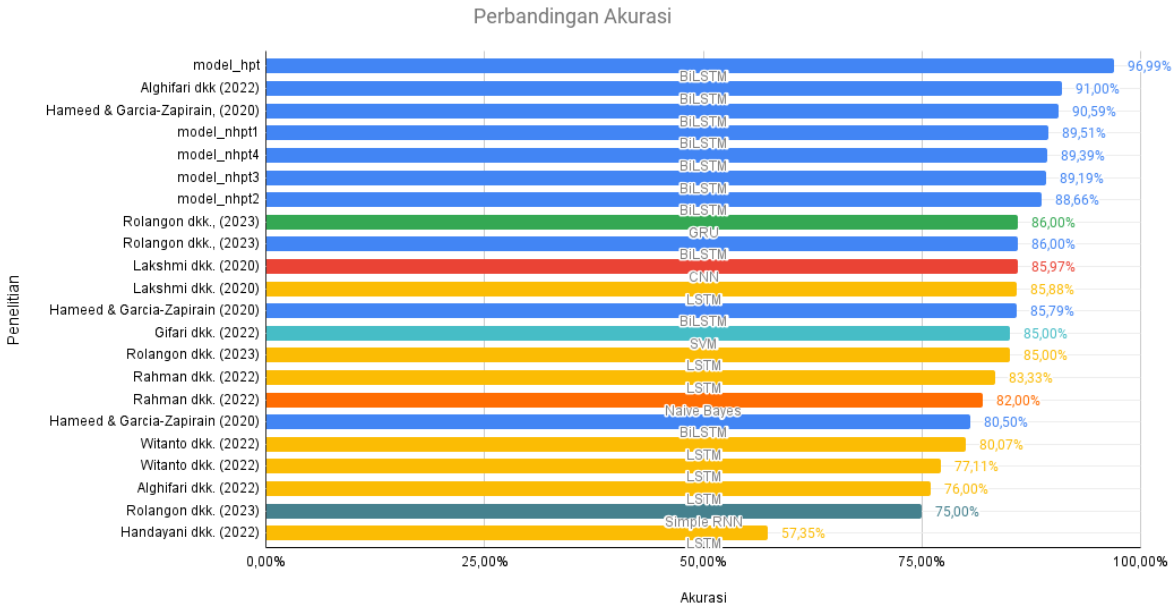
Berdasarkan confusion matrix diatas, dapat dilakukan analisis mengenai kinerja model dengan beberapa metrik seperti accuracy, *precision*, *recall*, dan f1-score. Hasil tersebut dapat dilihat pada Tabel 4.11.

Tabel 4.11 Evaluasi tiap fold model_hpt

| Fold | Accuracy | *Precision* | *Recall* | *F1-score* |
|---|---|---|---|---|
| Fold 1 | 0,9090 | 0,9269 | 0,8880 | 0,9070 |
| Fold 2 | 0,9704 | 0,9748 | 0,9658 | 0,9703 |
| Fold 3 | 0,9831 | 0,9927 | 0,9734 | 0,9829 |
| Fold 4 | 0,9923 | 0,9960 | 0,9886 | 0,9923 |
| Fold 5 | 0,9947 | 0,9950 | 0,9944 | 0,9947 |

Secara keseluruhan, model terbaik yang dihasilkan oleh random search memiliki kinerja yang konsisten dan sangat baik pada semua fold, dengan akurasi yang terdapat peningkatan pada tiap foldnya. Akurasi terbaik didapatkan pada fold 5 dengan akurasi yang dihasilkan yaitu sebesar 99,47%, menandakan bahwa model ini memprediksi dengan hampir sempurna. Akurasi rata-rata keseluruhan fold didapatkan pada angka 96,99%, yang menunjukkan bahwa

secara keseluruhan, hasil dari *hyperparameter tuning* yang dihasilkan oleh Random Search memiliki performa yang sangat baik dalam mengklasifikasikan sentimen.




