Implementasi Attention-Based BiLSTM dengan LoRA Parameter Tuning untuk Analisis Sentimen Ulasan Destinasi Wisata

Implementation of Attention-Based BiLSTM with LoRA Parameter Tuning for Tourist Destination Review Sentiment Analysis

Tristan Bey Kusuma[1], **I Made Widiartha**[2], **I Ketut Gede Suhartana**[3], **I Gusti Ngurah Anom Cahyadi Putra**[4]

[1,2,3,4]*Program Studi Informatika, Universitas Udayana*
Kuta Selatan, Badung, Bali, Indonesia
E-mail : tristanbeykusuma@gmail.com[1], madewidiartha@unud.ac.id[2], ikg.suhartana@unud.ac.id[3], anom.cp@unud.ac.id[4]*
**Corresponding author*

Received 15 November 2025; Revised 21 December 2025; Accepted 6 January 2026

Abstrak - Industri pariwisata merupakan salah satu sektor paling krusial dalam perekonomian Indonesia dan Bali, hal ini ditunjukkan oleh opini yang terdapat di media sosial dan situs pariwisata. Pada tahun 2024, situs informasi perjalanan TripAdvisor mengambil data sebanyak 31.1 juta ulasan objek wisata, yang mengungkap 2.7 juta ulasan palsu. Pengguna Indonesia menempati peringkat tertinggi dalam perihal jumlah ulasan berbayar, terutama untuk Bali, yang menempati posisi kedua di dunia dalam popularitas. Studi ini meneliti pengaruh metode *parameter efficient fine-tuning*, yaitu LoRA atau Low-Rank Adaptation, pada model *Attention-Based Bi-directional Long-Short Term Memory* (BiLSTM) dalam melakukan analisis sentimen ulasan objek wisata di Bali. Data yang dikumpulkan berasal dari ulasan wisata oleh pengguna di situs web TripAdvisor, yang terdiri dari 4966 sampel data. Berdasarkan evaluasi, model BiLSTM berbasis perhatian ini mencapai skor akurasi sebesar 0.78037 dibandingkan dengan model BiLSTM berbasis perhatian dengan LoRA, yang mencapai skor 0.79907. Waktu pelatihan untuk model *Attention-Based* BiLSTM dengan LoRA juga secara signifikan mengurangi waktu pelatihan dan penggunaan memori. Penelitian ini menunjukkan bahwa *Low-Rank Adaptation* efektif dalam meningkatkan kinerja dan efisiensi model ini.

Kata kunci - Pariwisata Bali, *Machine Learning*, Analisis Sentimen, BiLSTM, LoRA

Abstract – *The tourism industry is one of the most crucial sectors in the Indonesian and Balinese economy, as reflected in opinions shared on social media and tourism websites. In 2024, the travel information site TripAdvisor collected 31.1 million reviews of tourist attractions, revealing 2.7 million fake reviews. Indonesian users ranked highest in terms of paid reviews, particularly for Bali, which holds the second position worldwide in popularity. This study examines the impact of efficient fine-tuning parameter methods, namely LoRA or Low-Rank Adaptation, on the Attention-Based Bi-directional Long-Short Term Memory (BiLSTM) model in conducting sentiment analysis of tourist attraction reviews in Bali. The data were collected from user reviews on the TripAdvisor website, consisting of 4,966 samples. Based on the evaluation, the attention-based BiLSTM model achieved an accuracy score of 0.78037 compared to the attention-based BiLSTM model with LoRA, which reached a score of 0.79907. Training time for the Attention-Based BiLSTM with LoRA also significantly reduced training duration and memory usage. This research demonstrates that Low-Rank Adaptation is effective in improving the performance and efficiency of this model.*

Keywords – *Bali Tourism, Machine Learning, Sentiment Analysis, BiLSTM, LoRA*

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

1. PENDAHULUAN

Sekitar 4/5 aktivitas ekonomi di Bali pada tahun 2003 bergantung pada sektor pariwisata. Kondisi ini menunjukkan Bali sebagai daerah penyumbang devisa yang besar bagi perekonomian Indonesia. Di sini dapat dilihat pentingnya sektor pariwisata sebagai sektor utama yang menopang perekonomian Bali dan Indonesia secara keseluruhan [1]. Walau pandemi COVID-19 berpengaruh negatif pada sektor pariwisata di Bali, setelah pariwisata dimulai kembali jumlah wisatawan dari negara kembali meningkat. Ini menciptakan banyaknya opini-opini baru khususnya dalam media sosial mengenai lokasi pariwisata di Bali dari wisatawan-wisatawan yang baru berdatangan. Peningkatan aktivitas wisata ini dapat diamati pada berbagai ulasan dan opini yang diunggah pada situs wisata daring dan media sosial. Pada tahun 2024, ditemukan terdapat sebanyak 31.1 juta ulasan wisata online pada laman TripAdvisor di tahun 2024. Dari ulasan-ulasan tersebut ditemukan 2.7 juta ulasan palsu yang mencakup ulasan boosting atau promosi palsu, ulasan vandalisme atau review bombing, dan ulasan berbayar, dengan pengguna asal Indonesia menempati peringkat teratas terkait jumlah ulasan berbayar menurut negara (20%). Ulasan-ulasan palsu ini membutuhkan waktu untuk dideteksi oleh sistem, dimana sebagian membutuhkan waktu dari 6-12 jam hingga 13-24 jam [2].

Maka dari itu dibutuhkan sebuah analisis sentimen terhadap ulasan dan opini wisata tersebut. Sentimen ulasan wisata ini dapat dibagi menjadi tiga jenis, yaitu sentimen positif, netral, dan negatif. Maka dapat diimplementasikan suatu metode machine learning guna menganalisis beragam pendapat tersebut dan mengelompokkannya secara sistematis. Di antara berbagai metode machine learning, melalui pengelolaan aliran informasi melalui gerbang input, forget, dan output, LSTM (Long Short-Term Memory) memiliki kemampuan untuk mempelajari konteks sekuensial secara efektif dan menangani ketergantungan yang panjang. Kemudian, BiLSTM atau Bidirectional LSTM meningkatkan kemampuan LSTM standar dengan menangani urutan masukan dalam arah maju dan mundur secara bersamaan melalui dua lapisan. Pendekatan dua arah yang unik ini memungkinkan jaringan untuk memahami pola dan hubungan dari konteks masa lalu dan masa depan, sehingga sangat cocok untuk tugas yang melibatkan pemrosesan bahasa alami [3]. Pada implementasi model BiLSTM, ada yang disebut dengan mekanisme atensi. Hal ini memungkinkan model untuk memperhatikan semua token input pada setiap langkah prediksi. Sehingga model bisa fokus pada suatu kata yang dianggap penting terhadap konteks sentimen tersebut dibanding kata lainnya. Mekanisme atensi ini ditemukan dapat menghasilkan performa baik dengan memilih bagian sekuens yang dianggap berpengaruh [4]. Selain itu, ketika melakukan training, tentunya dibutuhkan tuning parameter. Hal ini biasanya memakan waktu dan memori yang cukup banyak, apalagi untuk model yang terdiri atas beberapa layer. Maka hadirlah Low-Rank Adaptation (LoRA), sebuah teknik yang memampukan adaptasi model yang efisien dengan parameter tambahan yang minimal. LoRA berfokus pada low-rank update pada bobot model, memungkinkan perolehan kinerja yang signifikan tanpa memerlukan daya komputasi yang besar [5]. Dalam penelitian ini, penulis meneliti pengaruh metode Low-Rank Adaptation (LoRA) dalam Attention-Based Bidirectional LSTM dalam *sentiment analysis* pada ulasan wisata di tempat wisata provinsi Bali yang diambil dari platform wisata online Tripadvisor. Penelitian akan membandingkan performa, efisiensi waktu pelatihan dan penggunaan memori dalam melakukan *fine-tuning* model tanpa LoRA dan model dengan LoRA tersebut untuk melakukan analisis sentimen teks ulasan wisata. Dari hasil perbandingan ini diharapkan dapat memilih model yang optimal tugas analisis sentimen tersebut. Selain itu juga diharapkan dapat menemukan *hyperparameter* yang sesuai agar menadapatkan hasil yang optimal untuk kedua model tersebut.

