BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Kajian terhadap penelitian terdahulu menjadi dasar penting dalam merumuskan arah penelitian. Melalui kajian ini, perkembangan topik, metode yang telah digunakan, pola penyelesaian masalah, serta kesenjangan penelitian dapat diidentifikasi secara sistematis [43]. Kelebihan dan keterbatasan dari setiap penelitian juga dapat menjadi dasar pertimbangan dalam menentukan pendekatan yang sesuai dengan kebutuhan kasus yang diteliti. Perbandingan antarstudi juga diperlukan untuk mendukung perumusan arah penelitian agar pendekatan yang dipilih tidak sekadar mengikuti metode yang sudah ada, melainkan sesuai dengan kebutuhan kasus [44]. Dengan demikian, arah penelitian yang dirumuskan berdasarkan kajian penelitian terdahulu diharapkan mampu memberikan kontribusi yang relevan bagi pengembangan ilmu pengetahuan.

Kontribusi penelitian dapat dirumuskan melalui pemetaan studi terdahulu yang relevan dengan ruang lingkup kajian. Pada penelitian ini, kajian difokuskan pada studi analisis sentimen berbasis teks (terutama teks *code-mixed*), analisis sentimen berbasis gambar, serta analisis sentimen yang menggabungkan kedua modalitas tersebut. Pada konteks teks *code-mixed*, kajian difokuskan pada dua pendekatan utama, yaitu *direct code-mixed modeling* yang memproses teks *code-mixed* dalam bentuk aslinya dan *translation-based English modeling* yang menerjemahkan teks *code-mixed* ke bahasa Inggris terlebih dahulu sebelum dianalisis lebih lanjut. Setiap studi ditinjau berdasarkan permasalahan, metode dan algoritma, hasil, kelebihan, kekurangan, serta kontribusi yang diberikan. Struktur kajian ini digunakan untuk memperjelas hubungan antara objek penelitian, permasalahan, metode yang digunakan, dan hasil penelitian terdahulu. Melalui susunan tersebut, celah penelitian dapat diidentifikasi secara lebih terarah, khususnya terkait analisis sentimen yang bersifat *code-mixed* dan multimodal. Berdasarkan ruang lingkup tersebut, penelitian terdahulu yang menjadi acuan dalam penelitian ini dirangkum secara sistematis pada Tabel 2.1.

Tabel 2.1 Kajian Penelitian Terdahulu

Penulis	Judul	Sumber Artikel	Masalah/ Kasus	Metode & Algoritma	Hasil Model Terbaik	Kelebihan & Kekurangan	Kontribusi/ Temuan
N. K. Ariyudhi, D. Lestari, dan G. T. Pranoto [24]	*Sentiment Analysis Review Threads Google Play Store with RoBERTa Model*	*Jurnal Nasional Teknik Elektro dan Teknologi Informasi,* Vol. 14, No. 4 (2025)	Klasifikasi sentimen ulasan Threads karena volume *review* besar	**Metode:** Analisis sentimen berbasis teks. **Algoritma:** RoBERTa	**RoBERTa** (akurasi 88%; *F1-score* 90% [positif] dan 84% [negatif])	**Kelebihan:** *robust* menangkap konteks teks ulasan. **Kekurangan:** *overfitting*	RoBERTa *robust* untuk klasifikasi sentimen *user review*
T. D. Van, H. D. Ngoc, dan N. N. Luu-Thuy [45]	*Sentiment Analysis in Code-Mixed Vietnamese-English Sentence-Level Hotel Reviews*	*Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation,* P. 54–61 (2022)	Sulit mengklasifikasi sentimen pada ulasan hotel *Vietnamese–English code-mixed*	**Metode:** *Code-mixed sentiment analysis* dan *translation-based*. **Algoritma:** SVM, MLP, CNN, LSTM, mBERT, XLM-R, LaBSE, RoBERTa	**LaBSE** (*F1-score* 82,24%); **RoBERTa (***F1-score* 80%)	**Kelebihan:** LaBSE kuat untuk *code-mixed*. **Kekurangan:** RoBERTa bergantung pada kualitas translasi	Model lintas-bahasa unggul dibanding *translation-based* pada sentimen *code-mixed*
H. Yang dan K. Li [25]	*Modeling Aspect Sentiment Coherency via Local Sentiment Aggregation*	*EACL 2024,* P. 182–195 (2024)	Sulit menangkap koherensi sentimen antar-aspek dalam satu ulasan	**Metode:** *Aspect-based sentiment classification*. **Algoritma:** LSAE-X, LSAS-X, DeBERTa	**LSAS-X-DeBERTa** (akurasi 86,21% [Laptop]); **LSAE-X-DeBERTa** (akurasi 91,85% [*Restaurant*])	**Kelebihan:** DeBERTa *robust* untuk *aspect-based*. **Kekurangan:** LSA kurang optimal pada satu aspek	LSA berbasis DeBERTa meningkatkan klasifikasi sentimen berbasis aspek

Penulis	Judul	Sumber Artikel	Masalah/ Kasus	Metode & Algoritma	Hasil Model Terbaik	Kelebihan & Kekurangan	Kontribusi/ Temuan
X. Sika, D. Kisbianty, M. Istoningtyas, D. Z. Abidin, dan A. N. Toscany [46]	*Optimized RoBERTa–DeBERTa Ensemble for Multi-Class Sentiment Analysis on Highly Imbalanced Data*	*Jurnal Teknik Informatika (JUTIF),* Vol. 7, No. 2, P. 1964–1980 (2026)	Sulit mengklasifikasi sentimen *multi-class* pada Amazon *review*	**Metode:** *Multi-class sentiment analysis* dan *ensemble learning*. **Algoritma:** BERT, DistilBERT, RoBERTa, DeBERTa-v3.	**RoBERTa–DeBERTa** (akurasi 91%); **DeBERTa-v3 (**akurasi 91%)	**Kelebihan:** RoBERTa–DeBERTa lebih seimbang. **Kekurangan:** terbatas pada *dataset English*.	RoBERTa–DeBERTa meningkatkan deteksi kelas netral pada *imbalance data*
M. H. Hosen, Y. Haque, D. C. Nijhum, dan M. N. Uddin [38]	*XLM-RoBERTa-LIME: A Novel Explainable Framework for Sentiment Analysis on Bangla–English–Hindi Code-Mixed Text*	*Proceedings of the 12th International Conference on Next Generation Computing, Communication, Systems and Security,* P. 44–52 (2025)	Sulit menganalisis sentimen *Bangla–English–Hindi code-mixed* karena variasi ejaan dan struktur tidak baku	**Metode:** *Code-mixed sentiment analysis* berbasis *explainable* AI. **Algoritma:** XLM-R, LIME, SVM, CNN, LSTM, DistilBERT, mBERT	**XLM-R** (akurasi 91,20%, *F1-score* 91,00%) **mBERT (**akurasi 85,60% *F1-score* 85,10%)	**Kelebihan:** XLM-R *robust* pada konteks *code-mixed*. **Kekurangan:** sulit membedakan kelas netral	XLM-R meningkatkan performa dan interpretabilitas analisis sentimen *code-mixed*
S. Alam, M. F. Ishmam, N. H. Alvee, M. S. Siddique, M. A. Hossain, dan A. R. M. Kamal [47]	*BnSentMix: A Diverse Bengali-English Code-Mixed Dataset for Sentiment Analysis*	*Proceedings of the First Workshop on Language Models for Low-Resource Languages,* P. 68–77 (2025)	Keterbatasan *dataset sentiment analysis* untuk teks *Bengali-English code-mixed*	**Metode:** *Code-mixed sentiment analysis* dan *dataset construction*. **Algoritma:** XLM-R, mBERT, BanglaBERT, BanglishBERT	**XLM-R** (akurasi 72,60%, *F1-score* 71,70%)	**Kelebihan:** cocok untuk teks *Bengali-English code-mixed*. **Kekurangan:** hasil *test set* kurang unggul	XLM-R sebagai *baseline* kuat untuk *sentiment analysis code-mixed*
K. Wang [48]	*Research on E-Commerce Platform*	*Proceedings of the 1st*	Klasifikasi gambar produk	**Metode:** *Image classification*.	**ResNet-18** (akurasi 85,90%)	**Kelebihan:***robust* tanpa ekstraksi	ResNet memiliki nilai

Penulis	Judul	Sumber Artikel	Masalah/ Kasus	Metode & Algoritma	Hasil Model Terbaik	Kelebihan & Kekurangan	Kontribusi/ Temuan
	Product Image Recognition Based on ResNet Network	*International* DAML, P. 87–91 (2023)	sulit karena jumlah produk besar dan visual mirip-mirip	**Algoritma:** ResNet-18, ResNet-34, ResNet-50		fitur manual. **Kekurangan:** visual yang mirip bisa tertukar	praktis untuk klasifikasi gambar produk *e-commerce*
J. A. William, D. A. Kristiyanti, H. C. Rustamaji, S. A. Sanjaya, D. Agustria, dan S. Y. Yuliani [29]	*A Comparative Study of Generative AI with CNN ResNet-18 and Transformer Models for Multimodal Sentiment Analysis in E-Commerce Product Review*	*Proceedings of the 2025 8th International CONMEDIA,* Vol. 139, P. 204–209 (2025)	Analisis sentimen berbasis teks pada *e-commerce fashion* kurang andal tanpa bukti visual	**Metode:** *Multimodal sentiment analysis* dan *text-to-image generation*. **Algoritma:** BERT, RoBERTa, FNet, ResNet-18, *Stable Diffusion* 1.5, v2-*Base*, SDXL *Turbo*	**ResNet-18–SDXL *Turbo*** (akurasi 75,18%); **ResNet-18–SDXL *Turbo*–FNet** (akurasi 91,20%)	**Kelebihan:** ResNet-18 *robust* untuk fitur visual. **Kekurangan:** bergantung pada kualitas *synthetic image*	Integrasi fitur visual sintetis dan teks meningkatkan performa klasifikasi sentimen *e-commerce*.\
H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, dan H. Jégou [42]	*Training Data-Efficient Image Transformers & Distillation Through Attention*	*Proceedings of Machine Learning Research,* Vol. 139, P. 10347–10357 (2021)	*Vision Transformer* membutuhkan data dan komputasi besar sehingga kurang efisien	**Metode:** *Image classification* berbasis *visual transformer* dan *distillation through attention*. **Algoritma:** DeiT, ViT, *distillation* token	**DeiT** (akurasi 83,10% [*ImageNet*])	**Kelebihan:** DeiT *robust* tanpa data eksternal. **Kekurangan:** bergantung pada *augmentation* dan *distillation*	DeiT *robust* untuk klasifikasi gambar tanpa data eksternal tambahan
N. Islam, M. M. H. Jony, E. Hasan, S. Sutradhar, A. Rahman, dan M. M. Islam [49]	*EWasteNet: A Two-Stream Data Efficient Image Transformer Approach for E-Waste Classification*	*2023 IEEE 8th International Conference ICSECS,* P. 435–440 (2023)	Sulit mengklasifikasi *e-waste* karena variasi visual perangkat elektronik	**Metode:** *Image classification* berbasis *two-stream visual transformer*. **Algoritma:** DeiT, Sobel *operator*, ASPP, *attention block*	**DeiT** (akurasi 96%)	**Kelebihan:** DeiT bisa menangkap fitur multi-skala. **Kekurangan:** bergantung pada jenis gambar & *dataset*	DeiT mampu mengintegrasikan fitur tepi dan konteks multi-skala untuk klasifikasi *e-waste*

