BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Peninjauan penelitian terdahulu merupakan langkah penting dalam menyusun penelitian yang sistematis dan dapat dipertanggungjawabkan secara ilmiah. Melalui kajian ini, gambaran mengenai perkembangan penelitian di bidang yang relevan diperoleh sekaligus mengetahui apa yang sudah diteliti dan apa yang masih memerlukan eksplorasi lebih lanjut. Proses ini juga bertujuan untuk menunjukkan posisi kontribusi penelitian dalam penelitian yang sudah ada, sehingga relevansi dan kebaruan penelitian dapat dijelaskan dengan jelas. Rangkuman studi-studi yang dijadikan acuan dalam penelitian ini disajikan pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

Judul	Metode	Hasil
Multilingual Fake News Detection: A Study on Various Models and Training Scenarios [33]	mBERT, XLM-RoBERTa, dan LASER *embeddings* dievaluasi pada tiga skenario berbeda: monolingual, *multilingual training*, serta *few-shot cross-lingual transfer* dengan variasi komposisi data pelatihan lintas bahasa	mBERT dan XLM-R menunjukkan performa stabil pada skenario multilingual; pelatihan gabungan meningkatkan F1 *few-shot* rata-rata hingga 12 poin dibanding skenario monolingual
Sentiment Analysis of Public Acceptance of Covid-19 Vaccines Types in Indonesia using Naïve Bayes, Support Vector Machine, and Long Short-Term Memory (LSTM) [34]	Naïve Bayes, SVM, dan LSTM dibandingkan secara komparatif untuk klasifikasi sentimen penerimaan publik terhadap vaksin Covid-19 di Indonesia menggunakan data Twitter berbahasa Indonesia	SVM mencapai akurasi tertinggi sebesar 84,89%, diikuti Naïve Bayes pada 84,65% dan LSTM pada 82,97%; SVM terbukti lebih efektif dibandingkan deep learning LSTM pada dataset teks pendek berbahasa Indonesia
One True Pairing: Evaluating Effective Language Pairings for Fake News Detection Employing Zero-Shot Cross-Lingual Transfer BT - Soft Computing and Its Engineering Applications [35]	mBERT dan XLM-R dimanfaatkan sebagai representasi kalimat untuk mengevaluasi efektivitas pasangan bahasa pada deteksi berita palsu *zero-shot*; pengujian dilakukan dengan model SVM dan *neural network* sebagai *classifier*	F1 *few-shot cross-lingual* tertinggi 0,90 pada pasangan Urdu-Bengali; pasangan bahasa *low-resource* yang berkerabat secara tipologis secara konsisten menghasilkan performa transfer yang lebih baik dibandingkan pasangan yang melibatkan bahasa *high-resource* dominan seperti Inggris dan Spanyol
Fake news detection in low-resource languages: A novel hybrid summarization approach [14]	mBERT dikombinasikan dengan pendekatan *hybrid summarization* ekstraktif-abstraktif untuk mengatasi batasan panjang sekuens pada deteksi berita palsu multilingual di lingkungan *low-resource*	Akurasi deteksi meningkat rata-rata 8,3% pada bahasa *low-resource* dibandingkan mBERT standar tanpa pra-pemrosesan *summarization*; performa pada Bahasa Indonesia mencapai 83,7%

Judul	Metode	Hasil
Parameter-efficient fine-tuning for low-resource text classification: a comparative study of LoRA, IA3, and ReFT [23]	Tiga metode PEFT yaitu LoRA, IA3, dan ReFT dibandingkan pada BERT dan RoBERTa untuk tugas klasifikasi teks NLP dalam kondisi data berlabel yang sangat terbatas	LoRA mencatatkan performa tertinggi di antara tiga metode PEFT dengan selisih hanya 2–3% dibanding full *fine-tuning* namun dengan reduksi parameter trainable hingga 140–280 kali lebih sedikit
Comparing Multilingual Language Models on Indic News Headline [36]	LoRA sebagai pendekatan PEFT diterapkan pada model LLM multilingual skala besar untuk klasifikasi judul berita multibahasa, dikomparasikan terhadap model BERT-like yang di-fine-tune secara penuh termasuk XLMRoBERTa dan DistilBERT	LLM dengan LoRA mencapai weighted F1 0,87 melampaui XLMRoBERTa pada 0,84 dan DistilBERT pada 0,82 yang menggunakan *full fine-tuning*
Comparison between parameter-efficient techniques and full fine-tuning: A case study on multilingual news article classification [20]	LoRA, *adapter*, dan BitFit sebagai metode PEFT dievaluasi secara komparatif terhadap *full fine-tuning* pada tiga tugas klasifikasi teks multibahasa dari SemEval 2023 Shared Task 3 dengan panjang teks dan kompleksitas berbeda	PEFT mereduksi *trainable parameter* 140–280 kali lebih sedikit dengan waktu pelatihan lebih singkat 32–44%; performa PEFT kompetitif terhadap *full fine-tuning* pada sebagian besar kondisi multilingual
Fake News Detection Using Explainable Artificial Intelligence BT - Data Science and Emerging Technologies [32]	DistilBERT dikombinasikan dengan LIME dan SHAP sebagai instrumen Explainable AI untuk membangun sistem deteksi berita palsu yang transparan; LIME diterapkan untuk menghasilkan atribusi token lokal per prediksi	DistilBERT mencapai akurasi kompetitif pada dataset LIAR; LIME berhasil mengidentifikasi token berpengaruh yang secara semantik konsisten dengan konten berita palsu
Can AI Outsmart Fake News? Detecting Misinformation with AI Models in Real-Time [37]	BERT dan GPT dikombinasikan dengan XAI berbasis LIME untuk deteksi misinformasi secara *real-time*; LIME digunakan untuk memberikan transparansi atas keputusan model kepada pengguna non-teknis termasuk jurnalis dan pembuat kebijakan	BERT mencapai akurasi tertinggi di antara model yang diuji; LIME secara konsisten mengidentifikasi pola linguistik yang berkorelasi kuat dengan konten yang terklasifikasi sebagai misinformasi
Interpretation of Deep Learning Models in Natural Language Processing for Misinformation Detection with the Explainable AI (XAI) Approach [38]	Tiga metode XAI yaitu SHAP, LIME, dan *attention visualization* diintegrasikan ke dalam *pipeline* deteksi misinformasi berbasis deep learning NLP; evaluasi dilakukan terhadap kontribusi masing-masing metode dalam meningkatkan transparansi keputusan model	Model deep learning NLP mencapai akurasi tinggi dalam deteksi misinformasi; LIME dan SHAP terbukti lebih efektif dibandingkan *attention visualization* dalam menghasilkan penjelasan yang *human-interpretable*
Evaluating Cross-Lingual Classification Approaches Enabling Topic Discovery for Multilingual Social Media Data [39]	mBERT dan DistilBERT dievaluasi secara komparatif untuk klasifikasi dan *topic discovery* pada data media sosial multibahasa dengan berbagai variasi pendekatan *cross-lingual*	mBERT memiliki kemampuan multilingual yang baik dengan akurasi 96.68% pada bahasa Inggris, namun performanya menurun pada skenario *few-shot* untuk bahasa lain (sekitar 50–53% akurasi), sementara pendekatan berbasis BERT/DistilBERT pada

Judul	Metode	Hasil
		pipeline terjemahan menghasilkan performa yang lebih stabil dengan akurasi hingga 97.72% (EN), 79.85% (JA), 86.03% (KO), dan 90.59% (HI) serta tetap unggul dalam kecepatan inferensi
AI-Based Fake News Detection Using Nlp [16]	Model BERT, RoBERTa, dan XLNet dikombinasikan dengan teknik XAI serta *Graph Neural Network* untuk membangun sistem deteksi berita palsu yang terintegrasi; LIME digunakan sebagai komponen penjelasan model pada level token	BERT berbasis *transformer* mencapai akurasi hingga 99,9% pada dataset Kaggle dan di atas 98% pada ISOT; integrasi GNN meningkatkan F1 sebesar 4–7% dibandingkan model *text-only baseline*
AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning [40]	AdaLoRA mengalokasikan budget parameter secara adaptif berdasarkan *importance score* masing-masing matriks bobot menggunakan parameterisasi berbasis SVD; rank yang tidak penting dipangkas secara dinamis selama pelatihan	AdaLoRA meningkatkan F1 sebesar 1,2% pada SQuADv2 dibandingkan LoRA standar dengan kurang dari 0,1% *trainable parameters*; lapisan atas secara konsisten menerima rank lebih tinggi dibandingkan lapisan bawah
BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models [26]	BitFit memodifikasi hanya *bias terms* model (sekitar 0,09% parameter) selama *fine-tuning* sementara seluruh bobot lainnya dibekukan; diuji pada BERT-Base, BERT-Large, dan RoBERTa-Base	BitFit mencapai rata-rata akurasi kompetitif terhadap *full fine-tuning* pada GLUE benchmark (misalnya 90,5% vs 90,6% pada QNLI) dengan hanya 0,09% parameter; mengungguli metode Diff-Pruning dan Adapter pada 4 dari 9 tugas
Low-Rank Adaptation for Multilingual Summarization: An Empirical Study [21]	LoRA diterapkan pada model multilingual untuk *summarization* dengan eksplorasi berbagai strategi *few-shot cross-lingual transfer* termasuk *continued LoRA tuning* dan komposisi dinamis modul LoRA per bahasa	LoRA kompetitif dengan *full fine-tuning* pada data berlimpah dan mengungguli pada skenario *low-data*; *continued LoRA tuning* menghasilkan performa *cross-lingual* terbaik dibandingkan strategi komposisi modul lainnya

Berdasarkan Tabel 2.1, sejumlah studi terkini mendokumentasikan perkembangan signifikan dalam penerapan model *transformer* multilingual untuk tugas deteksi misinformasi dan klasifikasi teks lintas bahasa. Eksperimen yang membandingkan mBERT, XLM-RoBERTa, dan LASER *embeddings* pada tiga skenario pelatihan berbeda menemukan bahwa strategi *multilingual training* terkombinasi secara konsisten meningkatkan kemampuan *few-shot transfer*, dengan peningkatan rata-rata F1 hingga 12 poin dibandingkan skenario monolingual [33]. Evaluasi efektivitas pasangan bahasa pada skenario *zero-shot* dengan

memanfaatkan mBERT dan XLM-R sebagai representasi kalimat mencatatkan F1 tertinggi sebesar 0,90 pada pasangan Urdu–Bengali, membuktikan bahwa pasangan bahasa *low-resource* yang berkerabat secara tipologis menghasilkan *transfer* yang lebih efektif [35]. Pada lingkup klasifikasi teks Bahasa Indonesia, perbandingan komparatif antara Naïve Bayes, SVM, dan LSTM untuk analisis sentimen vaksin Covid-19 menunjukkan bahwa SVM mencapai akurasi tertinggi 84,89%, mengungguli Naïve Bayes (84,65%) dan LSTM (82,97%), yang mengindikasikan bahwa metode *machine learning* klasik masih kompetitif terhadap *deep learning* konvensional pada teks pendek monolingual meskipun belum memanfaatkan arsitektur *transformer pre-trained* maupun skenario *cross-lingual* [34].