Gambar 4.20 Perbandingan akurasi model yang diujikan dengan penelitian sebelumnya

Berdasarkan Gambar 4.20, terlihat bahwa model_hpt yang merupakan hasil dari hyperparameter tuning memiliki akurasi lebih baik jika dibandingkan dengan model_nhpt1, model_nhpt2, model_nhpt3, dan model_nhpt4 yang tidak menggunakan hyperparameter tuning serta model lain pada penelitian sebelumnya. Oleh karena itu, model_hpt digunakan untuk memprediksi kata baru selama proses pengujian.

## 4.5 Pengujian

Tahap pengujian dilakukan untuk memverifikasi dan mengukur kinerja model dalam kondisi yang mirip dengan penggunaan nyata. Model yang digunakan untuk pengujian merupakan model terbaik yang didapat dari proses *hyperparameter tuning* menggunakan random search yang pada tahap evaluasi menghasilkan kinerja paling optimal. Pada tahapan ini model akan diuji menggunakan beberapa sampel ulasan film yang diambil dari situs IMDb dengan model akan diuji dengan data pengujian yang belum pernah digunakan selama proses pelatihan atau validasi. Hasil pengujian dapat dilihat pada Tabel 4.12.

Tabel 4.12 Pengujian model dengan sampel ulasan

| Ulasan film | Hasil Prediksi |
|---|---|
| Rating: 8/10<br>Link ulasan:<br>https://www.imdb.com/review/rw7736031/?ref_=tt_urv<br><br>**Absolutely hilarious but not for everyone**<br>"Deadpool" is absolutely hilarious. The story isn't groundbreaking, but it does the trick and the same goes for the action which isn't bad either. The greatness of this film lies in its humor and its uniqueness. It isn't like every other superhero film. Whether you talk about the constant breaking of the fourth wall, the R-rating, or any of the very different characteristics of this film that differentiate it from other superhero movies. To be frank, it's arguable doesn't even justify the "superhero" stamp. It's something different.<br><br>It needs to be said, the film might not be for everyone. It's very crude and it's filled with nudity and violence. The humor won't appeal to everyone either. It's quite dark sometimes and very satirical. Sure, "Deadpool" isn't for everyone, but to the right audience, it's a masterpiece. The humor is great and there are countless hilarious jokes throughout that will make you laugh. Ryan Reynolds is perfect for Deadpool and he does so well in the role. It is a truly great performance by him.<br><br>To sum up, "Deadpool" is a great film that might not be for everyone. I recommend it. | 1/1 [==============================] - 0s 84ms/step<br>[0.9999999]<br>"Deadpool" is absolutely hilarious. The story isn't groundbreaking, but it does the trick and the<br>same goes for the action which isn't bad either. The greatness of this film lies in its humor and<br>its uniqueness. It isn't like every other superhero film. Whether you talk about the constant<br>breaking of the fourth wall, the R-rating, or any of the very different characteristics of this<br>film that differentiate it from other superhero movies. To be frank, it's arguable doesn't even<br>justify the "superhero" stamp. It's something different. It needs to be said, the film might not be<br>for everyone. It's very crude and it's filled with nudity and violence. The humor won't appeal to<br>everyone either. It's quite dark sometimes and very satirical. Sure, "Deadpool" isn't for everyone,<br>but to the right audience, it's a masterpiece. The humor is great and there are countless hilarious<br>jokes throughout that will make you laugh. Ryan Reynolds is perfect for Deadpool and he does so<br>well in the role. It is a truly great performance by him. To sum up, "Deadpool" is a great film<br>that might not be for everyone. I recommend it.<br><br>Prediksi: positif |
| Rating: 1/10<br>Link ulasan:<br>https://www.imdb.com/review/rw6541962/?ref_=tt_urv<br><br>**Disappointing direction!**<br>I didn't like the story, the soundtrack have no sense with the visual narrative, there aren't other interesting characters on the picture.<br>Is a poor copy from the wonder woman animated movie. Less interesting, less action, less everything. | 1/1 [==============================] - 1s 627ms/step<br>[1.4233201e-09]<br>I didn't like the story, the soundtrack have no sense with the visual<br>narrative, there aren't other interesting characters on the picture.<br>Is a poor copy from the wonder woman animated movie. Less<br>interesting, less action, less everything.<br><br>Prediksi: negatif |
| Rating: 9/10<br>Link ulasan:<br>https://www.imdb.com/review/rw6083663/?ref_=tt_urv<br><br>**What's with all the hate?! I LOVE MAN OF** | [0.99917006]<br>Am I one of the few people who actually love this movie? Man of Steel has beautiful visuals, an<br>amazing cast ensemble, and action scenes to blow you away. The story is better than that awful |