2. METODE PENELITIAN

Penelitian ini dilakukan dengan enam tahap, yaitu mulai dari pengumpulan data, pra-pemrosesan data (*preprocessing*), *data splitting*, pembangunan model, *hyperparameter tuning* dan evaluasi antar model.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

2.1 Pengumpulan Data

Data yang digunakan di penelitian ini diperoleh melalui proses *web crawling* pada laman dengan bantuan alat *scraping*. Sumber data berasal dari situs tripadvisor.com, yang menyediakan berbagai ulasan objek wisata Bali umum yang ditulis dengan bahasa Indonesia. Seluruh data yang dikumpulkan merupakan ulasan objek wisata yang diunggah user pada laman tersebut. Sampel ulasan user yang diambil adalah data teks ulasan dan rating yang diunggah ke laman ulasan tempat wisata. Kemudian data ini dilabeli menurut ratingnya ke sentimen negatif, netral, serta positif. Data tersebut dibagi menurut rating yang berkisar dari 1 sampai 5, dengan mapping label seperti berikut : 1-2 : label negatif; 3 : label netral; 4-5 : label positif. Sehingga, setiap data yang diambil akan diberi label negatif, netral, atau positif menurut rating pada situs tersebut.

2.2 Preprocessing

Teks *preprocessing* merupakan sebuah metode yang digunakan untuk mempermudah representasi teks dan membuat dokumen menjadi konsisten. Proses *text mining* sangat bergantung dari metode text preprocessing yang digunakan [6]. Berikut ini adalah text preprocessing yang digunakan :

1. Filtering

Pada tahap ini akan menghapus data yang tidak sesuai dengan labelnya. Misalnya data sentimen positif yang dilabel negatif dan sebaliknya. Dalam tahap ini akan dilakukan filtering terhadap data tersebut dengan menggunakan list stopword dimana kalimat bersentimen atau berlabel positif yang mengandung kata-kata yang jelas merupakan negatif akan dihapus dan kalimat berlabel negative yang mengandung kata-kata positif akan dihapus sehingga tidak ada data yang salah label. Hanya beberapa kata dengan bobot tinggi di leksikon ini yang digunakan.

2. Case-Folding dan Cleaning

Case-folding merupakan proses mengubah seluruh karakter dalam teks menjadi huruf kecil. Pada penelitian ini, tahapan tersebut diterapkan untuk menyamakan format teks dalam dataset ulasan agar tidak terjadi perbedaan makna. Selain itu, di sini juga dilakukan penghapusan punktuasi. Selain itu, pada tahapan ini juga dilakukan cleaning yaitu menghapus karakter yang tidak dibutuhkan misal URL, angka, dan punktuasi atau tanda baca.

3. Stopword Removal

Stopword removal adalah tahap penghapusan kata-kata yang tidak berpengaruh signifikan terhadap analisis. Langkah ini dilakukan untuk mengurangi kata-kata yang tidak membawa makna penting, sehingga hanya tersisa kata yang relevan untuk konteks analisis sentimen.

4. Tokenisasi

Tokenization merupakan proses memecah teks menjadi satuan-satuan terkecil, seperti kata atau frasa. Proses ini mengubah teks menjadi urutan token yang dapat diolah oleh model analisis sentimen.

5. Stemming

Stemming merubah struktur tata bahasa dan mengubah setiap kata menjadi bentuk akarnya. Stemming mengambil sebuah kata berimbuhan dan mengenali awalan, akhiran, atau sisipan pada kata tersebut. Kemudian imbuhan dihapus dan kata dinormalisasi menjadi bentuk dasarnya. Dengan ini tiap kata akan lebih mudah diproses saat melakukan proses pelatihan dan tidak ada bentuk berbeda-beda dari kata yang sama [6].

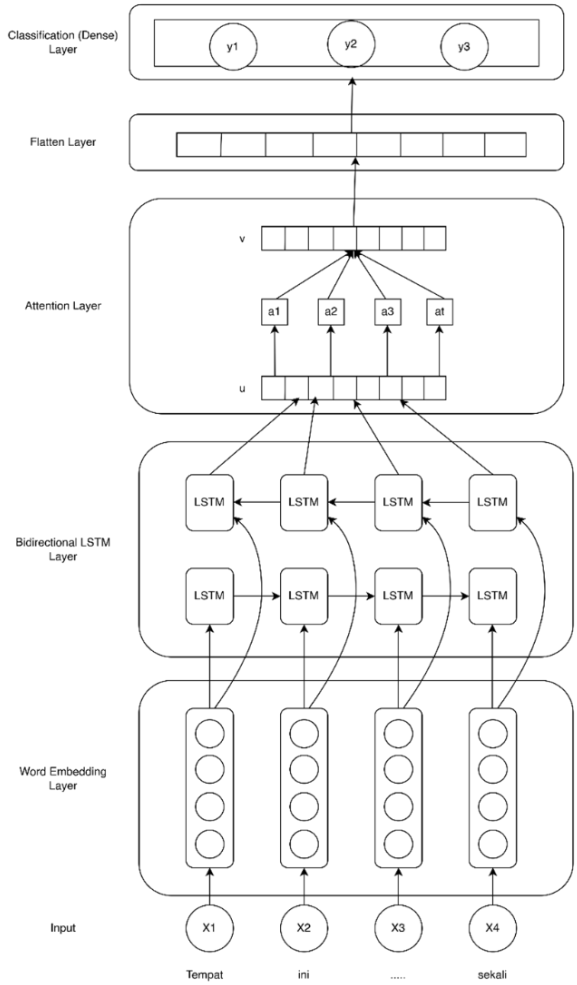
2.3 Data Splitting

Pembagian dilakukan pada dataset ini untuk bisa mengevaluasi kinerja model setelah pelatihan. Data yang telah melalui *preprocessing* ini dibagi beberapa, data training untuk melatih model, data validation digunakan untuk menilai performa selama pelatihan, dan data testing untuk evaluasi model dengan data yang belum dilihat. Namun, sebelum dibagi, pada tahap ini data diseimbangkan terlebih dahulu agar data tiap label berjumlah sama. Data sisa penyeimbangan ini akan digunakan sebagai data testing atau evaluasi model. Kemudian, data training dirubah

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

menjadi bentuk sekuens dengan melakukan padding agar panjang sekuens ini berukuran sama. Kemudian, label "*positive*", "*negative*", dan "*neutral*" dikonversi menjadi data numerik yaitu 0, 1 atau 2. Akhirnya, data hasil penyeimbangan tersebut memiliki proporsi 80:20 yaitu rasio data pelatihan dengan data validasi.

2.4 Pembangunan Attention-Based BiLSTM



Gambar 1. Arsitektur Model Attention-Based BiLSTM

Dalam membangun model tersebut pertama-tama diinisalisasikan layer model tersebut satu per satu dengan mendefinisikan input dan output setiap layer. Seperti terlihat pada Gambar 1, layer dimulai dari sebuah input layer kemudian layer embedding, layer BiLSTM, layer attention, flatten dan dense classification layer. Pelatihan model dimulai dari forward pass, kemudian akan dievaluasi. Jika memenuhi kondisi misalnya sudah mencapai akurasi yang diinginkan atau epoch maksimum maka pelatihan selesai. Jika belum, maka akan dilakukan backpropagation dimana loss function dihitung dan parameter di setiap layer di-update.