Penerapan mBERT yang dikombinasikan dengan strategi *hybrid summarization* menunjukkan peningkatan akurasi rata-rata sebesar 8,3% pada bahasa *low-resource* dengan performa deteksi pada Bahasa Indonesia mencapai 83,7% [14]. Evaluasi komparatif mBERT dan DistilBERT pada klasifikasi teks media sosial multibahasa menunjukkan bahwa DistilBERT mempertahankan kapabilitas klasifikasi yang kompetitif meski dengan kecepatan inferensi yang lebih tinggi, meskipun mBERT masih unggul pada variasi multilingual yang lebih kompleks [39].

Pada aspek efisiensi adaptasi model, tiga metode PEFT yakni LoRA, IA3, dan ReFT yang dibandingkan dalam kondisi data berlabel terbatas menghasilkan temuan bahwa LoRA secara konsisten mempertahankan performa tertinggi dengan selisih terhadap *full fine-tuning* hanya sebesar 2-3% namun dengan pengurangan parameter *trainable* sebesar 140 hingga 280 kali lipat [23]. Validasi lebih lanjut pada klasifikasi teks multibahasa mempertegas posisi ini, di mana model LLM multilingual yang diadaptasi dengan LoRA berhasil mencapai *weighted* F1 sebesar 0,87 dan melampaui XLM-RoBERTa dengan *full fine-tuning* pada 0,84 [37]. Evaluasi PEFT dalam skenario multilingual dan *cross-lingual* pada SemEval 2023 mengkonfirmasi bahwa LoRA dan *adapter* mampu mempertahankan performa yang kompetitif terhadap *full fine-tuning* dengan penghematan sumber daya komputasi yang substansial [20]. AdaLoRA membuktikan bahwa distribusi *rank* adaptif per lapisan secara konsisten melampaui LoRA standar dengan peningkatan F1 sebesar 1,2% pada SQuADv2, di mana lapisan atas dan modul FFN secara

otomatis memperoleh alokasi *rank* yang lebih tinggi [40]. BitFit menunjukkan bahwa dengan hanya 0,09% parameter yang dilatih, performa *fine-tuning* sudah kompetitif terhadap *full fine-tuning* pada mayoritas tugas GLUE *benchmark* [26]. Studi LoRA pada tugas multilingual *summarization* memberikan bukti langsung bahwa LoRA justru mengungguli *full fine-tuning* pada skenario *low-data* dan *cross-lingual transfer* [21].

Integrasi LIME sebagai instrumen *Explainable AI* ke dalam sistem deteksi berita palsu berbasis DistilBERT menunjukkan bahwa model ringan dapat menghasilkan atribusi token yang secara semantik konsisten dengan konten prediksi tanpa modifikasi arsitektur [32]. Studi yang menerapkan BERT dan GPT dengan dukungan LIME untuk deteksi misinformasi secara *real-time* memvalidasi peran XAI dalam menjembatani sistem prediksi kompleks dengan pemangku kepentingan non-teknis [37]. Perbandingan tiga metode XAI utama mengkonfirmasi bahwa LIME dan SHAP lebih unggul dibandingkan *attention visualization* dalam menghasilkan penjelasan yang dapat dipahami manusia [38]. Sistem yang mengintegrasikan *Graph Neural Network* dengan XAI berbasis LIME turut menunjukkan bahwa *transformer* berbasis BERT mampu mencapai akurasi hingga 99,9% pada *dataset* Kaggle dengan penambahan komponen GNN yang meningkatkan F1 sebesar 4-7% [16].

Analisis terhadap seluruh studi yang diidentifikasi menunjukkan bahwa meskipun masing-masing dimensi yakni *multilingual fake news detection*, *adaptive-rank* PEFT, *bias tuning*, dan XAI untuk misinformasi telah mengalami perkembangan signifikan, tidak satu pun dari studi tersebut mengintegrasikan keempat dimensi dalam satu kerangka evaluasi yang terpadu. Kesenjangan pertama terletak pada belum digunakannya PEFT *hybrid* yang mengombinasikan *adaptive-rank* LoRA, *bias tuning*, dan *selective embedding adaptation* secara bersamaan dalam skenario *few-shot cross-lingual hoax detection*. Kesenjangan kedua menyangkut Bahasa Melayu yang hingga kini belum pernah diposisikan sebagai target bahasa dalam evaluasi *few-shot cross-lingual* dengan metode PEFT *hybrid*. Kesenjangan ketiga berkaitan dengan tidak adanya analisis komparatif sistematis antara mBERT, DistilmBERT, dan XLM-RoBERTa yang secara khusus membandingkan *full fine-tuning* dan PEFT *hybrid* untuk deteksi hoaks, disertai

analisis interpretabilitas menggunakan LIME. Penelitian ini dirancang untuk menutup ketiga kesenjangan tersebut secara serempak melalui kerangka eksperimen ARL-LoRA^{+}.

2.2 Tinjauan Teori

2.2.1 Hoaks dan Misinformasi

Penyebaran informasi yang tidak akurat melalui media sosial dan platform daring telah menjadi permasalahan global yang semakin sulit dikendalikan seiring dengan meningkatnya volume konten digital. Gangguan informasi secara konseptual dibedakan menjadi tiga kategori utama, yaitu misinformasi yang merujuk pada penyebaran informasi salah tanpa adanya kesengajaan dari pengirim, disinformasi yang merupakan penyebaran informasi salah secara sadar dan terencana untuk tujuan tertentu, serta malinformasi yang merupakan penyebaran informasi faktual namun dilakukan dengan niat merugikan pihak tertentu [41]. Hoaks sebagai salah satu bentuk disinformasi memiliki karakteristik linguistik yang khas, seperti penggunaan judul yang provokatif, narasi yang berlebihan tanpa didukung sumber kredibel, serta gaya bahasa yang dirancang untuk memancing reaksi emosional pembaca [42]. Perbedaan pola tekstual antara hoaks dan berita kredibel inilah yang memungkinkan pendekatan berbasis pemrosesan bahasa alami untuk mendeteksi hoaks secara otomatis melalui analisis fitur-fitur linguistik pada teks.

2.2.2 *Natural Language Processing* (NLP)

Natural Language Processing (NLP) merupakan cabang kecerdasan buatan yang berfokus pada pengembangan sistem yang mampu membaca, memahami, dan menafsirkan bahasa manusia secara komputasional [43]. Mekanisme kerja NLP melibatkan pemrosesan teks pada beberapa tingkatan, mulai dari analisis sintaksis yang memeriksa struktur tata bahasa, analisis semantik yang mengidentifikasi makna kata dan kalimat, hingga analisis pragmatik yang memperhitungkan konteks penggunaan bahasa [44]. Perkembangan NLP mengalami evolusi dari pendekatan berbasis aturan ke *machine learning* statistik, dan kemudian ke era *deep learning* di mana

arsitektur *transformer* menjadi standar baru karena kemampuan mekanisme *self-attention* dalam menangkap hubungan semantik jarak jauh yang sebelumnya sulit ditangani oleh arsitektur sekuensial seperti LSTM dan CNN [45].

2.2.3 *Cross-Lingual* NLP

Cross-lingual NLP merupakan bidang kajian yang berfokus pada pengembangan model yang mampu memproses teks dari berbagai bahasa dalam satu kerangka terpadu [4]. Ide utamanya adalah membangun representasi semantik yang bersifat *language-agnostic*, di mana vektor representasi suatu teks dapat dibandingkan secara langsung lintas bahasa tanpa bergantung pada struktur leksikal masing-masing bahasa [46]. Pendekatan awal dalam bidang ini memanfaatkan *static embeddings* seperti MUSE dan VecMap, namun pendekatan tersebut memiliki keterbatasan dalam menangkap ambiguitas makna kata yang bergantung pada konteks kalimat, sehingga kemudian digantikan oleh model berbasis *transformer* seperti mBERT dan XLM-RoBERTa yang mampu menghasilkan representasi kontekstual secara dinamis [47].

Salah satu konsep sentral dalam *cross-lingual* NLP adalah *few-shot cross-lingual transfer*, yaitu kondisi di mana model dilatih menggunakan data berlabel dari bahasa sumber dengan disertai sejumlah kecil sampel dari bahasa target, kemudian dievaluasi pada bahasa target tersebut [13]. Efektivitas *transfer* ini dipengaruhi oleh beberapa faktor, di antaranya kedekatan tipologis antara bahasa sumber dan bahasa target, ukuran *vocabulary overlap* antar bahasa, serta kualitas representasi yang dihasilkan oleh model selama *pre-training* [48]. Berbagai studi telah membuktikan efektivitas *cross-lingual transfer* pada beragam domain tugas NLP, termasuk analisis sentimen, deteksi ujaran kebencian, dan identifikasi misinformasi, yang menunjukkan bahwa pendekatan ini dapat diterapkan secara luas untuk berbagai permasalahan klasifikasi teks lintas bahasa [49].

2.2.4 *Few-Shot Learning*

Few-shot learning merupakan konsep dalam *machine learning* yang bertujuan membangun model yang mampu mengenali pola baru berdasarkan sejumlah kecil contoh berlabel, berbeda dengan *supervised learning* konvensional yang umumnya memerlukan ribuan hingga jutaan sampel pelatihan untuk setiap kategori [50]. Pendekatan ini menjadi penting ketika data berlabel sulit diperoleh, baik karena keterbatasan biaya anotasi, kelangkaan data pada domain tertentu, maupun karena bahasa yang ditargetkan merupakan bahasa dengan sumber daya terbatas. Dalam spektrum ketersediaan data, *few-shot learning* berada di antara *zero-shot learning* yang tidak menggunakan data target sama sekali dan *full supervised learning* yang mengandalkan data target dalam jumlah besar, dengan memanfaatkan sejumlah kecil sampel berlabel sebagai sinyal adaptasi tambahan bagi model. Perbandingan ketiga konsep ini disajikan pada Tabel 2.2.