Penulis	Judul	Sumber Artikel	Masalah/ Kasus	Metode & Algoritma	Hasil Model Terbaik	Kelebihan & Kekurangan	Kontribusi/ Temuan
C. N. Karaeng dan D. A. Kristiyanti [50]	*Multimodal Sentiment Analysis Approach Combining Transfer Learning and Generative AI of E-Commerce Product Reviews*	*Proceedings of the 2025 4th ICERA,* P. 1–6 (2025)	Analisis sentimen Tokopedia perlu *multimodal* karena gambar berfungsi sebagai bukti valid ulasan	**Metode:** *Multimodal sentiment analysis* dengan *feature concatenation*. **Algoritma:** BERT, RoBERTa, FNet, CNN, *Stable Diffusion* v2-*base*	**CNN-BERT** (akurasi 90,60%); **CNN-RoBERTa** (akurasi 89,40%) pada data *training*.	**Kelebihan:** CNN-RoBERTa lebih *robust*. **Kekurangan:** CNN *synthetic image* berakurasi rendah	*Hybrid text-image* (*multimodal*) meningkatkan akurasi sentimen *e-commerce*
G. Thakkar, S. Hakimov, dan M. Tadić [51]	*M2SA: Multimodal and Multilingual Model for Sentiment Analysis of Tweets*	*Proceedings of LREC-COLING 2024,* P. 10833–10845 (2024)	Analisis sentimen *multimodal* lintas bahasa terbatas (mayoritas *English only*)	**Metode:** *Multimodal sentiment analysis* dengan *feature concatenation*. **Algoritma:** XLM-R, mBERT, CLIP, DINOv2	**XLM-R+CLIP** (*F1-score* 66,50%); **XLM-R** (*F1-score* 60,69%);	**Kelebihan:** mendukung fusi teks-gambar multibahasa. **Kekurangan:** performa antar bahasa berbeda	XLM-R+CLIP paling unggul untuk *multimodal* multibahasa
R. R. Balqis dan D. A. Kristiyanti [30]	*BERT-DeiT Model: Multimodal Sentiment Analysis for E-Commerce Product Review*	*Proceedings of the 2025 4th ICERA,* P. 221–226 (2025)	Analisis sentimen *e-commerce* sulit karena modalitas data tidak selalu lengkap	**Metode:** *Multimodal sentiment analysis* dengan *feature concatenation*. **Algoritma:** BERT, ALBERT, XLNet, ViT, DeiT, *Stable Diffusion* v2-*base*	**BERT-DeiT** (akurasi 93,49%); **BERT** (akurasi 92,16%); **DeiT** (akurasi 69,60%) pada data *training*	**Kelebihan:** BERT-DeiT lebih *robust*. **Kekurangan:** DeiT sensitif terhadap kualitas *synthetic image*	Integrasi BERT dan DeiT meningkatkan performa dibanding unimodal

Berdasarkan Tabel 2.1, perkembangan analisis sentimen berbasis teks menunjukkan bahwa model *transformer* semakin banyak digunakan karena mampu menangkap konteks kalimat ulasan secara lebih detail dibanding pendekatan konvensional. Pada ulasan aplikasi, RoBERTa menghasilkan akurasi 88% dengan *F1-score* 90% pada kelas positif dan 84% pada kelas negatif, sehingga menunjukkan kemampuan yang cukup *robust* dalam memahami konteks teks ulasan pengguna [24]. Pada tugas *aspect-based sentiment classification*, DeBERTa juga menunjukkan performa kuat melalui LSAS-X-DeBERTa dengan akurasi 86,21% pada *dataset* Laptop dan LSAE-X-DeBERTa dengan akurasi 91,85% pada *dataset Restaurant* [25]. Selain itu, pada data sentimen *multi-class* yang tidak seimbang, kombinasi RoBERTa–DeBERTa memperoleh akurasi 91%, sedangkan DeBERTa-v3 juga mencapai akurasi 91%, sehingga memperlihatkan bahwa RoBERTa dan DeBERTa-v3 relevan digunakan pada data sentimen yang kompleks [46]. Untuk data *code-mixed*, pendekatan *translation-based English* dapat membantu penggunaan model *English-based*, tetapi performanya masih bergantung pada kualitas hasil terjemahan dan karakteristik bahasa, seperti terlihat pada RoBERTa dengan *F1-score* 80% pada ulasan hotel *Vietnamese-English code-mixed* [45]. Sementara itu, pendekatan *direct code-mixed* menunjukkan hasil yang kuat melalui XLM-R dengan akurasi 91,20% dan *F1-score* 91,00% pada teks *Bangla-English-Hindi code-mixed*, serta XLM-R sebagai *baseline* terbaik pada *dataset Bengali-English code-mixed* dengan akurasi 72,60% dan *F1-score* 71,70% [[38][47]. Secara keseluruhan, penelitian terdahulu pada analisis sentimen teks telah menunjukkan keunggulan RoBERTa, DeBERTa-v3, dan XLM-R, tetapi sebagian besar masih berfokus pada teks saja dan belum menggabungkan sentimen *code-mixed* dengan bukti visual dari gambar ulasan produk.

Gambar ulasan produk dalam konteks *e-commerce* berfungsi sebagai bukti visual yang membantu memahami kondisi barang secara lebih nyata, seperti bentuk produk, kemasan, warna, kesesuaian barang, hingga potensi kerusakan yang tidak selalu dijelaskan secara lengkap dalam komentar teks. Pada klasifikasi gambar produk *e-commerce*, ResNet-18 memperoleh akurasi 85,90%, sehingga menunjukkan bahwa arsitektur CNN residual masih relevan untuk mengenali fitur

visual produk tanpa ekstraksi fitur manual [48]. Sebagai dasar model visual berbasis *transformer*, DeiT juga menunjukkan performa yang *robust* dengan akurasi 83,10% pada *ImageNet*, karena dirancang lebih optimal dibanding *Vision Transformer* biasa melalui mekanisme *distillation through attention* [42]. Pada domain lain yang tetap berbasis klasifikasi visual, DeiT juga mencapai akurasi 96% pada klasifikasi *e-waste*, sehingga memperlihatkan bahwa *visual transformer* mampu menangkap fitur tepi, bentuk, dan konteks visual secara lebih luas [49]. Dalam konteks analisis sentimen *e-commerce*, penggunaan ResNet-18 bersama gambar sintetis SDXL Turbo menghasilkan akurasi 75,18%, sedangkan kombinasi ResNet-18–SDXL Turbo–FNet mencapai akurasi 91,20%, yang menunjukkan bahwa integrasi fitur visual dan teks dapat meningkatkan performa klasifikasi sentimen [29]. Pendekatan *hybrid* lain juga menunjukkan hasil kompetitif, seperti CNN-BERT dengan akurasi 90,60% dan CNN-RoBERTa dengan akurasi 89,40%, meskipun performanya masih dipengaruhi kualitas gambar dan representasi visual yang digunakan [50]. Selain itu, kombinasi BERT-DeiT menghasilkan akurasi 93,49%, lebih tinggi dibanding BERT-*only* 92,16% dan DeiT-*only* 69,60%, sehingga menunjukkan bahwa penggabungan teks dan gambar dapat memperkuat sinyal sentimen dibanding penggunaan satu modalitas saja [30]. Secara keseluruhan, hasil penelitian terdahulu menunjukkan bahwa ResNet-18 dan DeiT layak digunakan sebagai pembanding *visual encoder*, namun masih perlu diuji lebih lanjut pada data ulasan produk *e-commerce* yang menggabungkan teks *code-mixed* dan gambar asli ulasan pelanggan.

Kebutuhan pengujian lanjutan pada kombinasi teks *code-mixed*, gambar ulasan, dan model *hybrid* menunjukkan adanya celah penelitian yang masih perlu dikaji. Analisis sentimen berbasis teks sudah banyak dilakukan menggunakan RoBERTa, DeBERTa-v3, XLM-R, LaBSE, mBERT, SVM, dan LSTM, tetapi sebagian besar masih berfokus pada komentar teks tanpa memanfaatkan bukti visual dari gambar ulasan produk [24][45][25][46][38][47]. Di sisi lain, analisis berbasis gambar menggunakan ResNet-18 dan DeiT sudah mampu mengenali fitur visual produk, tetapi gambar saja belum cukup untuk menjelaskan alasan sentimen yang muncul dalam komentar pelanggan [48][42][49]. Pendekatan *hybrid*

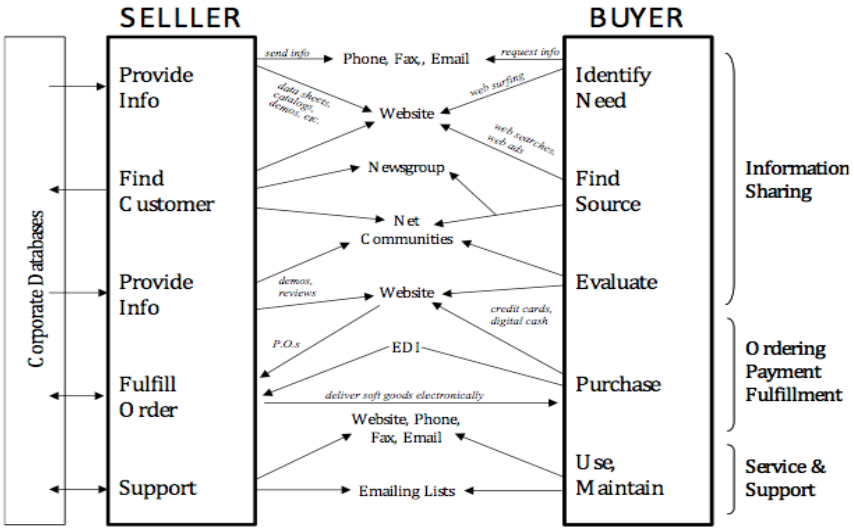
multimodal berbasis teks dan gambar berpotensi menghasilkan pemahaman sentimen yang lebih kaya, tetapi performanya belum selalu konsisten karena dipengaruhi karakteristik data, kualitas gambar, kualitas teks, dan kombinasi encoder yang digunakan [29][50][51][30]. Pada data *code-mixed*, pendekatan *direct code-mixed modeling* dan *translation-based English modeling* juga sudah digunakan, namun pengujian keduanya masih jarang dikaji dalam konteks ulasan produk *e-commerce* yang memiliki data teks dan gambar secara bersamaan [45][38][47]. Celah tersebut menjadi dasar kontribusi penelitian ini, yaitu menerapkan tiga *text encoder* berupa XLM-R, RoBERTa, dan DeBERTa-v3 yang mewakili dua pendekatan utama, yaitu *direct code-mixed modeling* melalui XLM-R serta *translation-based English modeling* melalui RoBERTa dan DeBERTa-v3. Pada sisi gambar, ResNet-18 dan DeiT dipilih sebagai *visual encoder* untuk membandingkan CNN residual berbasis fitur lokal dengan *vision transformer* berbasis relasi visual global. Perbedaan arsitektur tersebut menjadi dasar pembentukan enam model *hybrid code-mixed multimodal* melalui *feature concatenation* untuk membandingkan kemampuan model dalam merepresentasikan teks, gambar, dan gabungan kedua modalitas. Seluruh skenario dievaluasi menggunakan *accuracy*, *precision*, *recall*, *F1-score*, dan *execution time* agar model *hybrid multimodal* terbaik dapat ditentukan secara terukur sesuai karakteristik data ulasan produk *e-commerce* yang bersifat *code-mixed* dan *multimodal*.

2.2 Tinjauan Teori

2.2.1 *E-commerce*

E-commerce (*Electronic Commerce)* merupakan kegiatan jual beli barang atau jasa yang dilakukan melalui *platform* digital dengan memanfaatkan jaringan internet sebagai media transaksi [52]. Perkembangan teknologi membuat proses perdagangan tidak lagi bergantung pada toko fisik karena seluruh aktivitas mulai dari penawaran produk, pencarian informasi, pemesanan, pembayaran, hingga layanan setelah pembelian dapat dilakukan secara *online*. Selain memudahkan transaksi, *e-commerce* juga mendukung interaksi yang lebih cepat dan fleksibel antara penjual dan konsumen melalui berbagai fitur digital. Keberadaan *e-*

commerce memperluas jangkauan pasar karena produk dapat diakses oleh konsumen dari berbagai wilayah dalam waktu yang lebih cepat [53]. Seluruh aktivitas yang berlangsung dalam *e-commerce* menghasilkan berbagai bentuk data, seperti data produk, data transaksi, data pengguna, riwayat pembelian, dan data interaksi pelanggan [54]. Data tersebut dapat digunakan untuk memahami aktivitas belanja, perilaku pengguna, serta pola pelanggan dalam memilih dan membeli produk. Gambar 2.1 menunjukkan alur proses bisnis *e-commerce* yang menghubungkan penjual dan pembeli dalam rangkaian aktivitas belanja digital.