1. Layer Input

Di sinilah data mentah masuk ke dalam model. Dalam contoh ini, inputnya adalah serangkaian kata: "Tempat", "ini", "sekali". Setiap kata direpresentasikan oleh indeks numerik (X1, X2, X3, X4). Indeks ini sesuai dengan posisi kata dalam kosakata. Dalam kasus ini, kalimat "Tempat ini bagus" akan diubah menjadi seperti ini:
"Tempat" → indeks 10

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

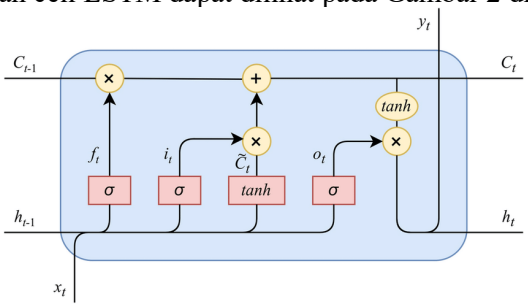
"ini" → indeks 50
"sekali" → indeks 120
2. Word Embedding

Layer ini mengubah indeks kata numerik menjadi representasi vektor padat (embedding). Pertama, layer ini membaca data sekuens tersebut per token. Kemudian akan dicari vektor embedding di lookup table embedding yang tersedia. Jika ditemukan maka akan dikembalikan vektor embedding tersebut sebagai representasi token tersebut, jika tidak maka akan dibuat sebuah *randomized vector* baru untuk token tersebut. Nilai-nilai vektor embedding ini akan diperbarui seiring pelatihan model. Saat memperbarui bobot dari representasi kata dari sebuah sekuens, hanya baris embedding dari kata-kata yang muncul yang akan diupdate [7].

3. Bidirectional Long Short-Term Memory (BiLSTM) Network

Metode BiLSTM bekerja dengan memproses urutan data dari dua arah: maju (dari awal ke akhir) atau LSTM *Forward* dan mundur (dari akhir ke awal) atau LSTM *Backward*. Dengan hasil vektor embedding sebagai input, sekuens ini akan diproses oleh LSTM *Forward* terlebih dahulu, kemudian akan diproses oleh LSTM Backward secara terpisah. Kemudian hasil hidden state dari kedua LSTM tersebut akan dikonkatenasi menjadi output BiLSTM. Tentunya, layer BiLSTM ini terdiri dari cell-cell LSTM [8].

LSTM mengatasi tantangan pemahaman konteks sekuensial melalui arsitektur yang lebih rumit, untuk secara efektif menangkap dan mempertahankan informasi penting. Melalui pengelolaan arus informasi melalui gerbang yang berbeda seperti gerbang input, forget, dan output, LSTM memiliki kemampuan untuk secara selektif menyimpan atau membuang data, sehingga unggul dalam pemodelan dan prediksi rangkaian di berbagai bidang pemrosesan bahasa alami [9]. Arsitektur sebuah cell LSTM dapat dilihat pada Gambar 2 dibawah ini.



Gambar 2. Arsitektur LSTM

Arsitektur LSTM menangani Memori Jangka Pendek serta Memori Jangka Panjang. Agar kita dapat melihat algoritma dengan mudah, maka digunakan konsep gerbang :

a. Forget Gate

Informasi yang tidak lagi berguna dalam status sel dihapus dengan gerbang lupa. Seperti di Gambar 2.2, dua masukan xt (masukan pada waktu tertentu) dan ht-1 (keluaran sel sebelumnya) dimasukkan ke gerbang dan dikalikan dengan matriks bobot diikuti dengan penambahan bias. Hasilnya dilewatkan melalui fungsi sigmoid yang memberikan keluaran biner. Jika untuk status sel tertentu, keluarannya adalah 0, bagian informasi dilupakan dan untuk keluaran 1, informasi disimpan untuk penggunaan di masa mendatang [9]. Persamaan fungsi *forget gate* yaitu:

$$ft = \sigma(Wf\,[ht - 1.xt] + bf) \qquad (1)$$

Keterangan :
xt : input variable x ke t; σ : Fungsi Sigmoid; Wf : Matriks bobot dari forgate gate; ht-1 : hidden state sebelumnya; bf : Bias forget state; dan ft : forget gate .

b. Input Gate dan Cell State Candidate

Penambahan informasi baru ke dalam *cell state* dilakukan melalui sebuah gerbang masukan. Pada tahap ini, nilai yang akan disimpan diatur oleh fungsi sigmoid yang berperan sebagai penyaring terhadap informasi dari hidden state sebelumnya menggunakan input ht-1 dan

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

xt. Kemudian, sebuah vektor dibuat menggunakan fungsi tanh yang memberikan keluaran dari -1 hingga +1, yang berisi semua nilai yang mungkin dari ht-1 dan xt [9]. Persamaan fungsi gerbang input dan *cell state candidate* yaitu:

$$it = \sigma(Wi\,[ht - 1.xt] + bi) \quad (2)$$

$$\dot{C}t = tanh(W\dot{C}t[ht - 1.xt] + bCt) \quad (3)$$

Keterangan :
Ċt : kandidat cell; σ : fungsi sigmoid t; tan h : fungsi aktivasi; xt : input variable x ke t; Wi : Bobot dari input gate; WĊt : Bobot dari cell aktivasi; ht-1 : hidden state sebelumnya; bi : Bias input state; bĊt : Bias cell aktivasi; dan it : input gate.

c. Cell State dan Output Gate

Pada bagian gerbang ini sel LSTM akan mengambil informasi yang berguna dari status sel pada *time step* tersebut untuk menghasilkan keluaran pada waktu itu. Pertama, sebuah *cell state* (Ct) dihasilkan dengan menerapkan hasil *forget gate* pada *cell state* sebelumnya dan *input gate* pada *cell state candidate*. Hasil *cell state* ini akan diterapkan fungsi tanh. Selain itu sebuah informasi cell akan diatur menggunakan masukan ht-1 dan xt pada *output gate* (ot). Akhirnya, nilai-nilai vektor dan nilai-nilai yang diatur dikalikan sebagai keluaran dan masukan ke sel berikutnya [9]. Persamaan untuk gerbang keluaran dan sel yaitu :

$$ot = \sigma\,(Wo\,[ht - 1.xt] + bo) \quad (4)$$

$$Ct = ft * Ct - 1 + it * \dot{C}t \quad (5)$$

$$ht = ot * tanh(Ct) \quad (6)$$

Keterangan :
ft : Forget gate; it : input gate; Ct-1 : cell gate sebelumnya; σ : Fungsi sigmoid; Wo : Bobot dari output gate; tan h : Fungsi aktivasi; ht-1 : hidden state sebelumnya; xt : input variable x ke t; bo : Bias output state; dan Ct : Bobot dari cell gate.

4. Layer Attention

Dalam analisis sentimen tidak semua kata dalam sekuens memiliki kontribusi yang sama terhadap sebuah sentimen. Mekanisme atensi merupakan metode yang memungkinkan model untuk memberikan fokus lebih besar pada bagian tertentu dari urutan input yang dianggap paling relevan terhadap tugas analisis sentimen. Mekanisme ini memberikan bobot perhatian (*attention weights*) pada setiap kata, sehingga model dapat memahami konteks hubungan antar kata dengan lebih baik. *Global attention* digunakan untuk menonjolkan fitur-fitur utama dari kata penting, sekaligus mengurangi pengaruh dari kata yang kurang relevan. Berikut persamaan untuk skor atensi (u_t), nilai probabilitas (a_t), dan vektor akhir atensi (v) *global attention*:

$$u_t = tanh(h_t.W_t + b_\alpha) \quad (7)$$

$$\alpha_t = \frac{exp(u_t)}{\sum_{t'} exp(u_{t'})} \quad (8)$$

$$v = \sum_t \alpha_t h_t \quad (9)$$

Keterangan :
h_t = Status tersembunyi (hidden state) dari LSTM/BiLSTM pada langkah waktu t;
Wt = Matriks bobot yang dapat dilatih untuk mentransformasikan h_t;
$b\alpha$ = Bias yang membantu dalam menggeser transformasi;
tanh = Fungsi aktivasi non-linear yang menambahkan kompleksitas pada transformasi;
u_t = Skor atensi (sebelum softmax) untuk status tersembunyi h_t.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

Persamaan ini mengubah setiap status tersembunyi ht dari BiLSTM menjadi skor awal ut, yang mengukur pentingnya status tersembunyi tersebut. Langkah berikutnya adalah menerapkan fungsi softmax untuk menormalkan skor ini menjadi bobot perhatian αt. Skor ini menentukan seberapa besar fokus yang harus dimiliki setiap status tersembunyi dalam vektor konteks akhir. Output akhir "v" adalah vektor konteks yang dihitung sebagai jumlah tertimbang dari semua status tersembunyi dari "u". Ini mewakili informasi yang paling relevan dari seluruh urutan [10].