Tabel 2.2 Perbandingan *Zero-Shot, Few-Shot,* dan *Full Supervised Learning*

Aspek Perbandingan	***Zero-Shot Learning***	***Few-Shot Learning***	***Full Supervised Learning***
Data berlabel bahasa/domain target	Tidak ada (0 sampel)	Sangat sedikit (1-100 sampel per kelas)	Banyak (ratusan hingga ribuan per kelas)
Sumber pengetahuan utama	Pre-training pada bahasa/domain sumber	Pre-training + sedikit sinyal adaptasi target	Data berlabel target secara langsung
Ketergantungan pada transfer	Sangat tinggi	Tinggi	Rendah
Risiko *overfitting* pada target	Tidak ada	Rendah	Sedang hingga tinggi
Biaya anotasi data	Tidak ada	Sangat rendah	Tinggi
Kesesuaian untuk *low-resource*	Baik	Sangat baik	Tidak sesuai
Performa pada bahasa berkerabat	Kompetitif	Lebih baik dari *zero-shot*	Optimal jika data cukup

2.2.5 Metrik Evaluasi

Evaluasi model klasifikasi merupakan tahap kritis untuk mengukur kemampuan generalisasi model pada data yang belum pernah dilihat sebelumnya [51]. Dalam penelitian ini, metrik evaluasi dipilih secara khusus berdasarkan karakteristik skenario *few-shot cross-lingual* dengan distribusi

kelas yang tidak seimbang antar bahasa. Pemilihan metrik yang tepat sangat penting karena metrik yang tidak sensitif terhadap ketidakseimbangan kelas dapat menghasilkan gambaran performa yang menyesatkan [51]. Keseluruhan metrik yang digunakan dalam penelitian ini mencakup: *F1-Weighted*, *Confusion Matrix*, Standar Deviasi & *Coefficient of Variation* (CV), dan *95% Confidence Interval* (CI), yang semuanya dihitung dari eksperimen *multi-seed* dengan 5 *random seed* berbeda.

1. *F1-Weighted*

   F1-Weighted adalah varian dari *F1-Score* yang menghitung rata-rata tertimbang nilai *F1-Score* dari setiap kelas, dengan bobot yang sebanding terhadap jumlah sampel nyata (*support*) masing-masing kelas [52]. Berbeda dengan *F1-Macro* yang memberikan bobot yang sama kepada setiap kelas tanpa mempertimbangkan frekuensinya, *F1-Weighted* memberikan kontribusi yang lebih besar kepada kelas dengan jumlah sampel lebih banyak sehingga lebih mencerminkan kinerja model secara keseluruhan pada distribusi data yang tidak seimbang [53]. *F1-Score* per kelas diperoleh melalui rata-rata harmonik antara *precision* (P) dan *recall* (R) dari kelas tersebut. Selanjutnya, bobot (w_i) setiap kelas ditetapkan sebagai proporsi jumlah sampel kelas ke-*i* terhadap total sampel. Terakhir, *F1-Weighted* dihitung sebagai jumlah tertimbang seluruh *F1-Score* per kelas. Ketiga perhitungan tersebut dinyatakan secara urut pada Persamaan 2.1, Persamaan 2.2, dan Persamaan 2.3 [52].

$$w_i = \frac{n_i}{N} \tag{2.1}$$

$$F1_i = \frac{2P_iR_i}{P_i + R_i} \tag{2.2}$$

$$\text{F1-Weighted} = \sum_i (w_i \times F1_i) \tag{2.3}$$

Rumus 2.1. Rumus *F1-Weighted*

   Keterangan:

   a. n_i: jumlah sampel aktual pada kelas ke-*i*.
   b. N: total jumlah seluruh sampel pada set evaluasi.

c. P_i: *precision* kelas ke-*i*, yaitu proporsi prediksi positif kelas ke-*i* yang benar.
d. R_i: *recall* kelas ke-*i*, yaitu proporsi sampel kelas ke-*i* yang berhasil terdeteksi.
e. $F1_i$: nilai *F1-Score* kelas ke-*i* yang merupakan rata-rata harmonik *precision* dan *recall* kelas tersebut.
f. w_i: bobot kelas ke-*i*, dihitung berdasarkan Persamaan 2.1.

2. *Confusion Matrix*

Confusion matrix adalah tabel yang menyusun hasil prediksi model secara sistematis berdasarkan kombinasi antara label aktual dan label prediksi untuk setiap kelas yang ada [51]. Melalui tabel ini, distribusi kesalahan model dapat diamati secara lebih rinci dibandingkan sekadar melihat angka akurasi, karena peneliti dapat mengetahui secara spesifik jenis kesalahan yang paling dominan terjadi. Studi menunjukkan bahwa analisis con*fusion matrix* sangat bermanfaat dalam mendiagnosis kelemahan model pada kondisi tertentu, misalnya ketika model cenderung bias terhadap salah satu kelas akibat distribusi data yang tidak merata [54]. Informasi yang diperoleh dari *confusion matrix* kemudian dapat digunakan sebagai dasar perbaikan model pada iterasi berikutnya.

3. Standar Deviasi *& Coefficient of Variation (CV)*

Standar deviasi (σ) merupakan ukuran statistik yang mengkuantifikasi sebaran nilai-nilai pengamatan terhadap nilai rata-ratanya [55]. Nilai standar deviasi yang kecil menandakan bahwa nilai-nilai pengamatan cenderung berkumpul di sekitar rata-rata, sedangkan nilai yang besar mengindikasikan sebaran yang luas. Dalam konteks evaluasi model, standar deviasi dari beberapa kali eksekusi dengan inisialisasi berbeda mencerminkan tingkat konsistensi atau stabilitas kinerja model tersebut [55]. *Coefficient of Variation* (CV) adalah ukuran variabilitas relatif yang dinyatakan dalam bentuk persentase, diperoleh dari rasio antara standar deviasi dan nilai rata-rata [55]. CV memungkinkan perbandingan tingkat variabilitas antar model atau konfigurasi yang memiliki rentang nilai rata-rata berbeda, karena sifatnya yang tidak berdimensi (*dimensionless*).

Semakin kecil nilai CV, semakin stabil dan dapat direproduksi kinerja yang dihasilkan [55]. Perhitungan standar deviasi dan CV dinyatakan pada Persamaan 2.4 dan Persamaan 2.5.

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2.4}$$

Rumus 2.2. Rumus Standar Deviasi

$$\mathrm{CV} = \frac{\sigma}{\bar{x}} \times 100\% \tag{2.5}$$

Rumus 2.3. Rumus *Coefficient of Variation*

Keterangan:

a. x_i: nilai pengamatan ke-*i* dari total *n* eksekusi.
b. $\bar{x}$: nilai rata-rata dari seluruh *n* pengamatan.
c. *n*: jumlah total eksekusi atau pengulangan eksperimen.
d. σ: standar deviasi sampel.
e. *CV*: koefisien variasi dalam persentase, nilai yang lebih kecil menunjukkan kinerja yang lebih stabil.

4. 95% *Confidence Interval* (CI)

Confidence Interval (CI) adalah suatu rentang nilai yang diestimasi secara statistik untuk merepresentasikan parameter populasi yang tidak diketahui, berdasarkan data sampel yang tersedia [55]. CI dengan tingkat kepercayaan 95% berarti bahwa apabila proses pengambilan sampel dan konstruksi interval diulangi sebanyak 100 kali, diharapkan sekitar 95 interval yang terbentuk akan mengandung nilai parameter populasi yang sesungguhnya [55]. Ketika jumlah pengamatan kecil ($n < 30$), distribusi-t *Student* digunakan sebagai pengganti distribusi normal untuk mengakomodasi ketidakpastian yang lebih besar akibat ukuran sampel yang terbatas [55]. Pelaporan CI memberikan informasi mengenai presisi estimasi rata-rata, sehingga selisih kinerja antar model atau konfigurasi dapat dinilai secara lebih objektif [55]. Perhitungan *95% Confidence Interval* dinyatakan pada Persamaan 2.6.

$$\mathrm{CI}_{95} = \bar{x} \pm t_{\alpha/2,n-1} \cdot \frac{\sigma}{\sqrt{n}} \quad (2.6)$$

Rumus 2.4. Rumus 95% *Confidence Interval*

Keterangan:

a. $\bar{x}$: nilai rata-rata dari seluruh *n* pengamatan.
b. $t(\alpha/2, n-1)$: nilai kritis distribusi t *Student* dengan derajat kebebasan $n - 1$ dan tingkat signifikansi $\alpha = 0{,}05$ (uji dua sisi).
c. σ: standar deviasi sampel.
d. *n*: jumlah total eksekusi atau pengulangan eksperimen.

5. *Trainable Parameters*

   Trainable parameters adalah jumlah parameter dalam model yang nilainya diperbarui selama proses pelatihan melalui mekanisme optimasi [23]. Pada model berbasis *deep learning* yang memiliki arsitektur besar, jumlah total parameter dapat mencapai ratusan juta, sehingga tidak selalu efisien untuk melatih seluruh parameter pada setiap skenario adaptasi. Pendekatan *Parameter-Efficient Fine-Tuning* hadir sebagai alternatif yang hanya mengaktifkan sebagian kecil parameter untuk dilatih, sehingga jumlah trainable parameters dapat ditekan secara drastis tanpa kehilangan kualitas representasi secara signifikan [23]. Pengukuran metrik ini memberikan gambaran konkret mengenai efisiensi suatu pendekatan pelatihan, dan sering digunakan sebagai indikator dalam membandingkan berbagai strategi adaptasi model.

6. *Execution Time*

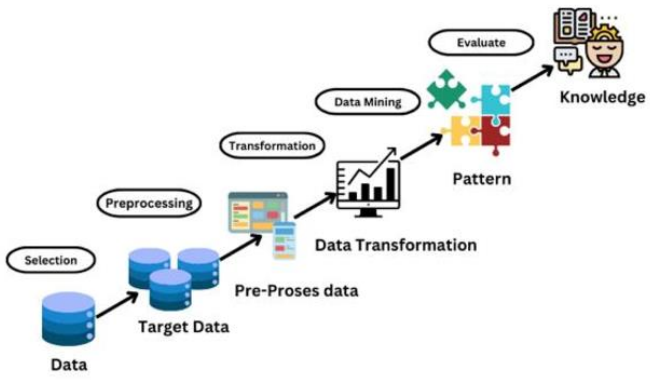
   Execution time mencatat total durasi waktu yang dibutuhkan model untuk menyelesaikan suatu proses, baik pada tahap pelatihan maupun inferensi [20]. Metrik ini relevan terutama ketika efisiensi komputasi menjadi salah satu kriteria penilaian, misalnya dalam skenario di mana model harus dijalankan pada perangkat keras dengan kapasitas terbatas atau dalam sistem yang membutuhkan respons cepat. Penelitian yang membandingkan berbagai pendekatan *fine-tuning* secara konsisten melaporkan perbedaan *execution time* yang cukup signifikan antar metode, yang menunjukkan

bahwa metrik ini perlu diukur dan dilaporkan secara eksplisit agar perbandingan antar pendekatan dapat dilakukan secara adil [18].