Gambar 2.1 Proses Bisnis *E-Commerce*
Sumber: [52]

Pada Gambar 2.1, alur *e-commerce* dimulai dari penjual yang menampilkan produk melalui *platform* digital agar dapat diakses oleh calon pembeli. Proses berikutnya mencakup pencarian produk, pemilihan produk, transaksi pembelian, dan pemenuhan pesanan hingga produk diterima oleh konsumen. Alur tersebut tidak berhenti pada tahap pembelian saja, karena masih terdapat layanan setelah transaksi seperti konfirmasi penerimaan produk, penanganan keluhan, pengembalian barang, penukaran produk, garansi, serta dukungan informasi setelah pembelian [52]. Rangkaian proses ini memperlihatkan bahwa *e-commerce* membentuk hubungan digital yang terhubung antara penjual, pembeli, dan *platform* dalam satu sistem.

2.2.2 Ulasan Produk

Ulasan produk merupakan tanggapan pelanggan setelah membeli atau menggunakan barang pada *platform e-commerce* [55]. Ulasan ini penting karena membantu calon pembeli mempertimbangkan produk, memberi ruang bagi pembeli untuk menyampaikan kepuasan atau keluhan, menjadi bahan evaluasi bagi penjual, serta membantu *platform* menilai kredibilitas toko dan menjaga kepercayaan pengguna. Karena dibuat langsung oleh pengguna berdasarkan pengalaman pascatransaksi, ulasan produk termasuk *user-generated content* yang biasanya ditulis dengan gaya bebas, informal, dan dapat mengandung campuran beberapa bahasa dalam satu komentar [56]. Ulasan produk terdiri dari *rating*, teks, dan gambar yang saling melengkapi, yaitu *rating* sebagai nilai numerik (bintang 1-5), teks sebagai penjelasan opini, dan gambar sebagai bukti kondisi barang. Kombinasi ketiga komponen tersebut menjadikan ulasan produk sebagai data *multimodal* yang dapat digunakan untuk memahami sentimen pelanggan secara lebih utuh.

2.2.3 Analisis Sentimen

Analisis sentimen adalah proses komputasional untuk mengidentifikasi opini, emosi, atau sikap terhadap suatu objek dan mengelompokkannya ke dalam kelas sentimen tertentu [57]. Berdasarkan jumlah kelasnya, analisis ini dibagi menjadi *binary sentiment analysis* (dua kelas: positif–negatif) dan *multi-class sentiment analysis* (lebih dari dua kelas) [58]. Analisis sentimen tidak hanya diterapkan pada teks, tetapi juga cocok pada data seperti gambar, audio, dan video yang memuat opini atau emosi. Pada *e-commerce*, analisis sentimen digunakan untuk memahami opini pelanggan dengan memanfaatkan isi ulasan karena rating saja tidak cukup merepresentasikan pengalaman pengguna. Teks ulasan sering memiliki keterbatasan seperti ambiguitas bahasa, slang, dan konteks yang tidak lengkap, sehingga gambar dapat ditambahkan sebagai modalitas pendukung untuk memperkuat pemahaman sentimen. Pendekatan yang menggabungkan beberapa modalitas ini dikenal sebagai *multimodal sentiment analysis*, yang membuat analisis tidak hanya bergantung pada satu informasi, tetapi juga pada hubungan antar data yang saling melengkapi [57].

2.2.4 *Code-Mixed*

Code-mixed adalah penggunaan dua bahasa atau lebih dalam satu ujaran, kalimat, atau komentar [59]. Fenomena ini muncul dalam komunikasi modern yang semakin terbuka terhadap bahasa Inggris sebagai bahasa internasional, sehingga banyak orang mencampurkan bahasa lokal mereka dengan bahasa Inggris dalam satu kalimat [60]. Dalam praktiknya, teks *code-mixed* juga sering mengandung transliterasi, singkatan, slang, *typo*, dan struktur kalimat yang tidak baku. Hal ini umum ditemukan pada ulasan produk *e-commerce*, karena pengguna menuliskan pengalaman dengan istilah yang dianggap paling sesuai. Namun, percampuran bahasa tersebut dapat menyulitkan pemahaman konteks, terutama bagi calon pembeli atau penjual yang tidak menguasai bahasa yang digunakan. Penanganan teks *code-mixed* umumnya bisa melalui dua cara, yaitu *direct code-mixed modeling* yang mempertahankan bentuk asli teks, serta *translation-based English modeling* yang menerjemahkan atau menyeragamkan teks ke bahasa Inggris terlebih dahulu sebelum analisis dilakukan [61].

2.2.5 *Transformer Text Encoder*

Transformer text encoder adalah model *deep learning* yang mengubah teks menjadi representasi numerik (vektor fitur) agar bisa diproses dalam tugas pemrosesan bahasa alami [62]. Model ini dikembangkan karena pemahaman teks tidak cukup hanya dari urutan atau frekuensi kata, tetapi perlu melihat hubungan antar kata dalam satu kalimat [63]. Melalui mekanisme *self-attention*, *transformer text encoder* menghasilkan *contextual representation*, yaitu representasi kata yang dibentuk berdasarkan keterkaitan antar token dalam kalimat sebagaimana ditunjukkan pada Rumus 2.1. Pendekatan ini banyak digunakan dalam tugas *Natural Language Processing* (NLP), seperti klasifikasi teks, ekstraksi informasi, pemahaman kalimat, dan analisis sentimen [64]. Karakteristik tersebut membuat *transformer text encoder* cocok untuk data teks yang tidak selalu formal, memiliki variasi penulisan, serta mengandung opini yang maknanya bergantung pada keseluruhan kalimat.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.1)$$

Rumus 2.1 *Self-Attention* pada *Transformer Text Encoder*
Sumber: [63]

Rumus 2.1 menjelaskan bahwa *self-attention* menghitung hubungan antar token dalam teks menggunakan tiga komponen, yaitu *query* (Q), *key* (K), dan *value* (V). *Query* (Q) adalah token yang mencari hubungan konteks dengan token lain, sedangkan *key* (K) adalah token pembanding untuk mengukur tingkat keterkaitannya. Notasi K^T adalah *key* yang ditransposisikan untuk dikalikan dengan *query* sehingga menghasilkan skor keterkaitan QK^T, yang kemudian dibagi dengan $\sqrt{d_k}$ (dimensi *key*) agar nilainya tidak terlalu besar. *Softmax* mengubah skor menjadi bobot perhatian, lalu bobot tersebut dikalikan dengan *value* (V) untuk menghasilkan representasi teks yang mempertimbangkan konteks antar kata dalam kalimat [63].

2.2.6 *Visual Encoder*

Visual encoder adalah model *deep learning* yang mengubah gambar menjadi representasi numerik (vektor fitur) untuk berbagai tugas visi komputer [65]. Model ini dikembangkan untuk mengekstrak ciri visual langsung dari citra yang berdimensi tinggi, berpola beragam, dan memiliki hubungan spasial yang sulit ditangkap menggunakan fitur manual [66]. Secara umum, *visual encoder* mengubah input gambar menjadi representasi fitur yang merangkum informasi visual penting, sebagaimana ditunjukkan pada Rumus 2.2. Representasi ini mencakup berbagai aspek visual seperti bentuk, warna, tekstur, objek, dan kondisi gambar, sehingga cocok untuk klasifikasi visual maupun tugas *multimodal*, termasuk analisis sentimen berbasis gambar yang memanfaatkan isyarat visual untuk menentukan polaritas ulasan [67].

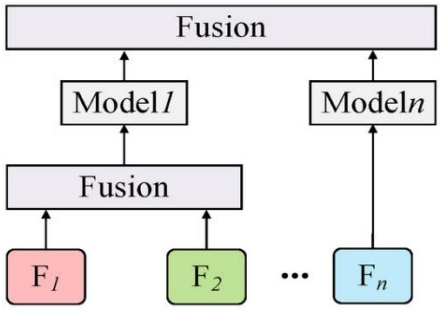
$$y = f_\theta(x) \quad (2.2)$$

Rumus 2.2 Representasi Fitur pada *Visual Encoder*
Sumber: [65]

Rumus 2.2 menunjukkan bahwa gambar masukan dipetakan oleh *encoder* menjadi representasi fitur. x adalah gambar masukan, f_θ adalah fungsi *encoder* dengan parameter yang dipelajari selama pelatihan, yaitu θ, sedangkan y merupakan representasi fitur yang merangkum ciri visual paling relevan dari gambar. Pemodelan ini menegaskan bahwa *visual encoder* tidak langsung menggunakan label, tetapi terlebih dahulu mengubah piksel menjadi fitur yang siap digunakan untuk klasifikasi atau digabungkan dengan modalitas lain [59].

2.2.7 *Hybrid Multimodal*

Konsep *hybrid* berarti menggabungkan beberapa metode atau arsitektur dalam satu sistem, sedangkan *hybrid multimodal* memperluas konsep ini dengan menggabungkan berbagai modalitas data (teks, gambar, audio) untuk menghasilkan representasi yang lebih lengkap dan akurat [68]. Model ini dikembangkan karena meningkatnya data digital yang heterogen dan berdimensi tinggi, sehingga pola tiap modalitas perlu dipelajari dan digabungkan [69]. Secara umum, prosesnya dimulai dari pembentukan vektor fitur tiap modalitas, lalu digabungkan menggunakan *early fusion*, *late fusion*, atau *attention-based fusion* dengan pemberian bobot pada informasi lintas modalitas [68]. Pada analisis sentimen, pendekatan ini cocok untuk kasus dengan sinyal opini yang tersebar di teks, gambar, audio, atau video seperti ulasan produk *online*, karena penggabungan beberapa modalitas menghasilkan representasi sentimen yang lebih lengkap dibandingkan hanya satu modalitas. Mekanisme penggabungan fitur dalam model *hybrid multimodal* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Arsitektur *Mid-Level Fusion* pada *Hybrid Multimodal*
Sumber: [70]

Gambar 2.2 menunjukkan proses *hybrid fusion* yang dilakukan setelah fitur tiap modalitas (F1 hingga Fn) diekstraksi namun sebelum keputusan akhir. Pada tahap ini, fitur digabungkan pada level menengah (*mid-level*) untuk menangkap informasi yang saling melengkapi sekaligus mempertahankan karakteristik masing-masing modalitas, lalu diproses lebih lanjut untuk menghasilkan *output* yang lebih komprehensif [70].

2.2.8 *Data Labeling*

Data labeling adalah proses pemberian label target pada data agar data mentah berubah menjadi data beranotasi yang siap dipakai dalam *supervised learning*. Dalam *supervised learning*, label digunakan sebagai acuan hasil agar model bisa mempelajari hubungan antara input dan target, lalu menerapkan pola tersebut saat memproses data baru [71]. Kualitas label perlu dijaga karena kesalahan atau ketidakkonsistenan dapat menurunkan kualitas data *training* dan memengaruhi hasil prediksi model [72]. Proses *data labeling* biasanya dilakukan dengan memberi label secara manual oleh manusia atau secara otomatis menggunakan algoritma yang sudah dilatih untuk mengenali dan menentukan label berdasarkan fitur data. Pada analisis sentimen, data umumnya dilabeli positif, negatif, dan netral, tetapi banyak penelitian hanya memakai dua label (positif dan negatif) karena lebih sederhana, mengurangi bias, dan meningkatkan akurasi dengan fokus pada polaritas yang jelas [73].