| Ulasan film | Hasil Prediksi |
|---|---|
| **STEEL!!!**<br>Am I one of the few people who actually love this movie? Man of Steel has beautiful visuals, an amazing cast ensemble, and action scenes to blow you away. The story is better than that awful Superman Returns movie and it's like the actors are doing their best. It's no Wonder Woman or Aquaman, but its a very fun watch and completely unforgettable!! | Superman Returns movie and it's like the actors are doing their best. It's no Wonder Woman or<br>Aquaman, but its a very fun watch and completely unforgettable!!<br><br>Prediksi: positif |
| Rating: 5/10<br>Link ulasan: https://www.imdb.com/review/rw9452688/?ref_=tt_urv<br><br>**Long, Boring and not needed.**<br>No matter how faithful a movie is to the book, it can't save itself from the fact that it's unneeded. Backstory to snow is fine if it adds something new to the universe, however nothing new is added here, we get taught everything we knew about the world and get tiny little snips of backstory of the origins of the Hunger Games. For the movies insane long run time, the final act of this film seems like a race to get to the end of the story. The slow paced world building it creates in the first two acts is immediately thrown out for the purpose of reaching that finish line. Unlike the last Hunger Games movies being split into two, this is the Hunger Games movie that should have been split into two. We don't get to the motivations and reasonings behind characters other than a line or two then we arrive at them. Another hour and a half of story telling could have fleshed these characters out and made the ending more meaningful. However we're left with a film that doesn't seem to ever end, masked with good acting, set design and great cinematography. But a film that is messy, long and drawn out only to be shot into a moc 10 speed at the end. Leaving this movie feel completely unnecessary. | [0.04960833]<br>No matter how faithful a movie is to the book, it can't save itself from the fact that it's<br>unneeded. Backstory to snow is fine if it adds something new to the universe, however nothing new<br>is added here, we get taught everything we knew about the world and get tiny little snips of<br>backstory of the origins of the Hunger Games. For the movies insane long run time, the final act of<br>this film seems like a race to get to the end of the story. The slow paced world building it<br>creates in the first two acts is immediately thrown out for the purpose of reaching that finish<br>line. Unlike the last Hunger Games movies being split into two, this is the Hunger Games movie that<br>should have been split into two. We don't get to the motivations and reasonings behind characters<br>other than a line or two then we arrive at them. Another hour and a half of story telling could<br>have fleshed these characters out and made the ending more meaningful. However we're left with a<br>film that doesn't seem to ever end, masked with good acting, set design and great cinematography.<br>But a film that is messy, long and drawn out only to be shot into a moc 10 speed at the end.<br>Leaving this movie feel completely unnecessary.<br><br>Prediksi: negatif |
| Rating: 10/10<br>Link ulasan: https://www.imdb.com/review/rw9189617/?ref_=tt_urv<br><br>**Incredibly Magical Last Hayao Movie**<br>Absolute magical, most likely last movie, made by Miyazaki Hayao. To give you an idea as to what you can expect from this movie, it has a | 1/1 [==============================] - 1s 502ms/step<br>[0.99997854]<br>Absolute magical, most likely last movie, made by Miyazaki Hayao. To give you an idea as to what<br>you can expect from this movie, it has a similar feel of "Alice in Wonderland" but imbuded with the<br>Ghibli magic you're familiar with. It includes elements of many of the other Ghibli movies |