5. Layer Flatten

Lapisan *Flatten* adalah lapisan pembentukan ulang data dalam jaringan, digunakan untuk mengubah data multidimensi menjadi vektor 1D sehingga dapat diteruskan ke lapisan fully connected (dense). Outputnya adalah vektor 1D, versi input yang diratakan [11].

6. Layer Klasifikasi Dense

Vektor 1D ini diumpankan ke lapisan *dense* dengan aktivasi softmax untuk memperoleh distribusi probabilitas klasifikasi. Prediksinya adalah vektor $y \in \mathbb{R}^2$ dengan probabilitas positif, netral, atau negatif. Karena itu, dense layer klasifikasi ini hanya memiliki 3 neuron [12]. Operasi yang dilakukan dense layer adalah :

$$y = f(Wx + b) \tag{10}$$

Keterangan :
b : bias; f : fungsi aktivasi softmax; W : matriks bobot; y : output dari layer; dan x : input vector.

2.5 Implementasi Low-Rank Adaptation (LoRA)

LoRA memungkinkan kita untuk melatih beberapa lapisan dalam jaringan neural network secara tidak langsung dengan mengoptimalkan matriks dekomposisi rank dari perubahan layer selama adaptasi, sambil tetap menjaga bobot yang telah dilatih sebelumnya tetap beku. LoRA membuat pelatihan lebih efisien dan mempermudah menggunakan optimizer adaptif karena kita tidak perlu menghitung gradien atau mempertahankan status *optimizer* untuk sebagian besar parameter. Sebaliknya, kita hanya mengoptimalkan matriks peringkat rendah yang disisipkan dan jauh lebih kecil [5].

Secara garis besar, metode LoRA menghasilkan input + Δx. Pada matriks bobot yang telah dilatih sebelumnya $W_0 \in R^{d \times k}$,, membatasi pembaruan bobot dengan merepresentasikan matriks bobot tersebut dengan dekomposisi peringkat rendah $W_0 + \Delta W = W_0 + BA$, di mana $B \in R^{d \times k}$, $A \in R^{d \times k}$, dan peringkat r << min(d, k). Selama pelatihan, W_0 dibekukan dan tidak menerima pembaruan gradien, sementara A dan B berisi parameter yang dapat dilatih. Perhatikan bahwa baik W_0 maupun ΔW = BA dikalikan dengan input yang sama, dan veckor output masing-masing dijumlahkan berdasarkan koordinat. Untuk h = $W_0\boldsymbol{x}$, forward pass yang dimodifikasi menghasilkan :

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{11}$$

LoRA menggunakan inisialisasi Gaussian acak untuk A dan nol untuk B, jadi ΔW = BA adalah nol pada awal pelatihan [5]. Terdapat beberapa perbedaan dengan model tanpa LoRA. Karena memanfaatkan layer Low-Rank Adaptation (LoRA) untuk mengerti semantik kalimat, di model ini juga meng-*freeze* layer BiLSTM agar pelatihan lebih efisien. Namun, karena layer BiLSTM belum di latih terlebih dahulu maka diimplementasikan sebuah start epoch yaitu beberapa epoch awal dimana BiLSTM dilatih sebelum di-*freeze*/tidak diupdate. Start epoch ini akan dicari nilai optimalnya dalam *hyperparameter tuning*. Pada LoRA layer ini, akan dilatih kedua matriks A dan B pada tiap epoch.

2.6 Hyperparameter Tuning

Optimizer yang digunakan dalam proses pelatihan adalah Adam. Algoritma Adaptive moment estimation (Adam) berfungsi sebagai algoritma optimasi alternatif dari stochastic gradient descent (SGD). Algoritma ini memperbarui bobot jaringan secara bertahap berdasarkan

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

data pelatihan [13]. Optimizer ini melakukan langkah-langkah berikut saat update parameter/bobot:

1.) Menghitung momen pertama (rata-rata gradien/mt) dengan menggunakan gradien bobot (gt)
2.) Menghitung momen kedua (rata-rata kuadrat gradien/vt) dengan menggunakan gradien bobot (gt)
3.) Melakukan koreksi bias untuk momen pertama (m^t) dan kedua (v^t)
4.) Memperbarui parameter (wt)

Dalam pelatihan, dibutuhkan juga pelatihan *hyperparameter*. Tujuannya adalah untuk menemukan kombinasi nilai *hyperparameter* yang menghasilkan kinerja model terbaik. Proses ini melibatkan evaluasi kinerja model dengan berbagai kombinasi *hyperparameter* pada set validasi. Berikut pada Tabel 1 adalah beberapa *hyperparameter* penting yang perlu di-tuning dalam model Attention-Based BiLSTM dan LoRA ini.

Tabel 1. Rentang Nilai Hyperparameter

No.	Nama Hyperparameter	Deskripsi	Rentang Nilai
Attention-Based BiLSTM			
1	embedding dimension	Ukuran dimensi vektor embedding (representasi kata)	100-1000 (+100)
2	lstm units	Jumlah unit pada layer BiLSTM (jumlah memori dalam sel)	32-160 (+32)
3	dropout rate	Rasio unit yang di-dropout untuk regularisasi	0.1-0.9 (+0.1)
4	learning rate	Laju pembelajaran untuk update parameter model pada Optimizer Adam	$1x10^{-4} – 9x10^{-4}$, $1x10^{-3} – 9x10^{-3}$, $1x10^{-2} – 9x10^{-2}$, $1x10^{-1} – 9x10^{-1}$
Attention-Based BiLSTM dengan LoRA			
5	lora rank	Rank dari Low-Rank Adaptation (LoRA) layer untuk efisiensi fine-tuning	2, 4, 8, 16, 32, 64
6	start epoch	Epoch awal sebelum mem-freeze layer BiLSTM	1,3,5,7,10

Langkah *hyperparameter tuning* diawali dengan menentukan rentang nilai untuk setiap *hyperparameter*. Dilatih model dengan berbagai kombinasi *hyperparameter* pada set pelatihan. Setelah itu, dipilih hasil nilai *hyperparameter* dengan akurasi terbaik. Optimasi *hyperparameter* ini dapat memanfaatkan teknik optimasi *hyperparameter Bayesian*. Di sini, akan digunakan

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

model surrogate seperti Tree-structured Parzen Estimator (TPE) yang mencoba memprediksi kinerja parameter (seperti loss), dan digunakan untuk menginformasikan *hyperparameter* selanjutnya. Secara singkatnya, ini akan memulai pelatihan dengan *hyperparameter* acak, dan setelah setiap epoch, hentikan jika kinerjanya tidak baik. Setelah pelatihan penuh selesai, baru akan memperbarui *Bayesian Optimizer*, dan memulai uji coba baru dengan hyperparameter baru yang disarankan. tahapan alur *Bayesian Optimizer* untuk *hyperparameter tuning* dijelaskan sebagai berikut [14].