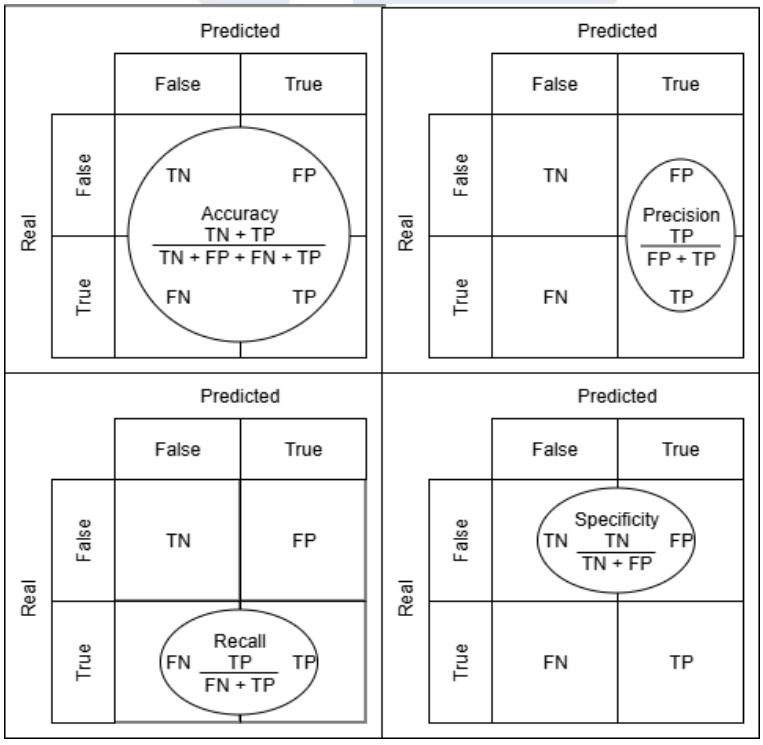
2.2.9 *Data Splitting*

Data splitting adalah proses membagi *dataset* ke beberapa *subset* agar pelatihan (*training*), pengaturan (*validation*), dan pengujian (*testing*) model dilakukan pada data yang berbeda [74]. *Data training* digunakan untuk melatih model dalam mempelajari pola, *data validation* digunakan untuk mengecek dan memilih konfigurasi selama pengembangan, sedangkan *data testing* digunakan untuk menguji model pada data baru yang belum pernah dilihat agar hasilnya lebih akurat [74]. Pada skema *hold-out*, *dataset* dibagi menjadi *training-validation-testing* dengan rasio seperti 70:15:15 atau 80:10:10. Namun rasio 80:10:10 terbukti optimal pada kasus *sentiment analysis* karena memberikan data *training* yang lebih

besar serta tetap menyediakan *validation* dan *testing* yang terpisah untuk evaluasi. Sebagai alternatif, skema *cross-validation* juga bisa digunakan dengan membagi data latih menjadi beberapa bagian untuk *training* dan *validation* secara berulang, sementara *test set* tetap dipisahkan untuk menjaga objektivitas evaluasi [75].

2.2.10 Metrik Evaluasi

Metrik evaluasi adalah ukuran kuantitatif untuk menilai kinerja model berdasarkan kesesuaian hasil prediksi dengan data sebenarnya, termasuk kemampuannya mengenali pola pada data baru [76]. Pemilihan metrik yang tepat diperlukan agar hasil evaluasi tidak hanya menunjukkan jumlah prediksi benar, tetapi juga mencerminkan kualitas model dari berbagai aspek. Dalam evaluasi model, *confusion matrix* digunakan sebagai dasar perhitungan dengan menampilkan hubungan antara label aktual dan prediksi melalui empat komponen, yaitu TP (*True Positive*), FP (*False Positive*), TN (*True Negative*), dan FN (*False Negative*). Hubungan komponen *confusion matrix* dengan metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *specificity* ditunjukkan pada Gambar 2.3.



Gambar 2.3 *Confusion Matrix*
Sumber: [77]

Berdasarkan Gambar 2.3, *confusion matrix* dianalisis dengan membandingkan posisi data pada baris *Real* dan kolom *Predicted*, di mana nilai diagonal menunjukkan prediksi yang benar sedangkan nilai di luar diagonal menunjukkan kesalahan klasifikasi. FP terjadi ketika data negatif diprediksi sebagai positif, sedangkan FN terjadi ketika data positif diprediksi sebagai negatif, sehingga keduanya menggambarkan pola kesalahan model dalam mengenali sentimen [77]. Hasil *confusion matrix* serta pengukuran waktu komputasi menghasilkan metrik evaluasi berupa *accuracy*, *precision*, *recall*, *F1-score*, *execution time* untuk menilai kinerja model secara menyeluruh. Berikut adalah penjelasannya:

1. *Accuracy*

   Accuracy adalah metrik yang mengukur jumlah prediksi benar dibandingkan dengan seluruh data yang diuji [78]. Metrik ini cocok digunakan ketika distribusi kelas relatif seimbang karena nilai *accuracy* menunjukkan gambaran umum ketepatan prediksi model. Semakin tinggi nilai *accuracy*, semakin besar jumlah data yang berhasil diklasifikasikan dengan benar. Rumus untuk menghitung *accuracy* dapat dilihat pada Rumus 2.3.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

Rumus 2.3 *Accuracy*
Sumber: [78]

2. *Precision*

   Precision adalah metrik yang mengukur seberapa tepat model ketika memprediksi suatu data sebagai kelas positif [76]. Perhitungan ini membandingkan prediksi positif yang benar dengan seluruh data yang diprediksi positif. Semakin tinggi nilai *precision*, semakin sedikit kesalahan model dalam memberi label positif pada data yang sebenarnya bukan positif. Rumus menghitung *precision* dapat dilihat pada Rumus 2.4.

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

Rumus 2.4 *Precision*
Sumber: [76]

3. *Recall*

   Recall adalah metrik yang mengukur kemampuan model dalam menemukan seluruh data yang sebenarnya termasuk kelas positif [78]. Perhitungan ini membandingkan prediksi positif yang benar dengan seluruh data positif yang sebenarnya ada. Semakin tinggi nilai *recall*, semakin kecil jumlah data positif yang terlewat atau salah diprediksi sebagai kelas lain. Rumus untuk menghitung *recall* dapat dilihat pada Rumus 2.5.

$$Recall = \frac{TP}{TP + FN} \tag{2.5}$$

Rumus 2.5 *Recall*
Sumber: [78]

4. *F1-score*

   F1-score adalah metrik yang menggabungkan *precision* dan *recall* melalui rata-rata harmonik. Metrik ini penting ketika data memiliki ketidakseimbangan kelas karena tidak hanya melihat jumlah prediksi benar secara umum, tetapi juga menilai keseimbangan antara ketepatan prediksi positif dan kemampuan menemukan kelas positif [77]. Semakin tinggi nilai *F1-score*, semakin seimbang performa model dalam menjaga *precision* dan *recall*. Rumus menghitung *F1-score* dapat dilihat pada Rumus 2.6.

$$F1 - Score = 2 * \left(\frac{Precision * Recall}{Precision + Recall}\right) \tag{2.6}$$

Rumus 2.6 *F1-score*
Sumber: [77]

5. *Execution Time*

   Execution time adalah metrik yang menghitung lama waktu proses model berjalan dari awal hingga selesai. Metrik ini digunakan sebagai pertimbangan tambahan karena model dengan hasil klasifikasi tinggi tetap perlu dilihat dari sisi waktu komputasi, terutama ketika beberapa model dibandingkan dalam skenario eksperimen yang sama [79]. Semakin kecil nilai *execution time*, semakin cepat proses model diselesaikan. Rumus untuk menghitung *execution time* dapat dilihat pada Rumus 2.7.

$$Execution\ Time = End\ Time - Start\ Time \quad (2.7)$$

Rumus 2.7 *Execution Time*
Sumber: [79]

2.2.11 *User Acceptance Testing* (UAT)

User Acceptance Testing (UAT) merupakan tahap pengujian penerimaan pengguna yang dilakukan untuk memastikan bahwa aplikasi telah berjalan sesuai kebutuhan pengguna akhir sebelum digunakan secara lebih luas [80]. Pada pengembangan sistem informasi, UAT berfokus pada kesesuaian fungsi aplikasi, alur penggunaan, dan hasil keluaran sistem terhadap kebutuhan pengguna, sehingga pengujian tidak hanya menilai apakah fitur dapat dijalankan, tetapi juga apakah fitur tersebut dapat diterima oleh pengguna [81]. Pelaksanaan UAT dapat dilakukan melalui skenario pengujian fitur, perbandingan antara *expected result* dan *actual result*, serta kuesioner dengan skala penilaian tertentu untuk mengukur tingkat penerimaan pengguna terhadap sistem [81]. Hasil UAT biasanya dinyatakan dalam status keberhasilan fitur dan persentase penerimaan pengguna, sehingga dapat digunakan sebagai dasar penilaian apakah aplikasi sudah layak digunakan sesuai tujuan pengembangannya.

2.3 *Framework* dan Algoritma

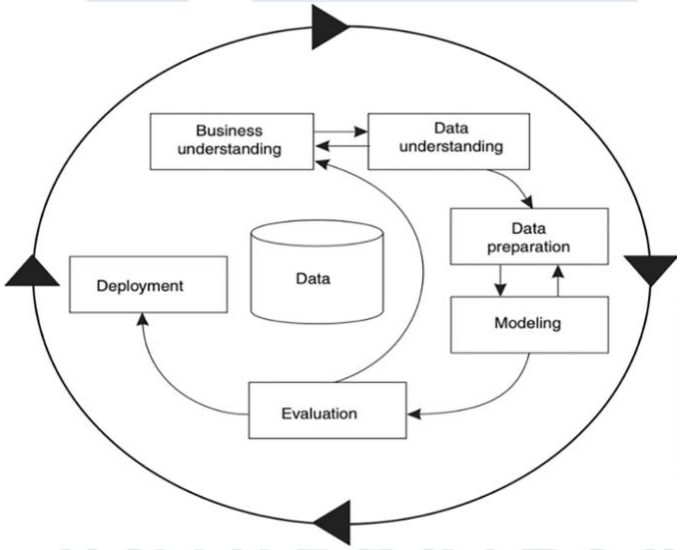
Kerangka kerja (*framework*) dan algoritma menjadi fondasi utama dalam proses pengolahan data dan pemecahan masalah secara sistematis. Sejalan dengan hal tersebut, penelitian ini memanfaatkan CRISP-DM sebagai *framework data mining*, serta XLM-R, RoBERTa, DeBERTa-v3, ResNet-18, dan DeiT sebagai algoritma dalam pemrosesan data teks dan gambar. Berikut merupakan uraian lebih lanjut mengenai *framework* dan algoritma yang digunakan dalam penelitian ini:

2.3.1 *Framework* CRISP-DM

Cross-Industry Standard Process for Data Mining (CRISP-DM) merupakan *framework data mining* yang menjelaskan tahapan proyek berbasis data mulai dari pemahaman tujuan sampai penyajian hasil analisis [82]. CRISP-DM bersifat umum, terstruktur, fleksibel, serta dapat digunakan pada berbagai bidang seperti *data*

mining, *machine learning*, *artificial intelligence*, dan *deep learning* [83]. Karakteristik tersebut menjadikan CRISP-DM relevan digunakan sebagai kerangka kerja analisis data yang sistematis melalui enam fase yang saling berkaitan.

CRISP-DM memiliki alur kerja berbentuk siklus yang berpusat pada data sebagai sumber utama proses analisis, seperti ditunjukkan pada Gambar 2.4. Alur tersebut terdiri dari *Business Understanding*, *Data Understanding*, *Data Preparation*, *Modeling*, *Evaluation*, dan *Deployment* [84]. Arah panahnya menunjukkan hubungan antar fase yang bersifat iteratif, sehingga prosesnya tidak selalu berjalan linear karena hasil pada satu fase dapat memerlukan penyesuaian pada fase sebelumnya [85]. Keterkaitan antarfase dalam siklus ini menegaskan bahwa keputusan analitis pada CRISP-DM dipengaruhi oleh kondisi data, kesiapan model, dan hasil evaluasi yang diperoleh selama proses berlangsung. Bentuk siklus pada bagian luarnya juga menunjukkan bahwa hasil akhir *data mining* dapat menjadi dasar bagi proses analisis berikutnya.



Gambar 2.4 Alur Kerangka Kerja CRISP-DM
Sumber: [83]

Uraian mengenai enam fase utama dalam kerangka kerja CRISP-DM yang ditunjukkan pada Gambar 2.4 dijelaskan sebagai berikut [83]:

1. *Business Understanding*

   Business Understanding merupakan tahap awal untuk memahami masalah, tujuan, kebutuhan, dan batasan proyek *data mining*. Tahap ini bertujuan

mengubah kebutuhan umum menjadi tujuan analisis data yang jelas, spesifik, dan terukur melalui aktivitas seperti penentuan dan perumusan tujuan *data mining*, identifikasi kondisi awal, dan penyusunan rencana kerja. Pada penelitian ini, tahap *business understanding* digunakan sebagai dasar untuk merumuskan masalah, tujuan, ruang lingkup, dan kebutuhan analisis sentimen pada ulasan produk *e-commerce*.