2.3 *Framework* dan Algoritma

2.3.1 KDD (*Knowledge Discovery in Databases*)

Knowledge Discovery in Databases (KDD) adalah kerangka kerja sistematis yang digunakan untuk mengekstraksi pengetahuan yang bermakna dari kumpulan data berukuran besar melalui serangkaian tahapan yang terstruktur dan saling berkaitan [56]. Kerangka ini tidak sekadar menjalankan algoritma pada data mentah, melainkan memandu seluruh proses mulai dari pemilihan data hingga interpretasi hasil akhir secara metodologis. KDD pertama kali diperkenalkan oleh Fayyad et al. dan sejak saat itu telah diadopsi secara luas dalam berbagai penelitian berbasis data, termasuk di bidang klasifikasi teks dan *natural language processing* [57]. Sifat KDD yang iteratif memungkinkan peneliti untuk kembali ke tahap sebelumnya apabila hasil pada tahap berikutnya belum memenuhi ekspektasi, sehingga kualitas pengetahuan yang dihasilkan dapat terus diperbaiki secara bertahap [56]. Secara keseluruhan, KDD terdiri dari lima tahapan utama yang ditunjukkan pada Gambar 2.1, yaitu *selection, preprocessing, transformation, data mining,* dan *evaluation*.



Gambar 2.1. Tahapan KDD
Sumber: [58]

1. *Selection*

   Tahap pertama dalam KDD adalah *selection*, yaitu proses pemilihan data yang relevan dari keseluruhan sumber data yang tersedia [57]. Tidak semua

data yang ada perlu digunakan dalam proses analisis, sehingga pada tahap ini peneliti menentukan subset data yang paling sesuai dengan tujuan penelitian berdasarkan kriteria yang telah ditetapkan sebelumnya. Pemilihan data yang tepat pada tahap awal ini sangat berpengaruh terhadap kualitas hasil akhir, karena data yang tidak relevan atau tidak representatif dapat menghasilkan model yang bias dan tidak dapat diandalkan [58].

2. *Preprocessing*

   Setelah data dipilih, tahap berikutnya adalah *preprocessing* yang bertujuan untuk membersihkan dan mempersiapkan data agar layak diproses lebih lanjut [59]. Data yang dikumpulkan dari sumber nyata umumnya mengandung berbagai ketidaksempurnaan seperti nilai yang hilang, duplikasi, *noise*, dan inkonsistensi format yang dapat mengganggu proses analisis. Pada tahap ini, berbagai teknik pembersihan data diterapkan untuk mengatasi permasalahan tersebut sehingga data yang dihasilkan memiliki kualitas yang lebih baik dan konsisten [56]. Kualitas data pada tahap ini secara langsung menentukan keandalan model yang akan dibangun pada tahap selanjutnya.

3. *Transformation*

   Transformation merupakan tahap di mana data yang telah dibersihkan diubah ke dalam format atau representasi yang sesuai dengan kebutuhan algoritma yang akan digunakan [56]. Proses ini dapat mencakup normalisasi nilai, pengkodean variabel kategorikal, reduksi dimensi, maupun pembagian data menjadi subset pelatihan, validasi, dan pengujian. Tujuan utama dari tahap ini adalah memastikan bahwa data berada dalam bentuk yang paling optimal untuk diolah oleh model, sehingga proses pembelajaran dapat berjalan secara efektif dan efisien [57].

4. *Data Mining*

   Tahap *data mining* merupakan inti dari seluruh proses KDD, di mana algoritma atau model diterapkan pada data yang telah dipersiapkan untuk menemukan pola, relasi, atau pengetahuan tersembunyi [60]. Pemilihan algoritma pada tahap ini bergantung pada jenis tugas yang ingin diselesaikan, apakah berupa klasifikasi, klasterisasi, regresi, atau tugas

lainnya. Proses ini melibatkan eksperimen dengan berbagai konfigurasi model dan *hyperparameter* untuk menemukan kombinasi yang menghasilkan performa terbaik pada data yang tersedia [47].

5. *Evaluation*

   Tahap terakhir adalah *evaluation*, yaitu penilaian terhadap pola atau pengetahuan yang dihasilkan oleh model pada tahap sebelumnya [61]. Pada tahap ini, model diuji menggunakan data yang tidak pernah dilihat selama proses pelatihan untuk mengukur kemampuan generalisasinya secara objektif. Hasil evaluasi diinterpretasikan menggunakan berbagai metrik yang relevan dengan jenis tugas yang dikerjakan, dan apabila hasilnya belum memenuhi standar yang ditetapkan, proses dapat diulang kembali dari tahap sebelumnya untuk melakukan perbaikan yang diperlukan.

2.3.2 Multilingual BERT

Multilingual BERT (mBERT) adalah varian dari arsitektur BERT yang dilatih secara bersamaan pada korpus Wikipedia dari 104 bahasa yang berbeda [62]. Model ini menggunakan *shared vocabulary* berukuran 110.000 token dari WordPiece yang memungkinkan *sub-word overlap* antar bahasa, di mana token-token yang muncul di beberapa bahasa sekaligus menjadi jembatan implisit bagi model untuk membangun representasi lintas bahasa meskipun tanpa menggunakan *parallel corpora* atau sinyal *cross-lingual* secara eksplisit selama proses *pre-training* [63]. Berbeda dengan BERT monolingual yang hanya dilatih pada satu bahasa, mBERT dirancang agar satu model tunggal dapat digunakan untuk memproses teks dari berbagai bahasa secara langsung.

Secara arsitektur, mBERT identik dengan BERT-Base yang terdiri dari 12 lapisan *encoder*, 768 dimensi *hidden*, 12 *attention head*, dengan total 177,9 juta parameter [62]. Teks input terlebih dahulu melalui proses tokenisasi menggunakan WordPiece, kemudian ditambahkan *embedding* posisional sebelum diproses oleh lapisan-lapisan *encoder* seperti pada Gambar 2.2. Mekanisme inti yang mendasari kemampuan representasi kontekstual pada

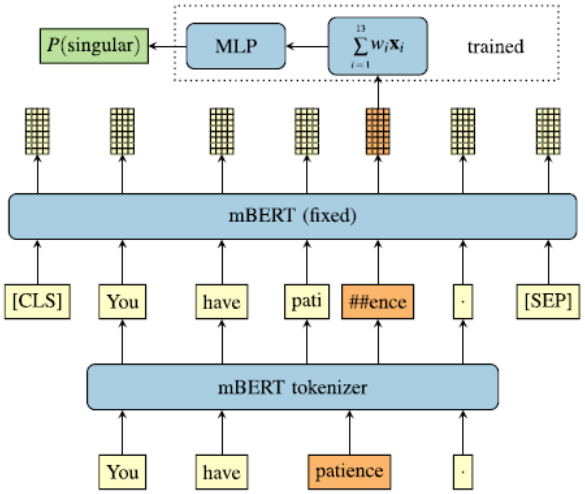
mBERT adalah *scaled dot-product self-attention* yang diformulasikan pada Persamaan 2.7 [63].

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{2.7}$$

Rumus 2.5. Mekanisme *Self-Attention*

Keterangan:

a. *Q, K, V*: Matriks *query*, *key,* dan *value* yang merupakan proyeksi linear dari representasi input.
b. d_k: Dimensi dari vektor *key*, digunakan sebagai faktor skala untuk menstabilkan gradien.
c. *softmax*: Fungsi normalisasi yang mengubah skor perhatian menjadi distribusi probabilitas.



Gambar 2.2. Arsitektur mBERT: Multilingual Pre-training dengan *Shared Vocabulary*
Sumber: [63]

mBERT juga mampu melakukan *cross-lingual transfer* yang efektif, baik dalam skenario *zero-shot* di mana bahasa target sama sekali tidak disertakan dalam pelatihan, maupun dalam skenario *few-shot* di mana sejumlah kecil sampel bahasa target diinjeksikan ke dalam data pelatihan untuk meningkatkan kemampuan adaptasi model [64]. Efektivitas *transfer* ini dipengaruhi oleh kedekatan tipologis antara bahasa sumber dan bahasa target, di mana pasangan bahasa yang berkerabat seperti Bahasa Indonesia dan Bahasa Melayu

menghasilkan *transfer* yang lebih baik dibandingkan pasangan yang tidak berkerabat [64]. Kemampuan ini muncul karena mBERT secara implisit membangun ruang representasi yang bersifat language-agnostic, sehingga kata-kata dengan makna serupa dari bahasa yang berbeda cenderung memiliki vektor representasi yang berdekatan dalam ruang laten yang sama [62].

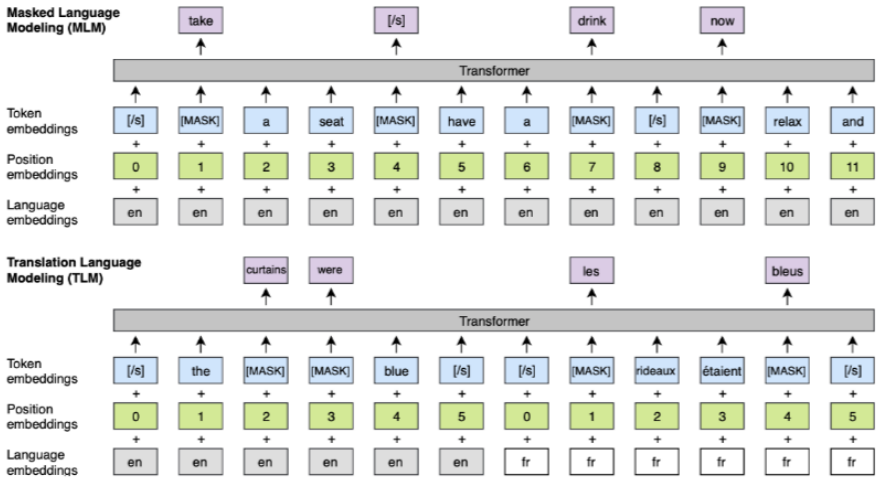
Namun demikian, kemampuan mBERT tidak merata untuk semua bahasa. Penelitian lanjutan menunjukkan bahwa bahasa-bahasa dengan sumber daya rendah (*low-resource languages*) mendapatkan representasi yang kurang optimal dalam mBERT karena porsi data pelatihan yang jauh lebih kecil dibandingkan bahasa dominan seperti Inggris atau Jerman [62]. Distribusi data yang tidak merata ini menjadi salah satu tantangan utama dalam penerapan mBERT untuk tugas deteksi hoaks lintas bahasa, terutama ketika bahasa target memiliki karakteristik morfologis yang berbeda dari bahasa-bahasa dengan sumber daya tinggi. Tahapan utama dimulai dari preprocessing teks, tokenisasi menggunakan WordPiece tokenizer, pembentukan *embedding* (token, *position*, dan *segment embedding*), hingga proses *forward pass* melalui *layer transformer*. Selain itu, ditunjukkan pula bagaimana representasi output (biasanya token CLS) digunakan sebagai input ke *classifier* untuk menghasilkan label prediksi.