| Ulasan film | Hasil Prediksi |
|---|---|
| similar feel of "Alice in Wonderland" but imbuded with the Ghibli magic you're familiar with. It includes elements of many of the other Ghibli movies "Princess Mononoke", "Pompoko", and "My neighbor Totoro" but adds something new that's never been made before by Hayao. To understand the full scope of it, you'd have to watch it a second time; to fully appreciate why the world is set up as it is and why the characters are the way they are.<br><br>I can't wait for the English release and the internationalization. Even though no promotional footage or trailers have been shown, I recommend this movie to anybody looking for something magical and mystical. | "Princess Mononoke", "Pompoko", and "My neighbor Totoro" but adds something new that's never been<br>made before by Hayao. To understand the full scope of it, you'd have to watch it a second time; to<br>fully appreciate why the world is set up as it is and why the characters are the way they are. I<br>can't wait for the English release and the internationalization. Even though no promotional footage<br>or trailers have been shown, I recommend this movie to anybody looking for something magical and<br>mystical.<br><br>Prediksi: positif |
| Rating: 4/10<br>Link ulasan: https://www.imdb.com/review/rw9482240/?ref_=tt_urv<br><br>**Another Disney flop.**<br>As a 100th anniversary celebration, this was seriously below par, given Disney's recent, dreary *output*, I'm not sure what I was expecting, but this was woeful.<br><br>It felt as though they'd borrowed bits of Disney classics, and bolted them together as a kind of eco friendly Frankenstein. It doesn't manage to spark into life, arguably the most forgettable Disney characters to date.<br><br>I've read several comments about the animation, and plenty seem to applaud the style, quality and colours, let me be honest, this sucks, the most basic and unimaginative animation I think I've seen, to me it looks unfinished, did they get it out in a rush, or were there budget cuts, this isn't Disney standard, let alone a Centenary special.<br><br>I genuinely fear for Disney, everything they're touching seems to be going to pot, they need a major shake up, Snow White doesn't look like a classic, where do Disney go from this?<br><br>Woeful. | 1/1 [==============================] - 0s 34ms/step<br>[3.1229952e-06]<br>As a 100th anniversary celebration, this was seriously below par, given Disney's recent, dreary<br>*output*, I'm not sure what I was expecting, but this was woeful. It felt as though they'd borrowed<br>bits of Disney classics, and bolted them together as a kind of eco friendly Frankenstein. It<br>doesn't manage to spark into life, arguably the most forgettable Disney characters to date. I've<br>read several comments about the animation, and plenty seem to applaud the style, quality and<br>colours, let me be honest, this sucks, the most basic and unimaginative animation I think I've<br>seen, to me it looks unfinished, did they get it out in a rush, or were there budget cuts, this<br>isn't Disney standard, let alone a Centenary special. I genuinely fear for Disney, everything<br>they're touching seems to be going to pot, they need a major shake up, Snow White doesn't look like<br>a classic, where do Disney go from this? Woeful.<br><br>Prediksi: negatif |

Berdasarkan hasil pengujian menggunakan beberapa sampel ulasan film yang berasal dari berbagai film dengan beragam macam rating, model dapat memprediksi dengan baik sentimen yang terkandung pada ulasan film tersebut. Ini artinya model bukan hanya memiliki kinerja terhadap data yang telah digunakan selama proses pelatihan dan validasi, tetapi juga menunjukkan kinerja yang baik dalam menggeneralisasi dan menerapkan pembelajaran yang diperoleh pada data-data yang benar baru dan berbeda. Hal ini juga membuktikan bahwa model dapat diaplikasikan dalam skenario dunia nyata, dimana model dapat menangani variasi dalam bahasa dan konteks yang ditemukan dalam ulasan film.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan hasil analisis dan pembahasan data dalam analisis sentimen ulasan film menggunakan model BiLSTM dengan parameter yang ditentukan dengan dan model BiLSTM dengan *hyperparameter tuning* menggunakan *Random search*, kesimpulan yang dapat diambil yaitu:

1. Penerapan *hyperparameter tuning* menggunakan Random Search pada model BiLSTM (model_hpt) berhasil memperoleh akurasi lebih baik jika dibandingkan dengan model pada penelitian sebelumnya dan model dengan parameter yang ditentukan sendiri (model_nhpt1, model_nhpt2, model_nhpt3, dan model_nhpt4).
2. Model_hpt dengan parameter hasil dari Random search memperoleh akurasi rata-rata keseluruhan yaitu 96,99%. Sedangkan model_nhpt1, model_nhpt2, model_nhpt3, dan model_nhpt4 dengan parameter yang ditentukan sendiri memperoleh rata-rata akurasi keseluruhan fold sebesar 89,51%, 88,66%, 89,19%, dan 89,39% dalam Stratified k-fold cross validation.
3. *Hyperparameter tuning* menggunakan Random search meskipun membutuhkan waktu serta komputasi yang banyak untuk mengeksplorasi berbagai kombinasi, *hyperparameter tuning* terbukti efektif dalam menemukan konfigurasi optimal model dalam meningkatkan akurasi.
4. Dengan menggunakan Random Search, model dapat mengeksplorasi ruang parameter lebih luas dan lebih variatif, yang mungkin tidak terjangkau ketika melakukan pencarian parameter optimal secara manual.
5. Diperlukan keahlian dan pengalaman lebih, dalam menentukan parameter optimal secara manual. Jika dibandingkan dengan menentukan parameter optimal menggunakan Random search, yang tidak diperlukan keahlian dan pengalaman lebih karena Random search secara otomatis melakukan pencarian terhadap kombinasi optimal.
6. Walaupun model yang dikonfigurasi secara manual dan model yang parameter-nya ditentukan melalui Random Search menunjukkan performa yang memuaskan, analisis kurva pembelajaran dari kedua model tersebut mengungkapkan terjadinya *osilasi* pada model tersebut.

## 5.2 Saran

Terdapat beberapa saran yang dapat dilakukan pada penelitian selanjutnya, yaitu:

1. Menggunakan data ulasan film terbaru untuk variasi kata yang lebih beragam dan memperoleh semantik kata yang lebih luas.
2. Melakukan penambahan beberapa penyesuaian seperti ukuran learning rate, batch size maupun kompleksitas model untuk menghindari terjadinya osilasi.
3. Dapat dilakukan *hyperparameter tuning* menggunakan model lain seperti *Grid Search* maupun *Bayesian Optimization*.
4. Menambahkan prediksi terhadap aspek yang dibahas dalam ulasan film.

# DAFTAR PUSTAKA


Alghifari, D. R., Edi, M., & Firmansyah, L. (2022). Implementasi Bidirectional LSTM untuk Analisis Sentimen Terhadap Layanan Grab Indonesia. *Jurnal Manajemen Informatika (JAMIKA)*, *12*(2), 89–99. https://doi.org/10.34010/jamika.v12i2.7764

Bergstra, J., Ca, J. B., & Ca, Y. B. (2012). Random Search for Hyper-Parameter Optimization Yoshua Bengio. In *Journal of Machine Learning Research* (Vol. 13). http://scikit-learn.sourceforge.net.

Cui, Z., Ke, R., & Wang, Y. (2018). *Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction*. https://api.semanticscholar.org/CorpusID:24760423

Eliashberg, J., & Shugan, S. (1997). Film Critics: Influencers or Predictors? *Journal of Marketing*, *61*. https://doi.org/10.2307/1251831

Fawcett, T. (2006). Introduction to ROC analysis. *Pattern Recognition Letters*, *27*, 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Gifari, O. I., Adha, M., Rifky Hendrawan, I., Freddy, F., & Durrand, S. (2022). Analisis Sentimen Review Film Menggunakan TF-IDF dan Support Vector Machine. *JIFOTECH (JOURNAL OF INFORMATION TECHNOLOGY*, *2*(1).