2. *Data Understanding*

   Data Understanding merupakan tahap untuk mengenali sumber, struktur, isi, dan kualitas data sebelum proses pengolahan lebih lanjut. Tahap ini bertujuan memperoleh pemahaman awal terhadap karakteristik data melalui pengumpulan data, deskripsi atribut, eksplorasi, pemeriksaan distribusi, dan pengecekan kualitas data. Pada penelitian ini, tahap *data understanding* digunakan untuk memahami karakteristik ulasan produk, rating, teks, gambar, metadata, serta komposisi *code-mixed dataset*.

3. *Data Preparation*

   Data Preparation merupakan tahap untuk menyiapkan data agar bersih, relevan, konsisten, dan sesuai dengan kebutuhan pemodelan. Tahap ini mencakup aktivitas seperti seleksi data, pembersihan data, transformasi, integrasi, pembentukan label, dan pembagian data untuk menghasilkan *dataset* akhir yang siap digunakan pada proses *training*, *validation*, dan *testing*. Pada penelitian ini, tahap *data preparation* meliputi *cleansing* data teks, pemilihan *image review* paling valid, *data balancing*, *data labeling*, *text translation*, dan *data splitting*.

4. *Modeling*

   Modeling merupakan tahapan untuk memilih, membangun, melatih, dan menguji teknik pemodelan yang sesuai dengan tujuan analisis dan data yang digunakan. Tahap ini mencakup pemilihan algoritma, pengaturan parameter, pelatihan, validasi, dan penyimpanan model untuk menghasilkan model terlatih beserta konfigurasi pemodelannya. Dengan demikian, tahap *modeling* menjadi bagian yang mengubah data siap pakai menjadi model prediksi

melalui proses eksperimen yang terukur dan dapat dievaluasi. Pada penelitian ini, tahap *modeling* meliputi pembangunan model *text-only*, *image-only*, dan model *hybrid multimodal*.

5. *Evaluation*

   Evaluation merupakan tahap untuk menilai kesesuaian model dan hasil analisis terhadap tujuan yang telah dirumuskan pada tahap awal. Tahap ini mencakup pengukuran kinerja model, analisis hasil setiap skenario, pemeriksaan kesalahan klasifikasi, dan peninjauan proses *data mining* untuk memastikan hasil pemodelan sesuai secara teknis maupun kebutuhan analisis. Pada penelitian ini, tahap *evaluation* digunakan untuk menilai performa model sekaligus menentukan model *hybrid multimodal* yang paling optimal sebagai hasil akhir penelitian.

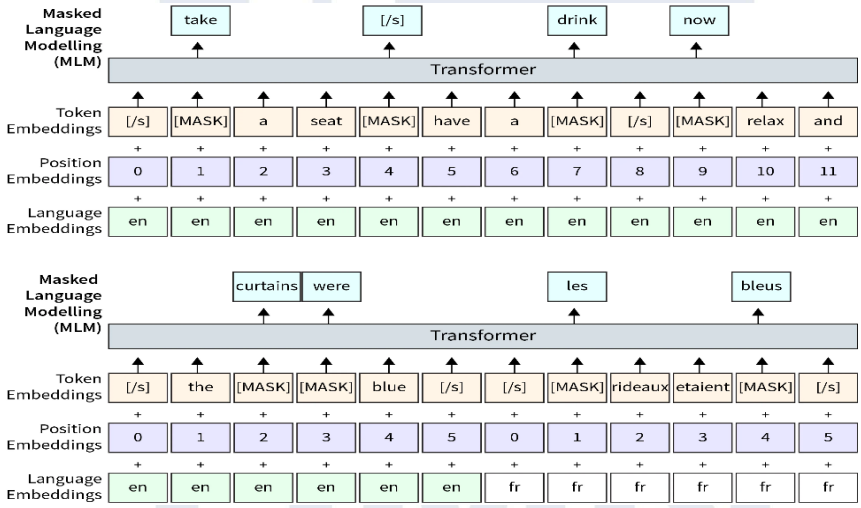
6. *Deployment*

   Deployment merupakan tahap untuk menyajikan hasil *data mining* agar dapat digunakan sebagai pengetahuan, laporan, rekomendasi, atau dasar pengambilan keputusan. Tahap ini mencakup penyusunan laporan akhir, penyajian hasil evaluasi, dokumentasi proses, dan penyampaian kesimpulan dari hasil analisis. Pada penelitian ini, tahap *deployment* dilakukan melalui web sederhana untuk menampilkan hasil prediksi sentimen, algoritma yang digunakan, dan performa model pada data uji.

2.3.2 XLM-R

XLM-R (XLM-RoBERTa) merupakan *transformer text encoder* yang menghasilkan representasi teks kontekstual melalui *masked language modeling*, yaitu teknik *pre-training* dengan menutupi sebagian token dalam kalimat agar model belajar memprediksi token tersebut berdasarkan konteks di sekitarnya [86]. Model ini dikembangkan sebagai perluasan dari XLM dan RoBERTa (*Robustly Optimized BERT Pretraining Approach*) untuk menjawab keterbatasan model sebelumnya (terutama mBERT), dalam menangani banyak bahasa, variasi aksara, serta teks tidak baku. Secara konsep, XLM-R menggabungkan kemampuan representasi lintas bahasa dari XLM dengan strategi *pre-training* RoBERTa,

kemudian dilatih menggunakan korpus CommonCrawl berskala besar agar mampu mempelajari pola bahasa dari berbagai sumber secara lebih luas [87]. Keunggulan XLM-R terletak pada *pre-training* berskala besar dan kemampuan transfer lintas bahasa, sehingga relevan untuk tugas klasifikasi teks, analisis sentimen, deteksi ujaran kebencian, serta pemrosesan teks *multilingual* maupun *code-mixed* pada berbagai domain data [88]. Cara kerja XLM-R dimulai dari proses tokenisasi *subword*, pembentukan representasi numerik token, pemrosesan melalui lapisan *transformer encoder*, hingga menghasilkan representasi teks kontekstual. Proses tersebut didukung oleh mekanisme *self-attention* yang membantu model memahami hubungan antarkata berdasarkan konteks kalimat secara menyeluruh [87]. Keterkaitan antara mekanisme *self-attention* dan *masked language modeling* pada arsitektur XLM-R ditunjukkan pada Gambar 2.5.



Gambar 2.5 Arsitektur XLM-R sebagai *Text Encoder*
Sumber: [88]

Arsitektur XLM-R pada Gambar 2.5 menunjukkan alur pemrosesan teks dari tahap pembentukan representasi input hingga masuk ke lapisan *transformer*. Pada tahap awal, setiap token direpresentasikan melalui *token embeddings*, *position embeddings*, dan *language embeddings* agar model dapat mengenali makna token, posisi dalam kalimat, serta informasi bahasa yang digunakan. Token [*MASK*] digunakan dalam proses *masked language modeling* dengan mekanisme prediksi token yang dinyatakan pada Rumus 2.8. Contoh pada bagian atas Gambar 2.5 menunjukkan pemrosesan satu rangkaian kalimat, sedangkan contoh pada bagian

bawah menunjukkan dua segmen bahasa yang tetap diproses dalam satu kerangka *transformer* [88].

$$\mathcal{L}_{\mathcal{MLM}} = -\sum_{i \in M} \log P_{\theta}\left(x_i \mid x_{\setminus M} \right) \tag{2.8}$$

Rumus 2.8 Fungsi *Masked Language Modeling* pada XLM-R
Sumber: [89]

Keterangan:

1) $\mathcal{L}_{\mathcal{MLM}}$ merupakan nilai *loss* pada *masked language modeling*.
2) M merupakan posisi token yang dimasking.
3) i merupakan indeks token yang dimasking.
4) x_i merupakan token asli yang diprediksi model.
5) $x_{\setminus M}$ merupakan teks masukan dengan token *mask* disembunyikan.
6) $P_{\theta}\left(x_i \mid x_{\setminus M} \right)$ merupakan probabilitas prediksi token oleh model.
7) θ merupakan parameter model XLM-R.
8) $-\sum$ merupakan penjumlahan negatif seluruh token *mask*.

Rumus 2.8 menjelaskan fungsi *masked language modeling* untuk melatih XLM-R dalam memprediksi token yang ditutup pada suatu urutan teks. Proses tersebut dimulai dari pemilihan beberapa posisi token dalam himpunan M, kemudian token asli pada posisi tersebut disembunyikan dari masukan model. Model menghitung probabilitas $P_{\theta}(x_i \mid x_{\setminus M})$ untuk menebak kembali token asli x_i berdasarkan konteks token lain yang masih tersedia [89]. Nilai *loss* menurun saat probabilitas prediksi token asli meningkat, sehingga model menghasilkan representasi teks yang lebih sesuai dengan konteks kalimat. Selain fungsi *masked language modeling*, XLM-R juga menggunakan mekanisme *language sampling* pada Rumus 2.9 untuk mengatur peluang pengambilan data tiap bahasa.

$$p_l = \frac{n_l}{\sum_{k=1}^{L} n_k} \tag{2.9}$$

$$q_l = \frac{p_l^{\alpha}}{\sum_{k=1}^{L} p_k^{\alpha}}$$

Rumus 2.9 *Sampling* Bahasa pada XLM-R
Sumber: [86]

Keterangan:

1) p_l merupakan proporsi awal data bahasa ke-l.
2) q_l merupakan probabilitas *sampling* bahasa ke-l.
3) n_l merupakan jumlah data bahasa ke-l.
4) L merupakan jumlah bahasa dalam korpus.
5) k merupakan indeks bahasa dalam korpus.
6) α merupakan parameter *smoothing* untuk *sampling* bahasa.

Strategi *sampling* bahasa pada XLM-R dilakukan melalui dua tahap perhitungan yang dinyatakan pada Rumus 2.9. Tahap pertama menghitung proporsi awal bahasa berdasarkan jumlah datanya, kemudian tahap kedua menyesuaikannya menjadi probabilitas sampling q_l menggunakan parameter *smoothing* α agar bahasa dengan data kecil tetap memiliki peluang. Tahapan tersebut membantu proses *pre-training* XLM-R menjadi lebih seimbang karena tidak hanya didominasi oleh bahasa dengan jumlah data besar. Tahapan kerja XLM-R secara ringkas ditunjukkan pada Tabel 2.2 melalui *pseudocode* proses *pre-training* model.

Tabel 2.2 *Pseudocode* XLM-R
Sumber: [89]

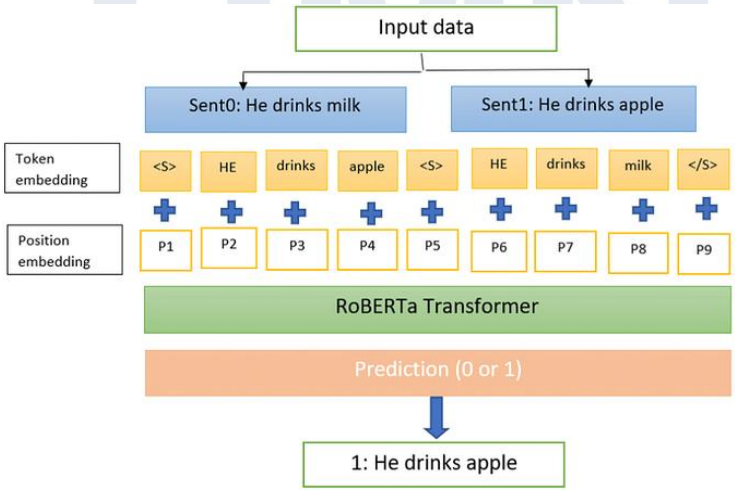
Input	*Dataset* teks asli $D = \{(x_i, y_i)\}_{i=1}^{n}$, dengan x_i sebagai teks masukan dan y_i sebagai label kelas.
(1)	Siapkan *tokenizer* dan model XLM-R hasil *pre-training masked language modeling*.
(2)	Ambil setiap teks x_i dari *dataset*.
(3)	Lakukan tokenisasi *subword* pada x_i.
(4)	Bentuk *input_ids* dan *attention_mask* dari hasil tokenisasi.
(5)	Masukkan *input_ids* dan *attention_mask* ke dalam XLM-R.
(6)	Proses token melalui lapisan *transformer encoder* berbasis *self-attention*.
(7)	Ambil representasi akhir teks h_{cls} dari *output* XLM-R.
(8)	Masukkan h_{cls} ke *classification head*.
(9)	Hitung probabilitas kelas menggunakan fungsi *softmax*, lalu diperoleh prediksi $\hat{y}_i$.
(10)	Saat *training*, bandingkan $\hat{y}_i$ dengan label asli y_i untuk hitung nilai *loss* klasifikasi.
(11)	Perbarui parameter model berdasarkan nilai *loss* sampai *training* selesai.
Output	Representasi teks h_{cls} dan hasil prediksi kelas $\hat{y}_i$.