2.3.3 XLM-RoBERTa

XLM-RoBERTa (*Cross-lingual Language Model RoBERTa*) adalah model *transformer* multilingual yang dikembangkan untuk mengatasi keterbatasan mBERT dalam menangani bahasa-bahasa dengan sumber daya terbatas [13]. Perbedaan mendasar terletak pada data pelatihan yang digunakan: mBERT hanya dilatih pada *Wikipedia dump* dari 104 bahasa, sedangkan XLM-RoBERTa dilatih pada korpus CommonCrawl CC-100 yang berukuran 2,5 TB dan mencakup 100 bahasa [18]. Volume data yang jauh lebih besar ini membuat bahasa-bahasa *low-resource* memperoleh proporsi data pelatihan yang lebih memadai sehingga representasi yang dihasilkan lebih kuat.

Dari sisi optimasi pelatihan, XLM-RoBERTa mengadopsi tiga perbaikan yang diperkenalkan oleh RoBERTa [19] Perbaikan pertama adalah penghapusan tugas *Next Sentence Prediction* (NSP) yang terbukti tidak memberikan kontribusi positif. Perbaikan kedua adalah penggunaan *dynamic masking* yang menghasilkan pola *mask* berbeda setiap *epoch* sehingga model tidak menghafal pola tertentu. Perbaikan ketiga adalah pelatihan dengan *batch size* yang lebih besar dan *training steps* yang lebih banyak untuk konvergensi yang lebih optimal.

XLM-RoBERTa menggunakan SentencePiece *tokenizer* dengan *vocabulary* berukuran 250.000 token, jauh lebih besar dibandingkan 110.000 token pada WordPiece mBERT [13]. Ukuran *vocabulary* yang lebih besar ini mengurangi dominasi bahasa tertentu dan memberikan cakupan yang lebih merata untuk bahasa-bahasa *low-resource*. XLM-RoBERTa Base memiliki 12 lapisan *encoder*, 768 dimensi *hidden*, 12 *attention head*, dan total 278 juta parameter (Gambar 2.3).



Gambar 2.3. Arsitektur XLM-RoBERTa: *Pre-training* Multibahasa dengan SentencePiece Tokenizer dan CommonCrawl Corpus
Sumber: [13]

Proses *pre-training* XLM-RoBERTa secara eksklusif menggunakan tujuan *Masked Language Modeling* (MLM), namun dengan penerapan *dynamic masking* yang menghasilkan pola masking berbeda pada setiap iterasi pelatihan. Secara formal, *dynamic masking* berarti bahwa untuk setiap sampel teks yang diberikan pada epoch ke-*e*, himpunan indeks token yang

disembunyikan $M^{(e)}$ diregenerasi secara acak, sehingga model terpapar pada variasi konteks yang lebih kaya dibandingkan *static masking* [52]. Fungsi *loss* yang dioptimalkan tetap mengikuti formulasi MLM standar sebagaimana dinyatakan pada Persamaan 2.8 [13].

$$\mathcal{L}_{XLM-R} = -\sum_{e=1}^{E} \sum_{i \in M^{(e)}} \log P\left(x_i \mid x_{\setminus i}^{(e)} \right) \tag{2.8}$$

Rumus 2.6. *Masked Language Modeling* dengan *Dynamic Masking* pada XLM-RoBERTa

Keterangan:

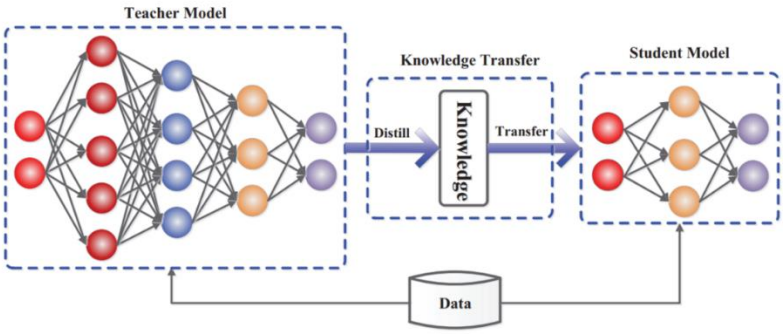
a. $\mathcal{L}_{XLM-R}$: Total fungsi *loss pre-training* XLM-RoBERTa yang diakumulasi lintas *epoch*.
b. E: Jumlah total *epoch* pelatihan.
c. $M^{(e)}$: Himpunan indeks token yang disembunyikan pada *epoch* ke-e, diregenerasi secara acak setiap *epoch* (*dynamic masking*).
d. x_i: Token ke-i yang menjadi target prediksi. e. $x\setminus_i^{(e)}$: Seluruh token lain dalam kalimat pada *epoch* ke-e yang digunakan sebagai konteks prediksi.

Keunggulan utama XLM-RoBERTa dibandingkan mBERT terletak pada kemampuannya menangani bahasa *low-resource* secara lebih optimal. Penelitian Conneau et al. menunjukkan bahwa XLM-RoBERTa mampu melampaui performa model *fine-tuned* secara monolingual pada beberapa tugas klasifikasi *cross-lingual*, terutama ketika data berlabel pada bahasa target sangat terbatas [13]. Fenomena ini dijelaskan oleh *curse of multilinguality* yang terjadi pada mBERT, di mana penambahan bahasa dalam satu model dengan kapasitas parameter tetap mengakibatkan pengurangan kualitas representasi per bahasa [13]. XLM-RoBERTa mengatasi masalah ini dengan menambah kapasitas model secara signifikan, sehingga setiap bahasa mendapat alokasi representasi yang lebih memadai.

2.3.4 DistilmBERT

DistilmBERT (DistilmBERT) merupakan versi yang lebih ringkas dari mBERT, dihasilkan melalui proses *knowledge distillation* di mana mBERT bertindak sebagai model *teacher* yang mentransfer pengetahuan ke model *student* yang berukuran lebih kecil [65]. Secara arsitektur, DistilmBERT

memiliki 6 lapisan *Transformer encoder* (setengah dari mBERT), 768 dimensi *hidden*, 12 *attention head*, dengan total 134 juta parameter dibandingkan 177,9 juta parameter pada mBERT [14]. Pengurangan jumlah lapisan ini menghasilkan kecepatan inferensi sekitar dua kali lebih cepat dari mBERT, sambil tetap mempertahankan sekitar 92% performa pada *benchmark* XNLI [65]. Proses distilasi pada DistilmBERT menggunakan kombinasi tiga fungsi *loss* diilustrasikan pada Gambar 2.4.



Gambar 2.4. Pengurangan Lapisan melalui *Knowledge Distillation*
Sumber: [63]

Proses *knowledge distillation* pada DistilmBERT menggunakan kombinasi tiga fungsi *loss*, yaitu *masked language modeling loss* (MLM) yang identik dengan *pre-training* BERT standar, *distillation loss* yang mengukur perbedaan antara distribusi probabilitas teacher dan student, serta *cosine-distance loss* yang mendorong agar vektor representasi *hidden state* antara *student* dan *teacher* memiliki arah yang serupa [65]. Kombinasi ketiga fungsi ini dinyatakan pada Persamaan 2.9 [65].

$$\mathcal{L}_{\text{total}} = \alpha \cdot \mathcal{L}_{CE} + \beta \cdot \mathcal{L}_{\text{MLM}} + \gamma \cdot \mathcal{L}_{\cos} \quad (2.9)$$

Rumus 2.7. *Triple Loss* untuk *Knowledge Distillation*

Keterangan:

a. $\mathcal{L}_{\text{total}}$: Total fungsi kerugian yang digunakan dalam pelatihan DistilmBERT.
b. $\mathcal{L}_{CE}$: *Cross-entropy loss* (*distillation loss*) antara distribusi probabilitas teacher dan student yang diskalakan dengan suhu T.
c. $\mathcal{L}_{\text{MLM}}$: *Masked Language Modeling loss* standar menggunakan label asli.
d. $\mathcal{L}_{\cos}$: *Cosine-distance loss* antara vektor hidden state student dan teacher.

e. α, β, γ: Bobot yang mengatur kontribusi relatif masing-masing komponen *loss*.

Pada fungsi *distillation loss*, digunakan parameter suhu (*temperature*, *T*) yang melembutkan distribusi probabilitas *output* model *teacher* sehingga model *student* dapat belajar dari pola distribusi yang lebih informatif. Formulasi *distillation loss* ini dinyatakan pada Persamaan 2.10 [65].

$$\mathcal{L}_{CE} = -\sum t^i \log\left(s^i\right) \tag{2.10}$$

Rumus 2.8. *Distillation Loss* dengan *Temperature Scaling*

Keterangan:

a. t^i: Probabilitas kelas ke-i dari *teacher model* setelah diskalakan dengan suhu T (*soft target*).
b. s^i: Probabilitas kelas ke-i dari *student model* setelah diskalakan dengan suhu T.
c. T: Parameter suhu yang mengontrol seberapa lunak distribusi probabilitas yang dihasilkan.

Tahapan *knowledge distillation* untuk DistilmBERT yang dimulai dari input teks, proses tokenisasi, hingga pembentukan *embedding* yang kemudian diproses melalui *layer transformer* yang telah dikompresi. Selain itu, ditunjukkan pula bagaimana *loss function* pada DistilmBERT menggabungkan *loss* terhadap label asli dan *loss* terhadap prediksi *teacher model* (*distillation loss*). Representasi ini membantu memperjelas perbedaan utama antara proses *fine-tuning* pada mBERT dan DistilmBERT, khususnya dalam hal efisiensi parameter dan strategi pembelajaran model.