1. Inisialisasi Hyperparameter Acak (*Random Trial*)

Pada tahap ini, beberapa kombinasi *hyperparameter* pertama dicoba secara acak untuk membangun dataset awal. Dataset ini terdiri dari pasangan (xi,yi) dengan xi yaitu konfigurasi *hyperparameter* dan yi yaitu hasil evaluasi model (misal akurasi).

2. Bangun Model Surrogate

Setelah evaluasi dengan *hyperparameter* acak maka akan membangun model probabilistik untuk memprediksi hasil dari konfigurasi *hyperparameter* baru. Model tersebut dapat berupa model seperti Tree-structured Parzen Estimator (TPE) yang memanfaatkan struktur tree untuk memodelkan *hyperparameter*. TPE memodelkan dua distribusi terpisah :

$$l(x) = \mathrm{p}(\mathrm{x} \mid \mathrm{y} < y*) \tag{12}$$

$$g(x) = \mathrm{p}(\mathrm{x} \mid \mathrm{y} \geq y*) \tag{13}$$

Pada persamaan, l(x) adalah distribusi dari *hyperparameter* yang menghasilkan performa baik, sementara g(x) adalah distribusi dari *hyperparameter* yang menghasilkan performa buruk atau sama. Ini dipisah menurut $y*$ yaitu threshold quantile dari hasil uji sebelumnya. TPE memodelkan probabilitas konfigurasi *hyperparameter* x, diberikan hasil y (p(x|y)) [14].

3. Fungsi Akuisisi

Fungsi akuisisi kemudian menentukan *hyperparameter* mana yang akan diuji selanjutnya. Fungsi akuisisi yang umum digunakan adalah fungsi *Expected Improvement* (EI) yaitu fungsi membandingkan peluang [14]. TPE memilih nilai *hyperparameter* x yang memaksimalkan rasio antara kedua distribusi, dimana untuk setiap kandidat x dihitung :

$$EI(x) = argmax_{\mathrm{x}} \frac{l(x)}{g(x)} \tag{14}$$

4. Evaluasi Model dengan Hyperparameter Baru

Konfigurasi terbaik yaitu kandidat x dengan skor tertinggi dari fungsi akuisisi dijalankan pada model sesungguhnya. Kemudian, nilainya ditambahkan ke dataset (xi,yi).

5. Iterasi

Setelah evaluasi model dengan *hyperparameter* baru maka akan diupdate model surrogate dan mengulang langkah sebelumnya secara iteratif hingga ditemukan *hyperparameter* optimal. Proses ini diulang hingga mencapai jumlah trial maksimum atau hasil sudah konvergen [14].

Khusus pada *tuning hyperparameter* epoch start, tidak dilakukan dengan iterasi pada model surrogate, namun dilakukan secara manual dimana pelatihan dilakukan untuk setiap nilai dan dilihat hasilnya untuk mencari titik optimal antara nilai akurasi dan waktu pelatihan.

2.7 Evaluasi antar Model

Evaluasi antar model adalah Langkah menilai performa model Attention-Based BiLSTM tanpa LoRA terhadap model Attention-Based BiLSTM dengan LoRA dalam tugas analisis sentimen ulasan wisata ini. Evaluasi dilakukan melalui perbandingan nilai akurasi, presisi, recall, dan skor f1 di hasil analisis sentimen pada data testing [15]. Dibawah ini diberikan persamaan untuk menghitung nilai-nilai tersebut:

a. Accuracy

Merupakan ukuran yang menunjukkan seberapa dekat hasil prediksi model terhadap nilai sebenarnya.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (15)$$

b. Precision
Menunjukkan proporsi data yang benar diambil positif dari seluruh data yang diklasifikasikan sebagai positif oleh model.

$$Precision = \frac{TP}{TP + FP} \quad (16)$$

c. Recall
Menunjukkan kemampuan model dalam menemukan seluruh data yang benar-benar bernilai positif di dalam dataset.

$$Recall = \frac{TP}{TP + FN} \quad (17)$$

d. F1-Score
Merupakan ukuran gabungan antara *precision* dan *recall* yaitu nilai seimbang antara keduanya.

$$F1\ Score = \frac{2\ X\ Precision\ X\ Recall}{Precision + Recall} \quad (18)$$

Di metrik-metrik evaluasi model tersebut, TP (*True Positive*) menunjukkan jumlah data diklasifikasikan positif yang kemudian memang benar positif, kemudian TN (*True Negative*) menunjukkan sampel diklasifikasikan negatif dan memang benar negatif. FP (*False Positive*) adalah data yang diambil positif namun seharusnya negatif, dan FN (*False Negative*) adalah data yang diambil negatif tapi harusnya positif [15]. Kemudian dalam membandingkan efisiensi model dengan dan tanpa LoRA penulis menggunakan nilai total waktu pelatihan (*training time*) dalam satuan detik dan penggunaan memori RAM saat training/CPU memory (*memory usage*) dalam Mb. Waktu pelatihan lebih pendek mengindikasikan efisiensi lebih tinggi.

3. HASIL DAN PEMBAHASAN

Pada penelitian ini dibandingkan nilai akurasi model, serta *precision*, *recall*, dan skor f-1, pada tiap besaran *hyperparameter*, yaitu melakukan proses tuning. Serta, juga dibandingkan performa, waktu pelatihan, dan penggunaan memori antara model Attention-Based BiLSTM dan model Attention-Based BiLSTM dengan LoRA. Hasil penelitian pada tiap tahapan dijelaskan berikut.

3.1 Pengumpulan Data

Crawling data dilakukan dengan tools scraping python. Data tersebut merupakan data berbahasa Indonesia yang diperoleh dari ulasan-ulasan pengguna di situs web Tripadvisor.com mengenai destinasi-destinasi wisata di Kabupaten Badung, Buleleng dan Kawasan Nusa Dua di Provinsi Bali. Data ini dikumpulkan pada tanggal 5-7 Mei 2025 berjumlah 4966 ulasan tempat wisata.

3.2 Preprocessing

Tahapan ini bertujuan untuk membersihkan data agar mudah diproses pada tahap-tahap berikutnya. Contoh hasil tahapan ini pada suatu sampel di dataset dapat dilihat dibawah ini.

Input Data Awal : Saat sore, istriku memutuskan untuk menikmati suasana Pantai seminyak dgn sunset!

Input Setelah Preprocessing : [‘sore’, ‘istri’, ‘putus’, ‘nikmat’, ‘suasana’, ‘pantai’, ‘seminyak’, ‘sunset’]

Selain itu, pada tahap filtering, sejumlah 946 sampel data dihapus hingga hasilnya tersisa 4020 jumlah data yaitu terdiri atas 1441 data positif, 1571 data negatif, dan 1008 data netral.

3.3 Data Splitting

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

Sebelum dilakukan data splitting, data akan melalui beberapa tahap modifikasi input agar dapat digunakan dalam training dan testing. Pertama, proporsi data antar ketiga label (positif, negatif, netral) diseimbangkan agar jumlahnya sama. Hal ini dilakukan dengan random sampling dimana data secara acak diambil hingga berjumlah sama antar tiap label. Hasilnya, data yang berjumlah 4020 sebelumnya hanya diambil 1008 jumlah data dari tiap label (positif, negatif, netral) sehingga berjumlah 3024. Sisa data tersebut diambil sejumlah 856 data layak sebagai data testing untuk evaluasi nantinya. Kemudian, data tersebut diformat agar dapat diinput ke model analisis sentimen. Data teks tersebut dikonversi menjadi bentuk sekuens dimana sebuah indeks token dibangun untuk setiap kata, kemudian setiap data teks dikonversi menjadi sekuens yang berisi indeks kata yang muncul secara berurut. Hasilnya, jumlah token yang diindeks ada 5508 token berbeda. Kemudian, agar panjang sekuens tiap sampel data berukuran sama, dilakukan padding dimana sekuens yang terlalu pendek ditambahkan token padding agar berukuran sama dengan sekuens yang panjang dimana panjang sekuens tiap data berukuran 245. Selanjutnya, tiap label dikonversi menjadi bentuk numerik, "positive" menjadi 0, "negative" menjadi 1, dan "neutral" menjadi 2. Setelah pemformatan input ini, barulah data di split dengan rasio 80% *training* data dan 20% *validation* data. Hasilnya, data latih berjumlah 2419 dan data uji berjumlah 605. Contoh hasil tiap tahap pemformatan ini dapat dilihat pada Tabel 2 dibawah ini.