Pseudocode pada Tabel 2.2 menjelaskan alur XLM-R sebagai *text encoder* untuk mengubah teks menjadi representasi numerik pada tugas klasifikasi. Proses

dimulai dari tokenisasi teks menjadi *subword*, pembentukan *input_ids* dan *attention_mask*, lalu pemrosesan melalui lapisan *transformer encoder* untuk menghasilkan representasi kontekstual. *Attention_mask* digunakan agar token *padding* tidak memengaruhi representasi, sedangkan representasi akhir *h_cls* diteruskan ke *classification head* dan fungsi *softmax* untuk menghasilkan probabilitas kelas. Dengan alur tersebut, XLM-R mampu membentuk representasi teks yang kontekstual dan siap digunakan untuk prediksi sentimen [89].

2.3.3 RoBERTa

RoBERTa atau *Robustly Optimized BERT Pretraining Approach* merupakan *transformer text encoder* yang dikembangkan untuk menghasilkan representasi teks kontekstual melalui proses *pre-training* yang lebih optimal [84]. Model ini dikembangkan untuk menyempurnakan BERT, terutama melalui penghapusan *next sentence prediction*, penerapan *dynamic masking*, penggunaan data pelatihan yang lebih besar, serta pengaturan *hyperparameter* yang lebih kuat [85]. RoBERTa banyak digunakan dalam klasifikasi teks, analisis sentimen, deteksi emosi, dan pemrosesan opini karena mampu memahami konteks kiri dan kanan dalam kalimat secara bersamaan [84]. Sebagai model *English-based*, RoBERTa lebih sesuai untuk teks berbahasa Inggris karena proses *pre-training* utamanya menggunakan korpus bahasa Inggris. Cara kerja RoBERTa dalam mengubah teks menjadi token hingga menghasilkan prediksi kelas ditampilkan pada Gambar 2.6.



Gambar 2.6 Arsitektur RoBERTa sebagai *Text Encoder*
Sumber: [90]

Arsitektur RoBERTa pada Gambar 2.6 menunjukkan alur pemrosesan teks dari input data hingga menghasilkan prediksi kelas. Input data pada Gambar 2.6 terdiri dari dua segmen kalimat, yaitu *Sent0* dan *Sent1*, kemudian setiap kata diubah menjadi *token embedding* dan digabungkan dengan *position embedding*. Simbol khusus <S > dan </S> digunakan sebagai penanda awal dan akhir urutan token, sedangkan P1 sampai P9 menunjukkan posisi token dalam urutan teks. Gabungan *token embedding* dan *position embedding* diproses oleh RoBERTa *Transformer* untuk membentuk representasi kontekstual, lalu representasi tersebut diteruskan ke lapisan prediksi untuk menghasilkan *output* berupa kelas 0 atau 1. Alur tersebut menunjukkan bahwa RoBERTa dapat menerima input teks, membentuk representasi melalui *transformer encoder*, dan menghasilkan *output* klasifikasi berdasarkan konteks urutan token [91]. Pembelajaran representasi pada tahap *pre-training* diperkuat melalui *dynamic masking* pada Rumus 2.10.

$$\mathcal{L}_{\text{RoBERTa-DM}}(\theta) = -\frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} E_{M\sim q(M|x)}\left[\sum_{i\in M}\log P_\theta\left(x_i \mid x_{\setminus M}\right)\right] \qquad (2.10)$$

Rumus 2.10 Fungsi *Dynamic Masking* pada RoBERTa
Sumber: [91]

Keterangan:

1) $\mathcal{L}_{\text{RoBERTa-DM}}(\theta)$ merupakan nilai *loss* RoBERTa dengan *dynamic masking*.
2) θ merupakan parameter model RoBERTa.
3) $\mathcal{D}$ merupakan *dataset pre-training*.
4) x merupakan urutan teks masukan.
5) M merupakan posisi token yang dimasking secara dinamis.
6) $q(M \mid x)$ merupakan distribusi pemilihan posisi *mask* pada teks x.
7) $E_{M\sim q(M|x)}$ merupakan ekspektasi dari berbagai pola *mask*.
8) i merupakan indeks token dalam posisi *mask*.
9) x_i merupakan token asli yang diprediksi model.
10) $x_{\setminus M}$ merupakan teks setelah token pada posisi M dimasking.
11) $P_\theta\left(x_i \mid x_{\setminus M}\right)$ merupakan probabilitas prediksi token asli berdasarkan konteks.
12) log merupakan fungsi logaritma untuk menghitung *loss*.

13) $-\sum$ merupakan penjumlahan negatif untuk meminimalkan kesalahan prediksi.

Rumus 2.10 menjelaskan fungsi objektif *dynamic masking* pada RoBERTa, yaitu mekanisme pelatihan yang membuat pola *mask* berbeda dari BERT karena posisi token yang dimasking tidak ditetapkan secara statis pada tahap *preprocessing*. Pada RoBERTa, pola *mask* M dihasilkan secara dinamis setiap kali urutan teks x dimasukkan ke model, yang dinyatakan melalui $M \sim q(M \mid x)$. Setelah posisi *mask* M dipilih, model menerima teks $x_{\setminus M}$ dan menghitung probabilitas $P_\theta(x_i \mid x_{\setminus M})$ untuk memprediksi token asli x_i pada setiap posisi yang dimasking. Nilai *loss* diperoleh dari penjumlahan negatif *log-probability* seluruh token *mask* pada data *training*, sehingga parameter θ diperbarui agar model semakin mampu memahami konteks kata, sesuai dengan karakteristik RoBERTa yang menggunakan *dynamic masking*, *full-sentences* tanpa *NSP* (*Next Sentence Prediction*) *loss*, *batch* besar, dan *byte-level* BPE (*Byte Pair Encoding*) [91]. Tahapan kerja RoBERTa ditunjukkan pada Tabel 2.3 melalui *pseudocode* model.

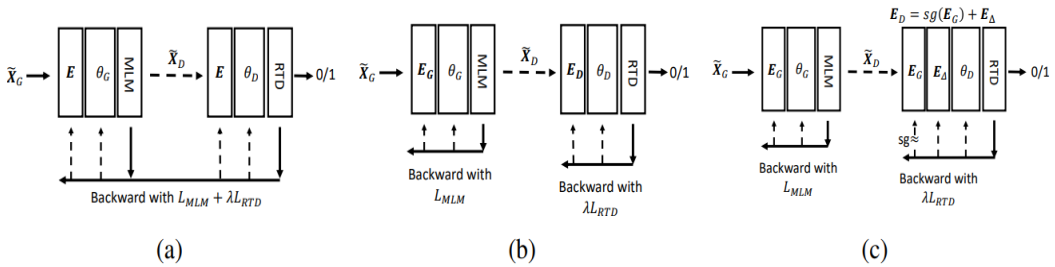
Tabel 2.3 *Pseudocode* RoBERTa
Sumber: [90]

Input	*Dataset* teks Inggris $D_{en} = \{(x_j, y_j)\}_{j=1}^{N}$, *vocabulary byte-level* BPE V_{BPE}, parameter awal RoBERTa θ, distribusi *dynamic masking* $q(M \mid x)$, jumlah *epoch* E, ukuran *mini-batch* B, dan jumlah kelas sentimen C.
(1)	Inisialisasi parameter RoBERTa θ dan *classification head* (W_c, b_c).
(2)	Untuk setiap *epoch* $e = 1, \ldots, E$, bentuk *mini-batch* besar $\mathcal{B} \subset D_{en}$.
(3)	Untuk setiap $x_j \in \mathcal{B}$, lakukan tokenisasi *byte-level* BPE: $T_j = \text{BPE}_{byte}(x_j, V_{BPE})$.
(4)	Bentuk input *full-sentences* tanpa NSP: $X_j = [\langle s \rangle, T_j, \langle /s \rangle]$.
(5)	Ambil posisi *mask* secara dinamis: $M_j \sim q(M \mid X_j)$.
(6)	Bentuk input termasking: $X_{j\setminus M_j} = \text{Mask}(X_j, M_j)$.
(7)	Proses input dengan *encoder* RoBERTa: $H_j = \text{RoBERTa}_\theta(X_{j\setminus M_j})$.
(8)	Ambil representasi token awal: $h_{\langle s \rangle} = H_j[0]$.
(9)	Hitung logit klasifikasi: $z_j = W_c h_{\langle s \rangle} + b_c$.
(10)	Hitung probabilitas kelas: $P(y_j \mid x_j) = \text{softmax}(z_j)$.
(11)	Hitung *loss* klasifikasi: $\mathcal{L}_{cls} = -\sum_{c=1}^{C} y_{j,c} \log P(y_{j,c} \mid x_j)$.
(12)	Pada tahap *pre-training*, hitung *loss dynamic masking*: $\mathcal{L}_{DM} = -\sum_{i \in M_j} \log P_\theta(x_i \mid X_{j\setminus M_j})$.
(13)	Perbarui θ, W_c, dan $b_c$untuk meminimalkan *loss*.
(14)	Tentukan prediksi akhir: $\hat{y} = \arg\max_c P(y = c \mid x_t)$.
Output	prediksi label sentimen $\hat{y}$, probabilitas kelas $P(y \mid x)$, dan representasi teks RoBERTa $h_{\langle s \rangle}$

Pseudocode pada Tabel 2.3 menjelaskan alur pemrosesan teks bahasa Inggris D_{en} menggunakan karakteristik pelatihan RoBERTa. Proses dimulai dengan teks x_j yang ditokenisasi menggunakan *byte-level* BPE, kemudian dibentuk dalam format *full-sentences* tanpa NSP. Pada tahap *pre-training*, posisi *mask* M_j diambil dinamis melalui $M_j \sim q(M \mid X_j)$, sehingga pola *mask* dapat berubah setiap kali data dimasukkan ke model. Representasi kontekstual dari *encoder* RoBERTa diambil melalui token awal $h_{\langle s \rangle}$, lalu diproses oleh *classification head* untuk menghasilkan probabilitas sentimen $P(y \mid x)$ dan menentukan label akhir melalui *argmax* [90].

2.3.4 DeBERTa-v3

DeBERTa-v3 atau *Decoding-enhanced BERT with Disentangled Attention version 3* merupakan *transformer text encoder* yang menghasilkan representasi teks kontekstual melalui mekanisme *disentangled attention*, yaitu pemisahan informasi isi token dan posisi token [88]. Model ini menyempurnakan DeBERTa melalui penggunaan *replaced token detection* sebagai pengganti *masked language modeling* serta penerapan *gradient-disentangled embedding sharing*. Mekanisme tersebut membantu DeBERTa-v3 memahami hubungan antar token secara lebih detail, sehingga relevan untuk klasifikasi teks, *natural language understanding*, dan *aspect-based sentiment classification* [89]. Sebagai model *English-based*, DeBERTa-v3 lebih umum digunakan pada teks berbahasa Inggris karena proses *pre-training* utamanya dibangun dari korpus bahasa Inggris [89]. Struktur arsitektur DeBERTa-v3 yang berbasis *disentangled attention* ditampilkan pada Gambar 2.7.