Dalam penerapan untuk tugas *few-shot cross-lingual classification*, DistilmBERT mengikuti prosedur yang sama seperti mBERT, yaitu dilatih pada data Bahasa Indonesia dan Inggris dengan penyertaan 100 sampel per kelas Bahasa Melayu sebagai *few-shot injection*, kemudian dievaluasi pada data Bahasa Melayu yang tidak diikutsertakan dalam pelatihan. Representasi token CLS dari lapisan terakhir *encoder* digunakan sebagai input bagi lapisan klasifikasi linier, dan *loss function* mengikuti persamaan *cross-entropy loss*

pada Persamaan 2.10. Keunggulan utama DistilmBERT terletak pada efisiensi komputasinya yang sekitar dua kali lebih tinggi dari mBERT, sehingga lebih praktis untuk skenario *real-time* maupun perangkat dengan sumber daya terbatas tanpa mengorbankan kualitas transfer lintas bahasa secara signifikan [65].

2.3.5 *Adaptive-Rank* LoRA *(ARL-LoRA)*

Adaptive-Rank LoRA (ARL-LoRA) merupakan pengembangan dari LoRA standar yang dirancang untuk mengatasi keterbatasan mendasar dalam penerapan *rank* yang seragam di seluruh lapisan model. Pada implementasi LoRA konvensional, nilai rank *r* ditetapkan sama untuk seluruh matriks bobot yang dikenai adaptasi, tanpa mempertimbangkan perbedaan peran fungsional setiap lapisan dalam arsitektur *transformer* terhadap performa *task* [40]. Penelitian terkait AdaLoRA mengonfirmasi bahwa pendistribusian *rank* secara adaptif menghasilkan performa yang lebih baik secara konsisten, terutama pada kondisi budget parameter terbatas, karena distribusi seragam menghasilkan adaptasi yang tidak optimal ketika setiap lapisan memiliki kontribusi yang berbeda [40]. Temuan ini memberikan landasan empiris yang kuat bagi pendekatan alokasi rank berbasis kelompok lapisan yang diterapkan dalam penelitian ini.

Studi pada model *transformer* menunjukkan bahwa lapisan bawah (*bottom layers*) cenderung menyimpan representasi linguistik universal yang relatif stabil lintas tugas, sementara lapisan atas (*top layers*) bertanggung jawab atas adaptasi yang bersifat *task-specific* [25]. Dalam konteks *cross-lingual transfer*, perbedaan peran ini menjadi semakin kritis karena modifikasi berlebihan pada lapisan bawah berisiko mengganggu kemampuan *transfer* lintas bahasa yang sudah terbentuk selama *pre-training* multilingual [11]. Sementara itu, lapisan atas justru perlu diadaptasi secara lebih agresif agar model mampu mengenali pola *task-specific*, seperti ciri-ciri linguistik hoaks yang bervariasi antar bahasa [21]. Dilema inilah yang mendorong perlunya mekanisme alokasi *rank* yang berbeda-beda per kelompok lapisan.

ARL-LoRA mengatasi permasalahan tersebut dengan mengalokasikan rank yang berbeda berdasarkan pengelompokan lapisan. *Rank* kecil pada lapisan bawah untuk mempertahankan representasi universal, rank sedang pada lapisan tengah sebagai zona *cross-lingual transfer*, dan rank besar pada lapisan atas untuk adaptasi *task-specific* yang lebih kuat [40][25]. Berbeda dari AdaLoRA yang menggunakan mekanisme SVD dinamis selama proses pelatihan, ARL-LoRA menentukan rank secara statis berdasarkan pengetahuan *a priori* mengenai peran fungsional lapisan, sehingga implementasinya lebih sederhana tanpa mengorbankan efektivitas adaptasi [40]. Formulasi matematis untuk *forward pass* ARL-LoRA pada lapisan ke-l disajikan pada Persamaan 2.11 berikut.

$$h_l = W_0 x + \left(\alpha_g(l)/r_g(l)\right) \cdot B_l \cdot A_l \cdot x \tag{2.11}$$

Rumus 2.9. *Forward Pass* ARL-LoRA per Lapisan

Keterangan:

a. h_l: Vektor output dari lapisan ke-l yang menerapkan ARL-LoRA.
b. W_0: Matriks bobot *pre-trained* yang dibekukan.
c. x: Vektor input yang diterima lapisan tersebut.
d. $g(l)$: Fungsi yang memetakan indeks lapisan l ke kelompok (*bottom, mid,* atau *top*).
e. $r_g(l)$: *Rank* LoRA yang ditentukan berdasarkan kelompok lapisan $g(l)$.
f. $\alpha_g(l)$: Konstanta skala yang disesuaikan untuk kelompok lapisan $g(l)$, biasanya $\alpha = 2 \times r$.
g. $B_l \in \mathbb{R}^{d \times r_g(l)}, A_l \in \mathbb{R}^{r_g(l) \times k}$: Pasangan matriks adaptasi berperingkat rendah untuk lapisan l.

Persamaan 2.9 menunjukkan bahwa *output* lapisan ke-l merupakan penjumlahan antara transformasi bobot asli W_0x dengan koreksi adaptif yang dihasilkan oleh pasangan matriks B_l dan A_l. Faktor skala $\alpha_g(l)/r_g(l)$ berperan mengatur besarnya kontribusi adaptasi relatif terhadap *output* asli, dan nilainya bervariasi antar kelompok lapisan. Matriks A_l diinisialisasi dengan distribusi Gaussian acak, sedangkan B_l diinisialisasi dengan nol, sehingga kontribusi adaptasi pada awal pelatihan bernilai nol dan model

belajar secara bertahap [40]. Pendekatan inisialisasi ini memastikan bahwa model mulai pelatihan dari titik yang identik dengan bobot *pre-trained*, menghindari gangguan representasi yang sudah terbentuk.

Konfigurasi *rank* per kelompok lapisan untuk model *transformer* 12 lapisan disajikan secara lengkap pada Tabel 2.3. Tabel ini merinci indeks lapisan, nilai rank, konstanta alpha, serta justifikasi pemilihan rank untuk masing-masing kelompok. Melalui konfigurasi ini, total parameter yang dilatih oleh ARL-LoRA tetap jauh lebih kecil dibandingkan *full fine-tuning*, namun distribusinya lebih sesuai dengan kebutuhan adaptasi per lapisan [25][40].

Tabel 2.3 Konfigurasi ARL-LoRA per Kelompok Lapisan (12-*layer Transformer*)

Kelompok	**Indeks Lapisan**	***Rank* (r)**	***Alpha* (α)**	**Justifikasi**
Bottom	0–3	8	16	Representasi linguistik universal; modifikasi minimal untuk menjaga kemampuan cross-lingual.
Mid	4–7	16	32	Zona *cross-lingual transfer*; *rank* sedang untuk adaptasi moderat tanpa mengganggu representasi bersama.
Top	8–11	64	128	Adaptasi task-specific; *rank* tinggi untuk menangkap pola kompleks.

**Untuk model DistilmBERT (6 lapisan): Bottom=0–1, Mid=2–3, Top=4–5.*

Tabel 2.3 memperlihatkan perbedaan signifikan antara rank lapisan bawah (r=8) dan lapisan atas (r=64), yakni rasio delapan kali lipat. Perbedaan ini mencerminkan asumsi bahwa adaptasi *task-specific* pada lapisan atas membutuhkan kapasitas representasi yang jauh lebih besar dibandingkan penyesuaian pada lapisan bawah. Studi yang dilakukan oleh Li et al. mendukung pendekatan ini dengan menunjukkan bahwa alokasi budget parameter yang lebih besar ke lapisan atas secara konsisten menghasilkan peningkatan performa yang lebih nyata pada tugas klasifikasi teks [25]. Nilai alpha yang dua kali lipat dari *rank* pada setiap kelompok merupakan praktik umum dalam implementasi LoRA untuk menjaga stabilitas gradien selama pelatihan [40].

2.3.6 *Bias Tuning* (BitFit)

BitFit (*Bias-terms Fine-Tuning*) adalah metode *fine-tuning* yang hanya memodifikasi *bias terms* pada model selama proses adaptasi, sementara seluruh matriks bobot utama tetap dibekukan [26]. Berbeda dari LoRA yang menambahkan parameter baru berupa matriks berperingkat rendah, BitFit tidak memperluas arsitektur model melainkan hanya mengaktifkan kembali subset parameter yang sudah ada. Jumlah parameter yang dilatih oleh BitFit sangat kecil, yakni kurang dari 0,1% dari total parameter model, sehingga metode ini termasuk dalam kategori *sparse fine-tuning* [26].

Dasar teoretis dari BitFit berkaitan dengan hipotesis bahwa proses *fine-tuning* pada model *pre-trained* lebih banyak berfungsi untuk mengekspos pengetahuan yang sudah dipelajari selama *pre-training*, bukan untuk mempelajari pola linguistik yang sepenuhnya baru [26]. *Bias terms* berperan sebagai pengatur titik aktivasi pada setiap neuron, sehingga modifikasi selektif pada parameter ini dapat mengubah perilaku keputusan model tanpa perlu memodifikasi matriks bobot yang menyimpan pengetahuan utama. Formulasi *forward pass* dengan BitFit dinyatakan pada Persamaan 2.12.

$$h = W_0 x + b^* \tag{2.12}$$

Rumus 2.10. *Forward Pass* dengan BitFit

Keterangan:

a. h: Vektor *output* dari lapisan linier.
b. W_0: Matriks bobot *pre-trained* yang dibekukan (tidak menerima gradien).
c. x: Vektor *input*.
d. b*: Vektor bias yang menjadi satu-satunya parameter yang diperbarui selama *fine-tuning*.

Persamaan 2.12 terlihat sederhana, namun implikasinya cukup dalam. Parameter b* yang dioptimalkan selama pelatihan berfungsi menggeser distribusi output setiap lapisan secara konstan, memungkinkan model menyesuaikan sensitivitasnya terhadap fitur yang relevan dengan tugas target tanpa memodifikasi bobot inti representasi linguistik [26]. Kemampuan ini menjadi semakin relevan pada skenario *cross-lingual*, karena perbedaan distribusi fitur antara bahasa sumber dan bahasa target dapat dikompensasi sebagian melalui pergeseran bias ini [9].