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5–6), 602–610. https://doi.org/10.1016/j.neunet.2005.06.042

Hameed, Z., & Garcia-Zapirain, B. (2020). Sentiment Classification Using a Single-Layered BiLSTM Model. *IEEE Access*, *8*, 73992–74001. https://doi.org/10.1109/ACCESS.2020.2988550

Handayani, S. F., Pratiwi, R. W., Dairoh, D., & Af'idah, D. I. (2022). Analisis Sentimen pada Data Ulasan Twitter dengan Long-Short Term Memory. *JTERA (Jurnal Teknologi Rekayasa)*, *7*(1), 39. https://doi.org/10.31544/jtera.v7.i1.2022.39-46

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

*Influence of movie critics reviews in the U.S. 2017*. (2017, February 16). Https://Www.Statista.Com/Statistics/682930/Movie-Critic-Reviews-Influence/.

Iswara, A. J. (2020, January 12). *Dalam 1 Dekade, Penonton Film Indonesia Meningkat 5 Kali Lipat*. Https://Www.Goodnewsfromindonesia.Id/2020/01/12/Penonton-Film-Indonesia.

Lakshmi, P. V, Kumar, N. S., Rao, G., & Medaka, S. (2020). A Deep Learning Approach for Sentiment Analysis of Movie Reviews. *Test Engineering and Management*, *83*.

Larochelle, H., Erhan, D., Courville, A. C., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:14805281

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, 142–150.

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/https://doi.org/10.1016/j.asej.2014.04.011

Pang, B., & Lee, L. (2005). *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*.

Pasaribu, D., Kusrini, K., & Sudarmawan, S. (2020). Peningkatan Akurasi Klasifikasi Sentimen Ulasan Makanan Amazon dengan Bidirectional LSTM dan Bert Embedding. *Inspiration: Jurnal Teknologi Informasi Dan Komunikasi*, *10*. https://doi.org/10.35585/inspir.v10i1.2568

Prusty, S., Patnaik, S., & Dash, S. K. (2022). SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer. *Frontiers in Nanotechnology*, *4*. https://doi.org/10.3389/fnano.2022.972421

Rahman, A., #1, I., Sulistiani, H., Miftaq, B., #3, H., Nurkholis, A., & #5, S. (2022). Analisis Perbandingan Algoritma LSTM dan Naive Bayes untuk Analisis Sentimen. *JEPIN (Jurnal Edukasi Dan Penelitian Informatika) , 8*.

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

Rolangon, A., Weku, A., & Sandag, G. (2023). Perbandingan Algoritma LSTM Untuk Analisis Sentimen Pengguna Twitter Terhadap Layanan Rumah Sakit Saat Pandemi Covid-19. *TeIKa*, *null*, null. https://doi.org/10.36342/teika.v13i01.3063

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*, *abs/1910.01108*. https://api.semanticscholar.org/CorpusID:203626972

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, *45*, 2673–2681. https://api.semanticscholar.org/CorpusID:18375389

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*. http://nlp.stanford.edu/

Tripathy, A., Agrawal, A., & Rath, S. K. (2015). Classification of Sentimental Reviews Using Machine Learning Techniques. *Procedia Computer Science*, *57*, 821–829. https://doi.org/https://doi.org/10.1016/j.procs.2015.07.523

Wang, S., & Manning, C. (2012). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In H. Li, C.-Y. Lin, M. Osborne, G. G. Lee, & J. C. Park (Eds.), *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 90–94). Association for Computational Linguistics. https://aclanthology.org/P12-2018

Widodo, S., Brawijaya, H., & Samudi, S. (2022). Stratified K-fold cross validation optimization on machine learning for prediction. *Sinkron*, *7*(4), 2407–2414. https://doi.org/10.33395/sinkron.v7i4.11792