Gambar 2.7 Arsitektur *Pre-Training* DeBERTa-v3
Sumber: [92]

Arsitektur Gambar 2.7 menunjukkan tiga bentuk hubungan *embedding* antara *generator* dan *discriminator* dalam proses *pre-training*. Bagian (a) menunjukkan

embedding sharing, yaitu *generator* dan *discriminator* memakai *embedding* yang sama sehingga pembaruan parameter terjadi dari gabungan *loss* MLM dan RTD. Bagian (b) menunjukkan *no embedding sharing*, yaitu *generator* dan *discriminator* memakai *embedding* berbeda sehingga pembaruan parameter dilakukan secara terpisah. Bagian (c) menunjukkan mekanisme utama DeBERTa-v3, yaitu *gradient-disentangled embedding sharing*, dengan *embedding discriminator*. Mekanisme *stop-gradient* sg mencegah *loss* RTD memperbarui *embedding generator* secara langsung, sementara E_Δ berfungsi sebagai *residual* yang diperbarui oleh *discriminator* sebagaimana dirumuskan dalam Rumus 2.11 [93].

$$E_D = \mathrm{sg}(E_G) + E_\Delta \tag{2.11}$$

Rumus 2.11 *Gradient-Disentangled Embedding Sharing* DeBERTa-v3
Sumber: [93]

Keterangan:

1) E_D merupakan *embedding token* pada *discriminator* DeBERTa-v3.
2) E_G merupakan *embedding token* pada *generator*.
3) $\mathrm{sg}(\cdot)$ merupakan operator *stop-gradient*.
4) $\mathrm{sg}(E_G)$ merupakan *embedding generator* yang digunakan oleh *discriminator* tanpa menerima pembaruan *gradient* dari *loss discriminator*.
5) E_Δ merupakan *residual embedding* yang diperbarui oleh *loss discriminator*.

Rumus 2.11 menjelaskan mekanisme *Gradient-Disentangled Embedding Sharing* (GDES) pada DeBERTa-v3, yaitu metode pembagian *embedding* antara *generator* dan *discriminator* dengan pemisahan aliran *gradient*. Pada mekanisme ini, *embedding discriminator* E_D dibentuk dari penjumlahan antara *embedding generator* yang diberi operator *stop-gradient* $\mathrm{sg}(E_G)$ dan *residual embedding* E_Δ. Operator *stop-gradient* mencegah *loss discriminator* memperbarui *embedding generator*, sehingga konflik *gradient* antara *generator* dan *discriminator* dapat dikurangi. Dengan demikian, *discriminator* tetap memperoleh manfaat dari informasi semantik *embedding generator*, tetapi pembaruan dari tugas *replaced token detection* diarahkan hanya melalui E_Δ. Setelah *training* selesai, E_Δ ditambahkan ke E_G untuk membentuk *embedding* akhir *discriminator* E_D, sehingga

DeBERTa-v3 dapat mempertahankan manfaat *embedding sharing* tanpa menyebabkan *tug-of-war gradient* antara dua objective *pre-training* [40]. Tahapan kerja DeBERTa-v3 ditunjukkan pada Tabel 2.4 melalui *pseudocode*.

Tabel 2.4 *Pseudocode* DeBERTa-v3
Sumber: [92]

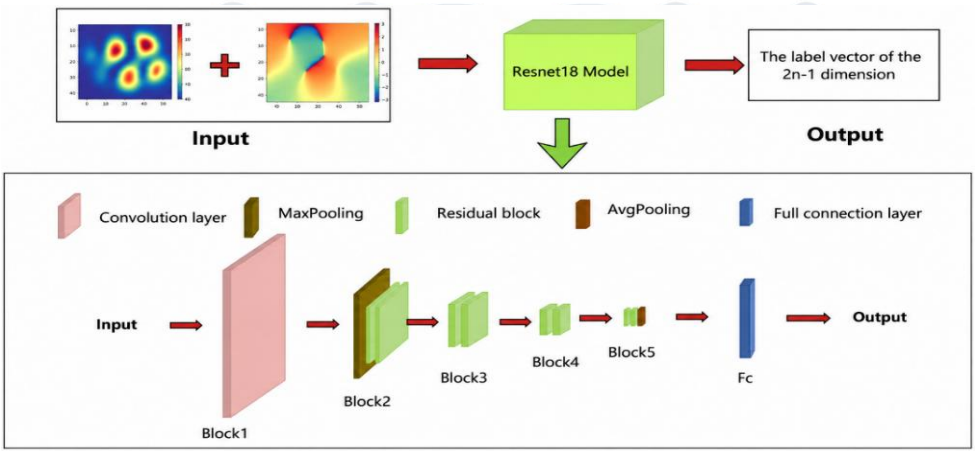
Input	*Dataset* teks Inggris $D_{en} = \{(x_j, y_j)\}_{j=1}^{N}$, *vocabulary/tokenizer* V, parameter *generator* θ_G, parameter *discriminator* θ_D, *embedding generator* E_G, *residual embedding* E_Δ, jumlah *epoch* E, ukuran *mini-batch* B, dan jumlah kelas sentimen C.
(1)	Inisialisasi G_{θ_G}, D_{θ_D}, dan classification head (W_c, b_c).
(2)	Bentuk embedding discriminator dengan GDES: $E_D = \text{sg}(E_G) + E_\Delta$.
(3)	Untuk setiap epoch $e = 1, \dots, E$, ambil mini-batch $\mathcal{B} \subset D_{en}$.
(4)	Untuk setiap $x_j \in \mathcal{B}$, lakukan tokenisasi: $T_j = \text{Tokenize}(x_j, V)$.
(5)	Tentukan posisi token yang dikorupsi: (M_j\subset{1,\ldots,
(6)	Bentuk input termasking: $T_{j\setminus M_j} = \text{Mask}(T_j, M_j)$.
(7)	Generator menghasilkan token pengganti: $\tilde{x}_i \sim P_{\theta_G}(x_i \mid T_{j\setminus M_j}),\ i \in M_j$.
(8)	Bentuk input hasil korupsi: $\tilde{T}_j = \text{Replace}(T_j, M_j, \tilde{x}_i)$.
(9)	Bentuk label RTD: $r_i = \mathbb{1}(\tilde{T}_{j,i} = T_{j,i})$.
(10)	Proses input dengan discriminator DeBERTa-v3: $H_j = D_{\theta_D}(\tilde{T}_j, E_D)$.
(11)	Hitung probabilitas token asli/pengganti: $P_{\theta_D}(r_i \mid \tilde{T}_j) = \sigma(w_r^T h_i + b_r)$.
(12)	Hitung loss RTD: (\mathcal{L}_{RTD}=-\sum_{i=1}^{
(13)	Perbarui θ_D dan E_Δ, sedangkan gradient ke E_G dihentikan melalui $\text{sg}(E_G)$.
(14)	Untuk klasifikasi sentimen, ambil representasi teks: $h_{[\text{CLS}]} = H_j[0]$.
(15)	Hitung logit: $z_j = W_c h_{[\text{CLS}]} + b_c$.
(16)	Hitung probabilitas kelas: $P(y_j \mid x_j) = \text{softmax}(z_j)$.
(17)	Hitung loss klasifikasi: $\mathcal{L}_{cls} = -\sum_{c=1}^{C} y_{j,c} \log P(y_{j,c} \mid x_j)$.
(18)	Perbarui θ_D, W_c, dan b_c untuk meminimalkan $\mathcal{L}_{cls}$.
(19)	Tentukan prediksi akhir: $\hat{y} = \arg\max_c P(y = c \mid x_t)$.
Output	Probabilitas kelas sentimen $P(y \mid x)$, label prediksi $\hat{y}$, dan representasi teks DeBERTa-v3 $h_{[\text{CLS}]}$.

Pseudocode pada Tabel 2.4 menjelaskan alur kerja DeBERTa-v3 dengan menekankan dua karakteristik utama, yaitu *Replaced Token Detection* (RTD) dan *Gradient-Disentangled Embedding Sharing* (GDES). Proses dimulai dari *dataset* teks D_{en} yang ditokenisasi menjadi T_j, lalu sebagian posisi token M_j dikorupsi untuk membentuk input $T_{j\setminus M_j}$. *Discriminator* DeBERTa-v3 D_{θ_D} memproses $\tilde{T}_j$ menggunakan *embedding* $E_D = sg(E_G) + E_\Delta$, sehingga *embedding generator* tetap dimanfaatkan tanpa mengalirkan *gradient discriminator* kembali ke E_G, lalu

memprediksi status token melalui $P_{\theta_D}(r_i \mid \tilde{T}_j)$. Setelah *pre-training*, representasi $h_{[CLS]}$ digunakan pada tahap *fine-tuning* untuk menghitung logit z_j, probabilitas sentimen $P(y_j \mid x_j)$, dan label akhir $\hat{y}$ (prediksi kelas sentimen) [92].

2.3.5 ResNet-18

ResNet-18 merupakan *visual encoder* berbasis *Convolutional Neural Network* (CNN) yang menggunakan konsep *residual learning* untuk mengekstraksi fitur visual dari gambar [94]. Model ini dikembangkan untuk mengatasi penurunan performa pada jaringan CNN, terutama melalui penggunaan *shortcut connection* yang meneruskan informasi input ke *output* blok secara langsung . Sebagai keluarga *Residual Network*, ResNet-18 memiliki 18 lapisan utama dengan *basic residual block* sebagai komponen pembentuknya [95]. Karakteristik tersebut membuat ResNet-18 banyak digunakan pada berbagai tugas klasifikasi citra, termasuk citra medis, citra hewan, *product review*, dan *remote sensing* berbasis fitur visual [96]. Struktur arsitektur ResNet-18 seperti pada Gambar 2.8, bertumpu pada susunan lapisan konvolusi dan blok residual yang mendukung representasi visual dipelajari secara bertahap tanpa menghilangkan informasi penting dari lapisan sebelumnya. Melalui mekanisme tersebut, ResNet-18 mampu menghasilkan representasi fitur visual yang stabil dan cocok untuk tugas pengenalan citra.



Gambar 2.8 Arsitektur ResNet-18
Sumber: [97]

Arsitektur ResNet-18 pada Gambar 2.8 menunjukkan alur pemrosesan gambar dari input hingga *output* klasifikasi. Input dari gambar diproses melalui

lapisan konvolusi awal pada *Block1* untuk mengambil ciri dasar seperti tepi, warna, dan pola sederhana. Proses berikutnya melewati *max pooling* dan beberapa *residual block* pada *Block2* sampai *Block5* untuk membentuk representasi visual yang semakin ringkas. Setelah fitur gambar terbentuk, *average pooling* merangkum fitur spasial menjadi vektor fitur, lalu *fully connected layer* mengubah vektor tersebut menjadi *output* kelas [97]. Skema tersebut menunjukkan bahwa ResNet-18 dapat menghasilkan representasi visual sebagai fitur citra melalui mekanisme *residual block* yang dinyatakan pada Rumus 2.12.

$$h_{k+1} = \sigma\left(BN\left(W_{k,2}^{3\times3} * \sigma\left(BN\left(W_{k,1}^{3\times3} * h_k\right)\right)\right) + S_k h_k\right), \quad k = 1, \dots, 8 \qquad (2.12)$$

Rumus 2.12 *Basic Residual Block* pada ResNet-18
Sumber: [97]

Keterangan:

1) h_k merupakan *feature map input* pada *residual block* ke-k.
2) h_{k+1} merupakan *feature map output*.
3) $k = 1, \dots, 8$ menunjukkan indeks *basic residual block* ResNet-18.
4) $W_{k,1}^{3\times3}$ merupakan bobot *convolution layer* pertama ukuran 3×3.
5) $W_{k,2}^{3\times3}$ merupakan bobot *convolution layer* kedua ukuran 3×3.
6) $\mathrm{BN}(\cdot)$ merupakan *Batch Normalization*.
7) $\sigma(\cdot)$ merupakan fungsi aktivasi ReLU.
8) $S_k h_k$ merupakan *shortcut connection*.