Komplementaritas antara BitFit dan LoRA bersumber dari perbedaan fundamental dalam jenis transformasi yang dimodelkan oleh masing-masing metode. LoRA memodelkan perubahan bobot sebagai matriks berperingkat rendah $\Delta W = BA$, yang secara geometris merepresentasikan *rotasi* dan *scaling* dalam ruang representasi [22]. BitFit memodelkan *translasi* atau pergeseran konstan dalam ruang yang sama [26]. Pada skenario *cross-lingual few-shot*, kedua jenis transformasi ini dibutuhkan secara bersamaan: LoRA mengadaptasi pola representasi untuk mengenali ciri-ciri hoaks, sementara BitFit menggeser sensitivitas keputusan agar sesuai dengan distribusi fitur bahasa target yang berbeda dari bahasa sumber [9]. Proses penerapan BitFit pada model *transformer* sangat ringkas karena tidak memerlukan modifikasi arsitektur, melainkan hanya pengaturan atribut requires_grad pada parameter bias yang ada. Kesederhanaan implementasi ini menjadikan BitFit sangat mudah dikombinasikan dengan metode PEFT lain seperti LoRA tanpa konflik yang berarti [26].

2.3.7 *Selective Embedding Adaptation* (SEA)

Selective Embedding Adaptation (SEA) adalah teknik adaptasi yang bekerja pada lapisan *token embedding* secara selektif, di mana hanya vektor *embedding* untuk token yang benar-benar muncul dalam data bahasa target yang diperbarui, sementara seluruh token lain tetap dibekukan [29]. Pada model multilingual *pre-trained*, *vocabulary* berisi ratusan ribu token dari berbagai bahasa dan hanya sebagian kecil yang relevan untuk satu bahasa tertentu. Memperbarui seluruh *embedding* secara bersamaan berisiko merusak representasi token bahasa lain yang sudah terbentuk dengan baik selama *pre-training*, sehingga pendekatan selektif menjadi lebih efisien dan aman [29].

Pada konteks bahasa *low-resource*, representasi vektor *embedding* untuk token yang jarang muncul dalam data *pre-training* cenderung kurang optimal karena frekuensi kemunculan yang rendah selama proses pelatihan awal [10]. SEA mengatasi permasalahan ini melalui mekanisme *gradient masking* yang memastikan bahwa hanya token bahasa target yang menerima pembaruan

gradient selama *fine-tuning*. Formulasi *gradient masking* pada SEA dinyatakan pada Persamaan 2.13.

$$\nabla E'(i) = M(i) \cdot \nabla E(i) \quad (2.13)$$

Rumus 2.11. *Gradient Masking* pada SEA

Keterangan:

a. $M(i)$: Nilai mask untuk token ke-i, bernilai 1 jika token ke-i termasuk dalam himpunan token target, dan 0 jika tidak.
b. $\nabla E(i)$: Nilai *gradient* asli pada baris *embedding* token ke-i sebelum dimodifikasi.
c. $\nabla E'(i)$: Nilai *gradient* yang telah dimodifikasi (*masked gradient*) pada baris *embedding* token ke-i setelah operasi masking diterapkan.

Persamaan 2.13 menunjukkan bahwa SEA bekerja di tingkat gradien, bukan di tingkat parameter secara langsung. Dengan menetapkan M(i) = 0 untuk token yang tidak relevan, gradien yang masuk ke baris *embedding* tersebut dinolkan sehingga nilai *embedding* tidak berubah meski model terus dilatih. Hanya baris *embedding* dengan $M(i) = 1$ yang memperoleh pembaruan, sehingga model secara efektif hanya belajar meningkatkan representasi token yang benar-benar diperlukan [29]. Pembangunan *mask* M dilakukan pada tahap *pre-processing*, sebagaimana dinyatakan pada Persamaan 2.14.

$$M(i) = \{1 \text{ jika } i \in T_{ms}, 0 \text{ jika } i \notin T_{ms}\} \quad (2.14)$$

Rumus 2.12. Konstruksi *Mask* SEA

Keterangan:

a. $M(i)$: Nilai *mask* untuk token ke-i dalam *vocabulary*.
b. T_{ms}: Himpunan token_id unik yang teridentifikasi muncul dalam data (*few-shot train* + validasi).

Persamaan 2.14 mendefinisikan T_ms sebagai himpunan semua token_id unik yang diperoleh dari proses tokenisasi data. Himpunan ini bersifat statis, artinya dibangun satu kali sebelum pelatihan dimulai dan tidak berubah sepanjang proses pelatihan berlangsung. Proses tokenisasi menggunakan *tokenizer* yang sama persis dengan yang digunakan saat *pre-training* model

untuk memastikan konsistensi representasi token_id [13]. Pendekatan sederhana ini memungkinkan implementasi SEA tanpa modifikasi apapun pada kode pelatihan utama, cukup dengan menambahkan satu *gradient hook* sebelum *loop* pelatihan dimulai [29].

Keunggulan utama SEA dibandingkan *full embedding tuning* terletak pada efisiensi parameter yang sangat signifikan. Pada percobaan dengan data Melayu yang terdiri dari 100 sampel *few-shot*, jumlah token unik yang teridentifikasi umumnya berkisar antara 3.000 hingga 6.000 dari total 250.000 token dalam *vocabulary* XLM-RoBERTa [13]. Ini berarti SEA hanya memperbarui sekitar 1–2% dari seluruh parameter embedding, sementara 98% sisanya mempertahankan representasi yang sudah teroptimasi selama *pre-training*. Pendekatan selektif ini juga mengurangi risiko *overfitting* yang sering terjadi ketika seluruh *embedding* dilatih ulang dengan data yang sangat terbatas [49].

2.3.8 Integrasi ARL-LoRA⁺: Kerangka *Hybrid* PEFT

ARL-LoRA⁺ merupakan kerangka *hybrid Parameter-Efficient Fine-Tuning* yang mengintegrasikan tiga mekanisme komplementer yaitu ARL-LoRA, BitFit, dan SEA dalam satu *pipeline* pelatihan terpadu. Integrasi ini didasarkan pada fakta bahwa ketiga komponen beroperasi pada level arsitektur yang berbeda dan tidak saling tumpang tindih: ARL-LoRA memodifikasi *attention weights* dan *feed-forward weights* melalui matriks berperingkat rendah, BitFit memodifikasi *bias terms* pada seluruh lapisan, dan SEA memodifikasi *token embeddings* secara selektif pada lapisan input. Dengan beroperasi pada tiga level yang berbeda, ketiga komponen dapat bekerja secara bersamaan tanpa menimbulkan konflik parameter.

Dari perspektif geometris ruang representasi, ARL-LoRA memodelkan rotasi dan *scaling* melalui matriks berperingkat rendah $\Delta W = BA$ [22], BitFit memodelkan translasi melalui pergeseran *bias* b* [26], dan SEA memodelkan *re-embedding* titik awal melalui koreksi vektor token [29]. Kombinasi ketiga jenis transformasi ini secara teoritis memungkinkan model menjangkau wilayah ruang representasi yang lebih luas dibandingkan penggunaan salah

satu komponen secara tunggal. Dengan masing-masing komponen mengintervensi aspek yang berbeda, potensi sinergi antar ketiganya menjadi lebih besar.

Urutan penerapan ketiga komponen dalam *pipeline* dimulai dari injeksi ARL-LoRA yang membekukan bobot *pre-trained* dan menyisipkan matriks adaptasi berperingkat rendah, kemudian BitFit diaktifkan dengan mengubah atribut *requires_grad* pada seluruh parameter *bias*, dan terakhir SEA mengidentifikasi token bahasa target untuk adaptasi selektif pada lapisan *embedding*. Formulasi *forward pass* lengkap ARL-LoRA$^+$ ada pada Persamaan 2.15.

$$h_l = W_0 \cdot E_{\text{SEA}}(x) + \left(\alpha_g(l)/r_g(l)\right) \cdot B_l \cdot A_l \cdot E_{\text{SEA}}(x) + b_l^* \qquad (2.15)$$

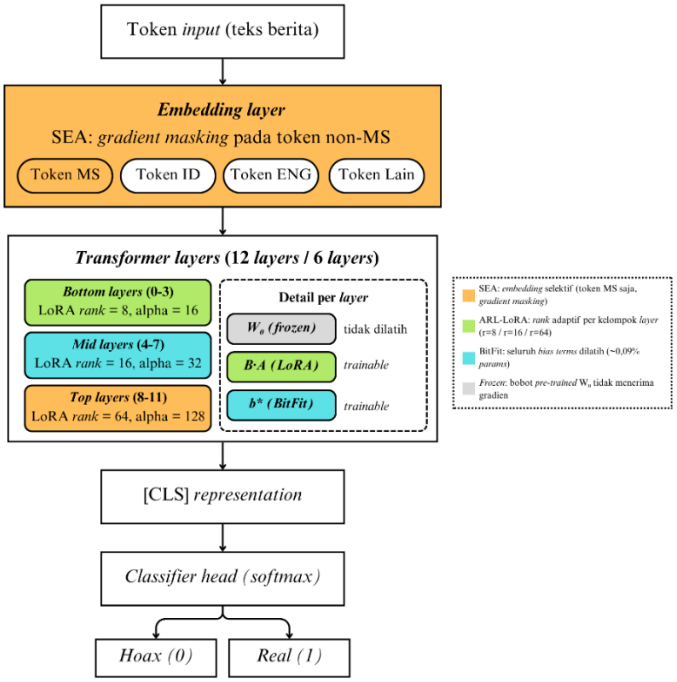
Rumus 2.13. *Forward Pass* ARL-LoRA$^+$ (Lengkap)

Keterangan:

a. h_l: Vektor *output* dari lapisan ke-*l*.
b. W_0: Matriks bobot *pre-trained* yang dibekukan.
c. $E_{\text{SEA}}(x)$: Representasi input *x* setelah melalui *embedding layer* yang diadaptasi secara selektif oleh SEA.
d. B_l , A_l: Matriks adaptasi ARL-LoRA untuk lapisan l dengan *rank* sesuai kelompok g(l).
e. $\alpha_g(l)$, $r_g(l)$: Alpha dan *rank* yang ditentukan berdasarkan kelompok lapisan.
f. b_l^*: Bias *term* lapisan *l* yang diperbarui oleh BitFit.

Persamaan 2.15 merupakan perluasan dari Persamaan 2.11 dengan dua penambahan penting, yaitu penggantian *x* menjadi $E_{\text{SEA}}(x)$ sebagai representasi input yang telah melalui adaptasi *embedding* oleh SEA dan penambahan komponen b_l^* dari BitFit. Ketiga kontribusi adaptasi, yakni dari SEA, ARL-LoRA, dan BitFit, terakumulasi secara aditif dalam satu ekspresi, mencerminkan arsitektur integrasi yang tidak mengganggu satu sama lain. W_0 tetap dibekukan sepanjang pelatihan dan satu-satunya parameter yang berubah adalah B_l , A_l , b_l^*, dan baris *embedding* yang dipilih oleh SEA [26][29].