Witanto, S., NgurahAgusSanjaya, E. R., Karyawati, A. E., Arya, Kadyanan, Suhartana, K. G., & Astuti, L. G. (2022). Implementasi LSTM Pada Analisis Sentimen Review Film Menggunakan Adam Dan RMSprop Optimizer. *JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)*, *null*, null. https://doi.org/10.24843/jlk.2022.v10.i04.p05

Zuhdi, A. M., Utami, E., & Raharjo, S. (2019). *ANALISIS SENTIMENT TWITTER TERHADAP CAPRES INDONESIA 2019 DENGAN METODE K-NN*. https://api.semanticscholar.org/CorpusID:213809098

**LAMPIRAN**

```
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import tensorflow as tf
import keras_tuner as kt

from datasets import load_dataset

from transformers import AutoTokenizer
from transformers import TFAutoModel

from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix, classification_report

from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, Bidirectional, LSTM, Dropout,
Dense
from keras.optimizers import Adam

from tensorflow import keras
from tensorflow.keras.layers import Input
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import plot_model

dataset = load_dataset("imdb")
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")

path = "/content/drive/MyDrive/"

train_df = pd.DataFrame(dataset['train'])
test_df = pd.DataFrame(dataset['test'])
```

```
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")

def preprocess(data):
    return tokenizer(data['text'], padding="max_length",
truncation=True, max_length=512)

train_df['processed'] = train_df.apply(lambda x: preprocess(x),
axis=1)
train_input_ids = np.array([x['input_ids'] for x in
train_df['processed']])
train_labels = train_df['label'].values

test_df['processed'] = test_df.apply(lambda x: preprocess(x),
axis=1)
test_input_ids = np.array([x['input_ids'] for x in
test_df['processed']])
test_labels = test_df['label'].values

combined_input_ids = np.concatenate([test_input_ids,
train_input_ids])
combined_labels = np.concatenate([test_labels, train_labels])

distilbert_model = TFAutoModel.from_pretrained("distilbert-base-
uncased")
embedding_layer =
distilbert_model.get_layer("distilbert").embeddings

def BiLSTM_non_hpt(tokenizer, embedding_layer):
    model = Sequential()
    model.add(Embedding(input_dim=tokenizer.vocab_size,
                        output_dim=embedding_layer.dim,
                        weights=[embedding_layer.weights[0]],
                        trainable=False))
    model.add(Bidirectional(LSTM(64)))
    model.add(Dropout(0.5))
    model.add(Dense(64))
```

```
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
    model.summary()
    plot_model(model, show_shapes=True)
    return model

model_instance = BiLSTM_non_hpt(tokenizer, embedding_layer)
plot_model(model_instance, show_shapes=True)

def hp_tuning(hp):
    model = Sequential()
    model.add(Embedding(input_dim=tokenizer.vocab_size,
                        output_dim=embedding_layer.dim,
                        weights=[embedding_layer.weights[0]],
                        trainable=False))
    model.add(Bidirectional(LSTM(hp.Int('lstm_units', min_value=32,
max_value=128, step=32))))
    model.add(Dropout(hp.Float('dropout_1', min_value=0.2,
max_value=0.5, step=0.1)))
    model.add(Dense(hp.Int('dense_units', min_value=32,
max_value=128, step=32)))
    model.add(Dropout(hp.Float('dropout_2', min_value=0.2,
max_value=0.5, step=0.1)))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
    return model

tuner_directory = os.path.join(path, 'hpt_random_search')

tuner = kt.RandomSearch(hp_tuning,
                        objective='val_accuracy',
                        max_trials=10,
```

```
                        executions_per_trial=1,
                        directory= tuner_directory,
                        project_name='bilstm')

tuner.search(train_input_ids, train_labels, epochs=15,
validation_data=(test_input_ids, test_labels))