Tabel 2. Hasil Pemformatan Input

Teks ke Sekuens		
No.	Teks	Sekuens
1.	lumayan tempat bayar pandang saji jangan lupa bawa topi kacamata hitam ya	[231, 114, 62, 12, 441, 70, 220, 57, 322, 133, 224, 1456]
Sekuens ke Padded Sekuens		
No.	Sekuens	Padded Sekuens (ukuran 245)
1.	[231, 114, 62, 12, 441, 70, 220, 57, 322, 133, 224, 1456]	[231, 114, 62, 12, 441, 70, 220, 57, 322, 133, 224, 1456 ... 0, 0 , 0]
Label Asli ke Label Numerik		
No.	Label Asli	Label Numerik
1.	positive	0

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

3.4 Hyperparameter Tuning

Ada enam jenis *hyperparameter* yang dicari nilai optimalnya, yaitu jumlah dimensi embedding, jumlah unit LSTM, *dropout rate*, *learning rate*, rank LoRA, dan epoch start. Di sini epoch start menggunakan tuning manual dengan melihat hasil pelatihan per nilai, sehingga tidak didapatkan rata-rata akurasi validasinya. Selain epoch start, *tuning hyperparameter* ini menggunakan sebuah *Bayesian Optimizer* berupa Tree-structured Parzen Estimator (TPE) untuk mengusulkan nilai-nilai selanjutnya yang akan diuji pada tuning trial. Dalam implementasinya, metode tersebut menggunakan *Bayesian Optimizer* dari library keras_tuner.

Terdapat suatu parameter Early Stopping sehingga apabila dalam tuning mengalami suatu penurunan loss selama tiga trial berturut-turut, tuning dihentikan agar mencegah redundansi. Di sini nilai *hyperparameter* optimal ditentukan dari nilai yang dapat mencapai akurasi tertinggi, namun juga melihat rata-rata akurasi validasi pada setiap trial yang menggunakan nilai tersebut guna mengevaluasinya. Hasil nilai *hyperparameter* optimal yang memperoleh *validation accuracy* tertinggi beserta rata-rata akurasi validasi pada nilai-nilai tersebut dipaparkan di Tabel 3 berikut ini.

Tabel 3. Hasil Nilai Hyperparameter Tuning Optimal

Hyperparameter	Nilai Hyperparameter	Validation Accuracy di Nilai Tersebut	Rata-Rata Akurasi Validasi di Nilai Tersebut
Jumlah Dimensi Embedding	500	0.758	0.717
Jumlah Unit LSTM	96	0.758	0.742
Dropout Rate	0.1	0.758	0.705
Learning Rate	0.001	0.758	0.754
Rank LoRA	8	0.761	0.752
Epoch Start	5	0.759	-

- Jumlah Dimensi Embedding

Setelah melakukan tuning, ditemukan bahwa jumlah dimensi embedding pada embedding layer ini berdampak pada kemampuan model memahami konteks tiap kata serta hubungan antar kata dengan sentimen kalimat. Dimensi embedding yang tinggi akan memberi konteks kata yang lebih baik, namun akan meningkatkan waktu pelatihan.

Selama tuning ditemukan rata-rata validasi akurasi bervariasi. Pada nilai 100-200, rata-rata akurasi berkisar 0.67-0.70. Kemudian di nilai 300-500 naik antara 0.7-0.74. Pada jumlah embedding 600-800 rata-rata mencapai titik jenuh yaitu sekitar 0.74-0.75, di sini ditemukan akurasi tertinggi. Dan turun kembali pada nilai 900-1000, yaitu antara 07-0.72. Setelah tuning, ditemukan akurasi tertinggi (0.758) dicapai pada kombinasi *hyperparameter* dengan dimensi embedding 500, sehingga nilai tersebut optimal.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

- Jumlah Unit LSTM

Jumlah unit sel LSTM pada layer BiLSTM ini berdampak pada kemampuan model mempelajari pola dan dependensi antar kata dalam kalimat/sekuens. Hasil menunjukkan bahwa nilai ukuran yang menghasilkan performa tinggi ada diantara 96-160.

Pada nilai 32-64, rata-rata akurasi berkisar 0.684-0.696. Kemudian di nilai 96 unit mencapai nilai akurasi tertinggi. Pada nilai 128-160, rata-rata akurasi validasi mengalami penurunan hingga 0.707. Setelah tuning, ditemukan bahwa akurasi tertinggi (0.758) dicapai pada jumlah unit LSTM 96, sehingga nilai tersebut yang dianggap optimal.

- Dropout Rate

Dropout rate ini berpengaruh pada kemampuan model agar dapat generalisasi ke data baru. Dropout yang tinggi dapat meningkatkan regularisasi model namun nilai yang terlalu tinggi dapat menyebabkan underfitting. Hasil menunjukkan bahwa nilai ukuran yang menghasilkan performa tinggi ada diantara 0.1-0.6, dimana rata-rata akurasi validasi berkisar antara 0.70-0.76. Namun turun setelah dropout 0.7, hingga rata-rata 0.662. Setelah tuning, ditemukan akurasi tertinggi (0.758) dicapai pada kombinasi *hyperparameter* dengan dropout rate 0.1, sehingga nilai tersebut dianggap optimal.

- Learning Rate

Learning rate menentukan seberapa jauh langkah pembaruan bobot dilakukan berdasarkan penghitungan gradien loss. Ini berdampak pada seberapa cepat model belajar. Learning rate yang tinggi akan mempercepat pelatihan namun nilai yang terlalu tinggi dapat menyebabkan model yang tidak stabil dan tidak bisa konvergen. Sebaliknya jika terlalu rendah maka dapat memperlambat training.

Pertama tuning dilakukan secara logaritmik dengan skala 10 antara 0-1. Hasil menunjukkan bahwa nilai ukuran yang menghasilkan performa tinggi ada diantara 0.001-0.009, dengan nilai rata-rata akurasi validasi tertinggi diantara 0.746-0.741, dan mencapai nilai akurasi tertinggi (0.758). Pada nilai 0.0001-0.0009, nilai rata-rata akurasi validasi masih rendah diantara 0.729-0.736. Di nilai 0.01-0.09, rata-rata akurasi validasi kembali rendah antara 0.734-0.718. Dan mencapai titik rendah di nilai learning rate 0.1-0.3, yaitu antara 0.574-0.517.

Selanjutnya, dilakukan tuning lebih spesifik yaitu tuning linear antara 0.001-0.009. Setelah tuning lebih spesifik, ditemukan nilai rata-rata akurasi validasi tertinggi berada diantara 0.001 dan 0.003 yaitu memperoleh rata-rata yang sama 0.754. Di nilai lebih tinggi 0.004-0.009, nilai rata-rata turun signifikan antara 0.727-0.734. Akurasi tertinggi (0.758) dicapai pada kombinasi *hyperparameter* dengan learning rate 0.003, sehingga nilai tersebut optimal.