Gambar 2.5 menyajikan arsitektur keseluruhan ARL-LoRA⁺, menggambarkan aliran data mulai dari token input melalui lapisan embedding (dengan SEA), melewati tiga kelompok lapisan *transformer* dengan rank LoRA dan bias trainable yang berbeda, hingga mencapai *classifier head*. Diagram ini mempertegas bagaimana ketiga komponen beroperasi pada level yang berbeda namun secara bersama-sama membentuk satu kerangka adaptasi yang koheren dan efisien dalam menghadapi skenario *few-shot cross-lingual* [26][40].



Gambar 2.5. Arsitektur keseluruhan ARL-LoRA⁺

Tabel 2.4 menyajikan perbandingan komprehensif antara ketiga komponen ARL-LoRA⁺ dari berbagai aspek teknis. Perbandingan ini penting untuk memahami kontribusi unik masing-masing komponen serta posisinya dalam kerangka integrasi. Melalui tabel ini, terlihat bahwa ARL-LoRA, BitFit, dan SEA menempati relung fungsional yang berbeda dan tidak redundan, sehingga kombinasi ketiganya memberikan cakupan adaptasi yang lebih menyeluruh dibandingkan penggunaan salah satu komponen secara terpisah [26][40].

Tabel 2.4 Perbandingan Komponen ARL-LoRA⁺

Aspek	ARL-LoRA	BitFit	SEA
Level operasi	*Transformer* layers (Q, K, V, FFN)	Bias terms seluruh lapisan	Token *embedding* layer
Jenis transformasi	Rotasi & *scaling* (matriks)	Translasi (*offset*)	Reposisi *embedding* (vektor)
Parameter tambahan	Ya (matriks A, B)	Tidak (bias sudah ada)	Tidak (*embedding* sudah ada)
% parameter dilatih	~1–2% (tergantung *rank*)	~0,09%	~0,5–1% (tergantung token target)
Tujuan utama	Adaptasi pola *task-specific* per lapisan	Pergeseran sensitivitas keputusan	Perbaikan representasi token bahasa target

Berdasarkan Tabel 2.4, total estimasi parameter yang dilatih oleh ARL-LoRA^{+} berkisar antara 1,5–3% dari total parameter model, bergantung pada konfigurasi rank dan jumlah token bahasa target yang teridentifikasi. Angka ini jauh lebih kecil dibandingkan *full fine-tuning* yang melatih 100% parameter, namun secara substansial lebih besar dari pendekatan PEFT tunggal seperti BitFit saja (0,09%) atau LoRA standar rank-8 saja. *Tradeoff* ini disengaja karena ARL-LoRA^{+} dirancang untuk mencapai performa yang mendekati *full fine-tuning* dengan efisiensi parameter yang jauh lebih baik, khususnya pada kondisi data berlabel yang sangat terbatas dalam skenario *few-shot cross-lingual* [21][26].

2.3.9 *Local Interpretable Model-Agnostic Explanations* (LIME)

Explainable Artificial Intelligence (XAI) merupakan bidang riset yang bertujuan membuat proses pengambilan keputusan model *machine learning* dapat dipahami oleh manusia [66]. Kebutuhan ini muncul karena model *deep learning* modern, termasuk *transformer*, bersifat *black-box* di mana mekanisme internal yang menghasilkan prediksi tidak dapat diamati secara langsung oleh pengguna [67]. Metode XAI secara umum terbagi menjadi dua kategori, yaitu *model-specific* yang dirancang khusus untuk arsitektur tertentu dan *model-agnostic* yang dapat diterapkan pada model apapun tanpa memerlukan modifikasi pada struktur internal model tersebut [68].

LIME (*Local Interpretable Model-Agnostic Explanations*) termasuk dalam kategori *model-agnostic* yang bekerja berdasarkan prinsip bahwa

meskipun perilaku global sebuah model *black-box* sangat kompleks, perilaku model di sekitar satu instans prediksi tertentu dapat didekati dengan baik oleh model yang jauh lebih sederhana [30]. Tiga sifat inti LIME tercermin dari namanya: *Local* berarti penjelasan yang dihasilkan hanya berlaku di lingkungan lokal suatu instans tertentu dan bukan secara global; *Interpretable* berarti penjelasan disajikan dalam format yang dapat dipahami manusia seperti bobot kontribusi per kata; dan *Model-Agnostic* berarti LIME dapat digunakan pada model apapun selama model tersebut menghasilkan output probabilitas [30].

Mekanisme kerja LIME untuk klasifikasi teks dimulai dengan pemilihan satu instans teks yang ingin dijelaskan, kemudian LIME menghasilkan sejumlah sampel teks terperturbasi dengan cara menghapus subset kata secara acak dari teks asli. Setiap sampel terperturbasi diproses oleh model asli untuk memperoleh probabilitas prediksi, lalu LIME melatih model linier sederhana pada data terperturbasi tersebut dengan memberikan bobot lebih besar pada sampel yang lebih mirip dengan instans asli. Koefisien model linier tersebut kemudian digunakan sebagai penjelasan mengenai kata mana yang paling berkontribusi terhadap keputusan klasifikasi model [30]. Formulasi optimasi LIME dijelaskan pada Persamaan 2.16 [30].

$$\xi(x) = arg\ \frac{min}{g \in G}\ L(f, g, \pi_x) + \Omega(g) \tag{2.16}$$

Rumus 2.14. Formulasi Optimasi LIME

Keterangan:

a. $\xi(x)$: Penjelasan (*explanation*) yang dihasilkan LIME untuk instans x.
b. G: Himpunan semua model yang dapat diinterpretasikan, misalnya model linier atau pohon keputusan.
c. g: Model pengganti yang dipilih dari himpunan Gsebagai representasi lokal perilaku model.
d. $\mathcal{L}(f, g, \pi_x)$: Fungsi *loss* yang mengukur seberapa tidak setia $g$dalam mendekati $f$pada lingkungan lokal $\pi_x$di sekitar x.
e. π_x: Fungsi *proximity measure* atau kernel yang mendefinisikan seberapa dekat suatu sampel perturbasi dengan instans x.

f. $\Omega(g)$: Ukuran kompleksitas model pengganti g, digunakan untuk menjaga agar penjelasan tetap sesederhana mungkin.

Fungsi kernel yang digunakan untuk menghitung bobot setiap sampel perturbasi berdasarkan jaraknya dari instans asli didefinisikan pada Persamaan 2.17 [30].

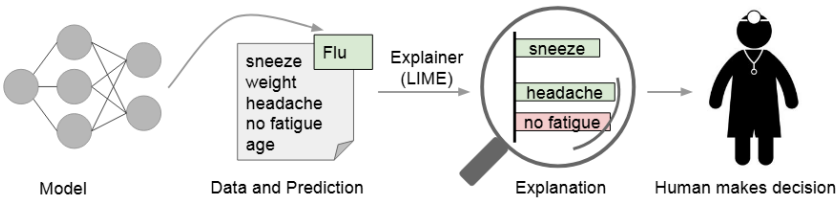
$$\pi_x(z) = exp(-\frac{D(x,z)^2}{\sigma^2}) \quad (2.17)$$

Rumus 2.15. Fungsi Kernel Eksponensial pada LIME

Keterangan:

a. $\pi_x(z)$: Bobot yang diberikan kepada sampel perturbasi z berdasarkan kedekatannya dengan instans asli x.
b. $D(x,z)$: Fungsi jarak antara instans asli $x$dan sampel perturbasi z, biasanya jarak kosinus atau Euclidean.
c. σ: Lebar kernel yang mengontrol seberapa cepat bobot menurun terhadap pertambahan jarak.

Gambar 2.6 mengilustrasikan mekanisme kerja LIME pada klasifikasi teks secara keseluruhan. Terlihat bahwa LIME menghasilkan sampel perturbasi di sekitar instans asli, kemudian melatih model pengganti linier yang tertimbang pada sampel-sampel tersebut. Garis batas lokal yang dihasilkan model pengganti memberikan aproksimasi perilaku model *black-box* di lingkungan terdekat instans, dan kata-kata dengan bobot tertinggi pada model pengganti inilah yang disajikan sebagai penjelasan akhir.



Gambar 2.6. Mekanisme LIME: Aproksimasi Lokal Model *Black-Box* melalui Model Pengganti yang Interpretatif
Sumber: [30]

Salah satu keunggulan LIME dibandingkan metode XAI lain seperti *attention visualization* atau *gradient-based attribution* adalah sifat *model-agnostic* yang memungkinkan penerapan pada arsitektur model apapun tanpa modifikasi internal, sehingga perbandingan pola interpretabilitas antar model yang berbeda dapat dilakukan secara konsisten [30]. Studi yang menerapkan LIME pada sistem deteksi berita palsu berbasis *transformer* menunjukkan bahwa kata-kata dengan bobot tertinggi secara konsisten berkorelasi dengan fitur linguistik yang relevan secara semantik, seperti kata-kata provokatif, klaim yang tidak terverifikasi, atau frasa sensasional [69]. Keterbatasan utama LIME terletak pada potensi ketidakstabilan penjelasan yang dihasilkan akibat sifat stokastik dari proses perturbasi acak, di mana dua eksekusi LIME pada instans yang sama dapat menghasilkan penjelasan yang sedikit berbeda, meskipun hal ini dapat dikelola melalui penentuan *random seed* yang konsisten dan jumlah sampel perturbasi yang memadai [31].

2.4 *Tools*/*Software* yang Digunakan

Implementasi dalam penelitian ini menggunakan bahasa pemrograman Python yang telah menjadi standar dalam pengembangan sistem berbasis *machine learning* dan *deep learning* berkat ekosistem pustaka yang lengkap, termasuk PyTorch sebagai *framework* komputasi, HuggingFace Transformers untuk akses model *pre-trained*, dan scikit-learn untuk evaluasi metrik [70]. Lingkungan pengembangan menggunakan Visual Studio Code sebagai *code editor* utama yang mendukung berbagai ekstensi untuk *debugging* dan integrasi *version control* [71]. Pelatihan model dilakukan pada Google Colaboratory, sebuah platform *notebook* berbasis *cloud* yang menyediakan akses terhadap akselerator GPU secara gratis, sehingga

memungkinkan pelatihan model *deep learning* tanpa memerlukan *hardware* lokal yang mahal [72].