best_model = tuner.get_best_models(num_models=1)[0]
best_model.summary()
plot_model(best_model, show_shapes=True)

def BiLSTM_hpt():
  return best_model

def visualize_results(histories, conf_matrices):
    plt.figure(figsize=(20, 6))
    for i, history in enumerate(histories):
        plt.subplot(2, len(histories), i+1)
        plt.plot(history['accuracy'])
        plt.plot(history['val_accuracy'])
        plt.title(f'Fold {i+1} Accuracy')
        plt.ylabel('Accuracy')
        plt.xlabel('Epoch')
        plt.legend(['Train', 'Val'], loc='upper left')

        plt.subplot(2, len(histories), i+1+len(histories))
        plt.plot(history['loss'])
        plt.plot(history['val_loss'])
        plt.title(f'Fold {i+1} Loss')
        plt.ylabel('Loss')
        plt.xlabel('Epoch')
        plt.legend(['Train', 'Val'], loc='upper left')

    plt.tight_layout()
    plt.show()

    for i, cm in enumerate(conf_matrices):
```

```
        plt.figure(figsize=(10, 7))
        sns.heatmap(cm, annot=True, fmt='g', cmap='Blues')
        plt.title(f'Confusion Matrix for Fold {i+1}')
        plt.ylabel('Actual')
        plt.xlabel('Predicted')
        plt.show()

def evaluate_model(create_model, input_ids, labels, n_splits=5,
epochs=15, save_dir='/content/drive/MyDrive/'):
    skf = StratifiedKFold(n_splits=n_splits, shuffle=True,
random_state=42)
    scores = []
    history_list = []
    conf_matrices = []
    saved_models = []

    if not os.path.exists(save_dir):
        os.makedirs(save_dir)

    best_score = 0
    best_model = None

    for fold, (train_index, test_index) in
enumerate(skf.split(input_ids, labels)):
        X_train_fold, X_test_fold = input_ids[train_index],
input_ids[test_index]
        y_train_fold, y_test_fold = labels[train_index],
labels[test_index]

        model = create_model()
        history = model.fit(X_train_fold, y_train_fold,
epochs=epochs, verbose=1, validation_data=(X_test_fold,
y_test_fold))
        history_list.append(history.history)

        score = model.evaluate(X_test_fold, y_test_fold,
verbose=0)[1]
```

```
        scores.append(score)

        model_path = os.path.join(save_dir, f'model_fold_{fold +
1}.h5')
        model.save(model_path)
        saved_models.append(model_path)

        if score > best_score:
            best_score = score
            best_model = model_path

        y_pred = model.predict(X_test_fold)
        y_pred = np.round(y_pred).astype(int).flatten()
        cm = confusion_matrix(y_test_fold, y_pred)
        conf_matrices.append(cm)

    visualize_results(history_list, conf_matrices)

    print(f"Best model saved at: {best_model}")
    return scores, history_list, conf_matrices, saved_models

best_model.save('/content/drive/MyDrive/bestmodel.keras')

tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")

model_predict = load_model('/content/drive/MyDrive/model_fold_1.h5')
def predict_sentiment(text):
    inputs = tokenizer.encode(text, return_tensors="tf",
truncation=True, padding='max_length', max_length=512)

    prediction = model_predict.predict(inputs)[0]
    print(prediction)

    sentiment = 'positif' if prediction > 0.5 else 'negatif'

    return sentiment
```

```
def print_pretty_text(text, column_width):
    words = text.split()
    current_line = ""
    for word in words:
        if len(current_line) + len(word) + 1 <= column_width:
            current_line += word + " "
        else:
            print(current_line.strip())
            current_line = word + " "
    if current_line:
        print(current_line.strip())

def predict_and_display():
    while True:
        input_text = input("\nMasukkan teks atau ketik 'exit' untuk
keluar: ")

        if input_text.lower() == 'n':
            print("Keluar dari program.")
            break

        prediction = predict_sentiment(input_text)

        print_pretty_text(input_text, 100)
        print("\nPrediksi:", prediction)

predict_and_display()
```