- Rank LoRA

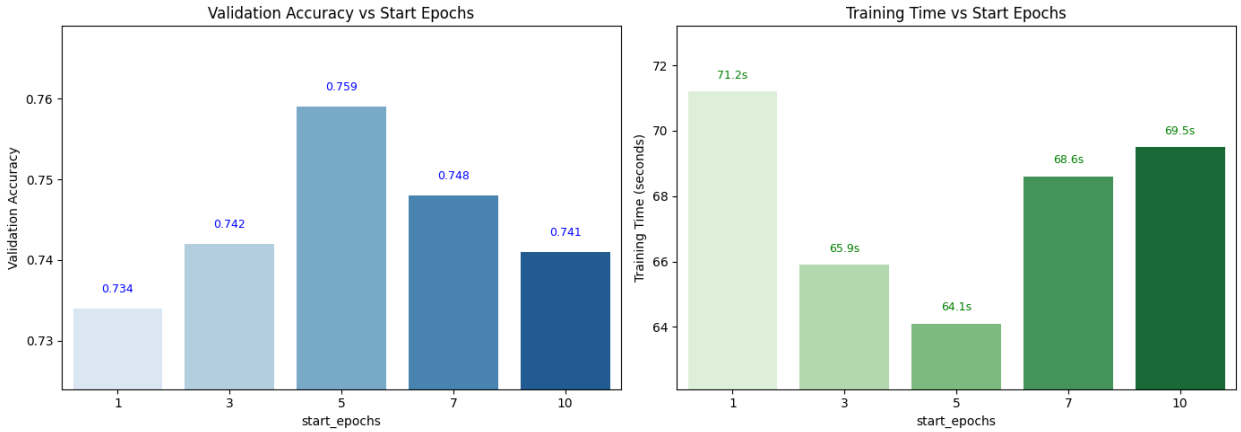
LoRA rank yang lebih kecil berdampak pada model yang lebih efisien dan ringan sehingga memakan waktu dan memori yang lebih sedikit, namun jika terlalu rendah dapat menyebabkan kualitas representasi dan performa turun. Sebaliknya, LoRA rank yang besar artinya semakin besar kapasitas untuk belajar namun lebih banyak parameter yang dilatih sehingga memakan waktu dan memori lebih banyak.

Hasil menunjukkan bahwa nilai ukuran yang menghasilkan performa tinggi ada diantara 2-8, dimana rata-rata akurasi validasi pada LoRA rank 2 yaitu 0.756, LoRA rank 4 turun ke 0.747, namun naik ke 0.752 pada LoRA rank 8. Pada LoRA rank 16-32 rata-rata stagnan diantara 0.747-0.748. Setelah tuning, ditemukan akurasi tertinggi (0.761) dicapai pada kombinasi *hyperparameter* dengan rank 8, sehingga nilai tersebut optimal.

- Epoch Start

Start epoch ini merupakan epoch awal dimana layer BiLSTM masih dapat diupdate bobotnya. Setelah start epoch ini selesai maka layer BiLSTM akan di-freeze. Meningkatkan start epoch ini akan meningkatkan akurasi namun juga meningkatkan waktu pelatihan karena parameter yang akan dilatih semakin banyak. Maka perlu dicari titik start epoch optimal antara akurasi dan efisiensi model. Hal ini dilakukan dengan menampilkan diagram nilai akurasi dan waktu pelatihan pada tiap start epoch, yang dapat dilihat pada Gambar 3 berikut.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089



Gambar 3. Diagram Nilai Validation Accuracy dan Training Time terhadap Epoch Start

Pada bagian ini dilakukan dua kali tuning untuk akurasi dan efisiensi yaitu keccepatan pelatihan. Setelah tuning, ditemukan titik optimal antara akurasi (0.759) dan waktu pelatihan (64.1 detik) yang dicapai pada kombinasi *hyperparameter* dengan epoch start 5. Selain itu, nilai akurasi validasi ini lebih tinggi daripada epoch start 3 dan 7 yaitu mencapai nilai 0.742 dan 0.748. Dan waktu pelatihan yang lebih rendah dicapai dibanding nilai epoch start 3 dan 7 yaitu 65.9 detik dan 68.6 detik. Sehingga ditemukan bahwa epoch start 5 adalah nilai optimal.

3.5 Evaluasi Model

Dievaluasi kedua metode analisis sentimen ulasan tempat wisata di Bali yaitu metode Attention-Based BiLSTM tanpa LoRA dan Attention-Based BiLSTM dengan LoRA. Kedua model ini dibandingkan dengan menggunakan metrik performa (*precision, recall*, skor f1, akurasi) dan efisiensi (*training time* dan *memory usage*). Model dievaluasi dengan data testing berjumlah 856 sampel.

1. Evaluasi Attention-Based BiLSTM

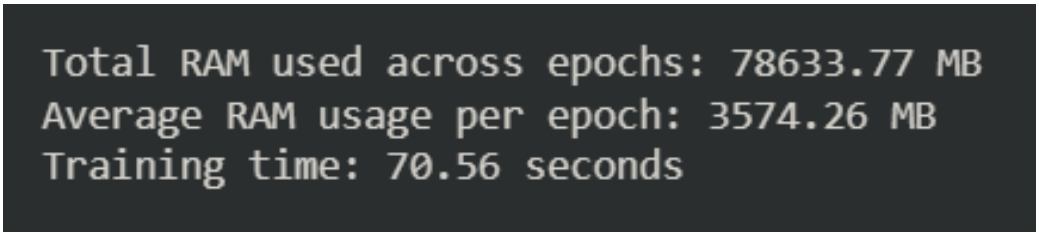
Tabel 4. Skor Performa Attention-Based BiLSTM

Jenis Nilai	Presisi	Recall	Skor F1	Akurasi
Macro Average	0.71451	0.73111	0.71522	0.78037
Weighted Average	0.81605	0.78037	0.79416	

Metrik evaluasi dihitung melalui dua jenis rata-rata, *weighted average* dan *macro average*. *Macro average* adalah penghitungan rata-rata tanpa melihat jumlah sampel antar kelas. Sementara *weighted average* menghitung rata-rata dengan memperhitungkan support atau jumlah sampel antar kelas, sehingga memberi kepentingan lebih pada kelas dengan sampel lebih banyak. Dapat dilihat pada Tabel 4 bahwa nilai *precision* (0.71), *recall* (0.73), dan skor f1 (0.71) jika dihitung dengan *macro average* lebih rendah, namun hal ini disebabkan jumlah sampel antar kelas yang tidak seimbang, dimana kelas netral berjumlah sedikit dibanding kelas lainnya.

Jika dilihat pada *weighted average* hasil *precision* lumayan tinggi yaitu 0,816. Hal ini juga dapat dilihat pada nilai recall dan skor f1, dimana model mendapatkan recall senilai 0.78 dan skor f1 senilai 0.794. Akhirnya, nilai akurasi model keseluruhan didapatkan senilai 0.78. Kemudian, kita dapat melihat skor efisiensi model.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089



Gambar 4. Skor Efisiensi Attention-Based BiLSTM

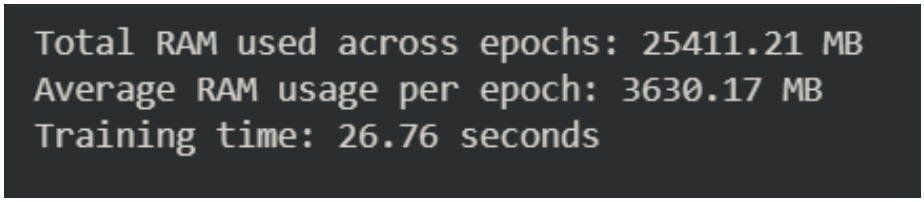
Dapat dilihat pada Gambar 4 bahwa waktu pelatihan model ini cukup pendek yaitu 70,56 detik. Kemudian total RAM atau memori yang digunakan pada keseluruhan epoch adalah sekitar 78633 Mb sementara rata-rata memori yang digunakan pada tiap epoch adalah 3574 Mb.

2. Evaluasi Attention-Based BiLSTM dengan Low-Rank Adaptation

Tabel 5. Skor Performa Attention-Based BiLSTM dengan LoRA

Jenis Nilai	Presisi	Recall	Skor F1	Akurasi
Macro Average	0.72197	0.72913	0.72278	0.79907
Weighted Average	0.82007	0.79907	0.80792	

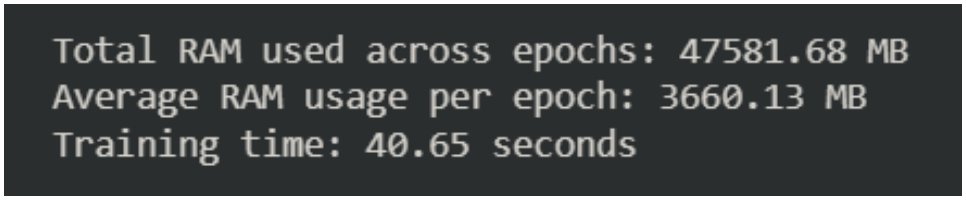
Dapat dilihat pada Tabel 5 bahwa nilai precision (0.72), recall (0.729), dan skor f1 (0.722) pada *macro average* juga lebih rendah akibat ketidakseimbangan jumlah kelas. Jika dilihat pada *weighted average* hasil *precision* yaitu perbandingan hasil prediksi dengan data yang diminta yaitu 0,82. Hal ini juga dapat dilihat pada nilai recall dan skor f1, dimana model mendapatkan recall senilai 0.79 dan skor f1 senilai 0.8. Nilai akurasi model tersebut didapatkan senilai 0.799. Selanjutnya kita dapat melihat skor efisiensi.



Gambar 5. Skor Efisiensi Attention-Based BiLSTM dengan LoRA sebelum Layer Freeze

Dapat dilihat pada Gambar 5 bahwa waktu pelatihan model ini pada epoch ke 5 yaitu 26,76 detik. Kemudian total RAM atau memori yang digunakan pada keseluruhan epoch adalah sekitar 25411 Mb sementara rata-rata memori yang digunakan pada tiap epoch adalah 3630 Mb. Kemudian, setelah epoch di-*freeze* dapat dilihat bahwa total memori dan waktu pelatihan ditampilkan pada Gambar 6 berikut.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089



Gambar 6. Skor Efisiensi Attention-Based BiLSTM dengan LoRA Setelah Layer Freeze

Dapat dilihat pada Gambar 6, yaitu waktu pelatihan model setelah epoch ke 5 (dari epoch ke 6) sampai epoch ke 20 yaitu 40,65 detik. Sehingga total waktu pelatihan dari awal sampai akhir yaitu 67,41 detik. Kemudian total RAM atau memori yang digunakan pada epoch setelah *freeze* ini adalah sekitar 47581 Mb sementara rata-rata memori yang digunakan pada tiap epoch adalah 3660 Mb. Sehingga total memori yang digunakan dari awal sampai akhir adalah sekitar 72992 Mb.

Sehingga dapat diambil penilaian bahwa model Attention-Based BiLSTM dengan LoRA lebih efisien dimana waktu pelatihan berkurang relatif 4.47% dari 70.56 detik ke 67.41 detik. Penggunaan memori berkurang relative 7.18% dari 78533 Mb ke 72992 Mb. Serta, metode tersebut cukup berpengaruh pada performa model dimana akurasi naik dari 0.78037 ke 0.79907. Setelah melihat hasil jumlah data prediksi model tersebut, ditemukan model Attention-Based BiLSTM dengan LoRA berhasil mengambil 317 jumlah data dengan sentimen positif, 310 dengan sentimen negatif, dan 57 sampel dengan sentimen netral secara akurat. Namun sejumlah 66 dari kelas positif, 48 dari kelas negatif, dan 59 dari kelas netral yang diklasifikasikan salah.

4. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian, dapat diambil sebuah kesimpulan yaitu model Attention-Based BiLSTM dengan LoRA cukup berpengaruh pada performa model, dan dapat menghasilkan analisis sentimen yang akurat dengan pelatihan yang cukup efisien. Hasil evaluasi menunjukkan bahwa model ini dapat menghasilkan nilai akurasi sebesar 0.79907. Selain itu dapat diambil beberapa kesimpulan lainnya :

a. Hasil *hyperparameter tuning* pada Attention-Based BiLSTM dan Attention-Based BiLSTM dengan LoRA mendapatkan nilai optimal 500 untuk dimensi embedding, 96 untuk jumlah unit LSTM, 8 untuk rank Low-Rank Adaptation (LoRA), dan 5 untuk start epoch.
b. Model Attention-Based BiLSTM dengan LoRA menggunakan memori sebanyak 72992 Mb dan waktu pelatihan selama 67,41 detik selama pelatihan menggunakan 3024 jumlah data ulasan wisata.
c. Berdasarkan nilai performa dan efisiensi, metode Attention-Based BiLSTM dengan LoRA mempunyai nilai relatif lebih tinggi dibandingkan dengan metode Attention-Based BiLSTM.
d. Berdasarkan hasil analisis sentimen pada data testing dari Attention-Based BiLSTM dengan LoRA berhasil mengambil 317 jumlah data bersentimen positif, 310 data sentimen negatif, dan 57 data sentimen netral secara akurat.

Secara keseluruhan, penelitian ini memberikan kontribusi positif bagi pengembangan model analisis sentimen ulasan wisata berbahasa Indonesia di Bali. Hasil penelitian ini dapat menjadi pedoman untuk pengembangan model analisis sentimen yang lebih optimal di masa mendatang.

DAFTAR PUSTAKA

[1] Y. Soritua, “Analisis Peran Sektor Pariwisata Menjadi Pendapatan Utama Daerah (Studi Banding: Peran Sektor Pariwisata di Provinsi Bali),” *Jurnal Ilmu Manajemen dan Akuntansi*, vol. 3, no. 2, pp. 1–7, 2017, doi: doi.org/10.33366/ref.v3i2.506.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089

[2] B. Foley, "Transparency report 2025: Fighting fraud." Accessed: Mar. 23, 2025. [Online]. Available: https://www.tripadvisor.com/TransparencyReport2025#article

[3] Z. Shaikh and S. Ramadass, "Unveiling deep learning powers: LSTM, BiLSTM, GRU, BiGRU, RNN comparison," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 35, p. 263, Nov. 2024, doi: 10.11591/ijeecs.v35.i1.

[4] M. Ainur Rohman, Suhartono, and T. Chamidy, "Bidirectional GRU dengan Attention Mechanism pada Analisis Sentimen PLN Mobile," *Techno.Com: Jurnal Teknologi Informasi*, vol. 22, no. 2, pp. 358–372, Mar. 2023.

[5] E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," in *International Conference on Learning Representations*, 2021. doi: doi.org/10.48550/arXiv.2106.09685.

[6] J. Camacho-Collados and M. T. Pilehvar, "On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*, 2018, pp. 40–46. doi: 10.48550/arXiv.1707.01780.

[7] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *International Conference on Learning Representations*, 2013.

[8] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997, doi: 10.1109/78.650093.

[9] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[10] R. Pratiwi, Y. Sari, and Y. Suyanto, "Attention-Based BiLSTM for Negation Handling in Sentimen Analysis," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 14, p. 397, Oct. 2020, doi: 10.22146/ijccs.60733.

[11] P. Klęsk, "Understanding the Flows of Signals and Gradients: A Tutorial on Algorithms Needed to Implement a Deep Neural Network from Scratch," *Applied Sciences*, vol. 14, no. 21, p. 9972, Nov. 2024, doi: 10.3390/app14219972.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed. Cambridge: MIT Press, 2016.

[13] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*, San Diego, Jan. 2015. doi: 10.48550/arXiv.1412.6980.

[14] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Conference on Neural Information Processing Systems*, 2011, p. 2546.

[15] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents," in *ACM SIGIR Forum*, New York, 2000, pp. 41–48. doi: 10.1145/345508.345545.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15089