Rumus 2.12 menjelaskan mekanisme *basic residual block* pada ResNet-18, yaitu *feature map input* h_k diproses melalui dua *convolution layer* berukuran 3×3 yang masing-masing dikombinasikan dengan BN dan aktivasi ReLU. Hasil transformasi tersebut kemudian dijumlahkan dengan *shortcut connection* $S_k h_k$, sehingga model mempelajari *residual mapping*, bukan pemetaan langsung. Jika ukuran *feature map* tidak berubah, *shortcut* menggunakan *identity mapping*, sedangkan jika terjadi perubahan dimensi, *shortcut* menggunakan *projection* atau *downsampling* agar dapat dijumlahkan dengan hasil residual. Setelah proses penjumlahan, aktivasi ReLU menghasilkan *feature map output* h_{k+1}, sehingga informasi dari layer awal tetap dapat diteruskan ke layer berikutnya melalui jalur

shortcut [98]. Tahapan kerja ResNet-18 ditunjukkan pada Tabel 2.5 melalui *pseudocode* model.

Tabel 2.5 *Pseudocode* ResNet-18
Sumber: [98]

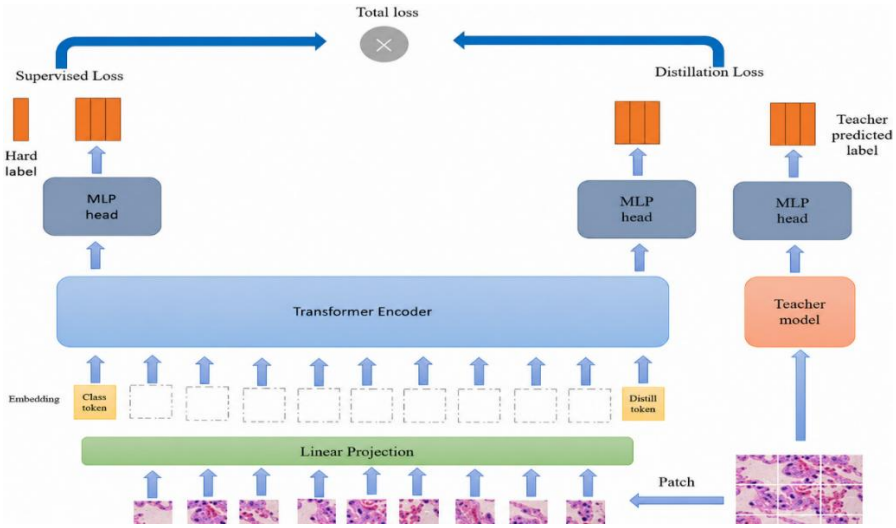
Input	*Dataset* gambar $D_{img} = \{(I_j, y_j)\}_{j=1}^{N}$, ukuran gambar $H \times W$, parameter θ_R, jumlah epoch E, ukuran mini-batch B, dan jumlah kelas sentimen C.
(1)	Inisialisasi ResNet-18 dengan konfigurasi residual block $[2, 2, 2, 2]$.
(2)	Untuk setiap epoch $e = 1, \dots, E$, ambil mini-batch $\mathcal{B} \subset D_{img}$.
(3)	Untuk $I_j \in \mathcal{B}$, bentuk input standar: $\tilde{I}_j = \text{Normalize}(\text{Resize}(I_j, H, W))$.
(4)	Hitung fitur awal: $h_0 = \sigma(BN(W_0^{7\times7} * \tilde{I}_j))$.
(5)	Reduksi spasial awal: $h_1 = \text{MaxPool}(h_0)$.
(6)	Proses $conv2\_x$: $h_2 = \text{Stage}_2(h_1; K_2 = 2)$.
(7)	Proses $conv3\_x$: $h_3 = \text{Stage}_3(h_2; K_3 = 2)$.
(8)	Proses $conv4\_x$: $h_4 = \text{Stage}_4(h_3; K_4 = 2)$.
(9)	Proses $conv5\_x$: $h_5 = \text{Stage}_5(h_4; K_5 = 2)$.
(10)	Untuk setiap block ke-k, terapkan $h_{k+1} = \sigma\left(BN\left(W_{k,2}^{3\times3} * \sigma\left(BN\left(W_{k,1}^{3\times3} * h_k\right)\right)\right) + S_k h_k\right)$.
(11)	Bentuk representasi visual: $v_j = \text{GAP}(h_5)$.
(12)	Hitung logit: $z_j = W_f v_j + b_f$.
(13)	Hitung probabilitas kelas: $P(y_j \mid I_j) = \text{softmax}(z_j)$.
(14)	Hitung loss klasifikasi: $\mathcal{L}_{cls} = -\sum_{c=1}^{C} y_{j,c} \log P(y_{j,c} \mid I_j)$.
(15)	Perbarui parameter θ_R, W_f, dan b_f untuk meminimalkan $\mathcal{L}_{cls}$.
(16)	Tentukan prediksi akhir: $\hat{y} = \arg\max_c P(y = c \mid I_t)$.
Output	Probabilitas kelas $P(y \mid I)$, label prediksi $\hat{y}$, dan representasi visual v_j.

Pseudocode pada Tabel 2.5 menjelaskan alur kerja ResNet-18 secara spesifik berdasarkan susunan delapan *basic residual block* yang terbagi dalam empat *stage*, yaitu $conv2\_x$, $conv3\_x$, $conv4\_x$, dan $conv5\_x$, masing-masing dengan konfigurasi dua *block* $[2, 2, 2, 2]$. Proses dimulai dari input gambar I_j yang distandardisasi menjadi $\tilde{I}_j$, kemudian diproses oleh *convolution* awal $W_0^{7\times7}$, *Batch Normalization*, aktivasi ReLU, dan *MaxPooling* untuk membentuk fitur visual awal. Setelah itu, fitur tersebut melewati empat *residual stage*, dan pada setiap *basic residual block* diterapkan dua *convolution layer* 3×3 serta *shortcut connection* $S_k h_k$ untuk menjaga aliran informasi dari input *block* ke *output block*. *Output* akhir dari $conv5\_x$, yaitu h_5, diringkas menggunakan *Global Average Pooling* menjadi vektor fitur visual v_j. Vektor ini kemudian masuk ke *fully connected layer* untuk

menghasilkan logit z_j, dihitung probabilitas kelasnya melalui softmax, dan diklasifikasikan menjadi label akhir $\hat{y}$ [98].

2.3.6 DeiT

DeiT atau *Data-efficient Image Transformer* merupakan *visual encoder* berbasis *vision transformer* yang memproses citra dengan membaginya menjadi sejumlah *patch* [99]. Model ini dikembangkan untuk mengurangi ketergantungan *vision transformer* terhadap data latih berskala sangat besar dan komputasi yang tinggi melalui mekanisme *knowledge distillation* dengan bantuan *teacher model* [100]. Berbeda dari CNN yang mengekstraksi fitur melalui filter konvolusi, DeiT membangun representasi visual berdasarkan hubungan antar-*patch*, sehingga relevan untuk berbagai tugas klasifikasi citra digital. Dalam prosesnya, citra diubah menjadi *patch embedding* yang diproses oleh *transformer encoder* untuk menangkap hubungan visual, kemudian representasi *class token* digunakan untuk klasifikasi dan *distillation token* membantu transfer pengetahuan dari *teacher model* selama *training* [101]. Struktur arsitektur DeiT seperti pada Gambar 2.9, bertumpu pada pembentukan *patch embedding*, penggunaan *class token* dan *distillation token*, serta pemrosesan fitur visual melalui lapisan *transformer encoder*.



Gambar 2.9 Arsitektur DeiT
Sumber: [102]

Arsitektur DeiT pada Gambar 2.9 menunjukkan alur pemrosesan citra dari *patch* gambar sampai prediksi kelas. Citra masukan dipotong menjadi beberapa

patch, lalu setiap *patch* diubah menjadi *embedding* melalui *linear projection*. Setelah itu, *class token* dan *distillation token* ditambahkan ke urutan *embedding* sebelum diproses oleh *transformer encoder*. *Output* dari *class token* diarahkan ke MLP (*Multi-Layer Perceptron*) *head* untuk menghitung *supervised loss* terhadap label asli, sedangkan *output* dari *distillation token* diarahkan ke MLP *head* lain untuk menghitung *distillation loss* terhadap label dari *teacher model*. Total *loss* diperoleh dari gabungan *supervised loss* dan *distillation loss*, sehingga hasil prediksi tidak hanya belajar dari label data, tetapi juga dari arahan *teacher model* [99]. Mekanisme pembelajaran tersebut dirumuskan melalui fungsi *hard distillation loss* pada Rumus 2.13.

$$\mathcal{L}_{DeiT} = \frac{1}{2}\mathcal{L}_{CE}(\psi(z_{cls}), y) + \frac{1}{2}\mathcal{L}_{CE}(\psi(z_{dist}), y_T), \quad y_T = \arg\max_c z_T(c) \qquad (2.13)$$

Rumus 2.13 *Hard Distillation Loss* pada DeiT
Sumber: [102]

Keterangan:

1) $\mathcal{L}_{\text{DeiT}}$ adalah total *loss* gabungan *supervised loss* dan *distillation loss*.
2) $\mathcal{L}_{CE}$ adalah fungsi *cross-entropy loss*.
3) $\psi(\cdot)$ adalah fungsi *softmax*.
4) z_{cls} adalah logit dari *class head* berdasarkan representasi *class token*.
5) z_{dist} adalah logit dari *distillation head* berdasarkan representasi *distillation token*.
6) y adalah label asli atau *ground-truth label*.
7) y_T adalah *hard label* dari *teacher model*.
8) $z_T(c)$ adalah logit *teacher model* untuk kelas ke-c.
9) $\arg\max_c z_T(c)$ adalah operasi untuk memilih kelas dengan nilai logit *teacher* tertinggi.

Rumus 2.13 menjelaskan fungsi *hard distillation loss* pada DeiT, yaitu *loss* yang menggabungkan *supervised learning* dari label asli dan *knowledge distillation* dari *teacher model*. *Class token* digunakan untuk menghasilkan z_{cls} dan dilatih terhadap label asli y, sedangkan *distillation token* digunakan untuk menghasilkan z_{dist} dan dilatih terhadap *hard label teacher* y_T. *Hard label teacher* diperoleh dari

kelas dengan logit *teacher* tertinggi melalui $y_T = \arg\max_c z_T(c)$. Kedua komponen *loss* dijumlahkan dengan bobot $\frac{1}{2}$, sehingga DeiT dapat belajar dari *ground-truth label* dan pengetahuan *teacher model* sekaligus melalui *distillation token* dalam arsitektur *vision transformer* [101]. Tahapan kerja DeiT ditunjukkan pada Tabel 2.6 melalui *pseudocode* model.

Tabel 2.6 *Pseudocode* DeiT
Sumber: [101]

Input	*Dataset* gambar $D_{img} = \{(I_j, y_j)\}_{j=1}^{N}$, ukuran *patch* $P \times P$, parameter θ_D, *teacher model* T_{θ_T}, *class token* t_{cls}, *distillation token* t_{dist}, *positional embedding* E_{pos}, jumlah *epoch* E, *mini-batch* B, dan jumlah kelas C.
(1)	Inisialisasi *student model* DeiT S_{θ_D}, *teacher model* T_{θ_T}, *class head* H_{cls}, dan *distillation head* H_{dist}.
(2)	Untuk setiap *epoch* $e = 1, \dots, E$, ambil *mini-batch* citra $\mathcal{B} \subset D_{img}$.
(3)	Untuk setiap citra $I_j \in \mathcal{B}$, lakukan standardisasi input: $\tilde{I}_j = \text{Normalize}(\text{Resize}(I_j, H, W))$.
(4)	Bagi citra menjadi *patch*: $\mathcal{P}_j = \{p_1, p_2, \dots, p_n\}$, dengan $n = \frac{HW}{P^2}$.
(5)	Proyeksikan setiap *patch* menjadi *embedding*: $X_p = [E(p_1), E(p_2), \dots, E(p_n)]$.
(6)	Bentuk token input DeiT dengan *class token* dan *distillation token*: $X_0 = [t_{cls}; X_p; t_{dist}] + E_{pos}$.
(7)	Proses token melalui *transformer encoder* DeiT: $Z_L = S_{\theta_D}(X_0)$.
(8)	Ambil representasi *class token* dan *distillation token*: $h_{cls} = Z_L[0]$, $h_{dist} = Z_L[n+1]$.
(9)	Hitung logit dari *class head*: $z_{cls} = H_{cls}(h_{cls})$.
(10)	Hitung logit dari *distillation head*: $z_{dist} = H_{dist}(h_{dist})$.
(11)	*Teacher model* menghasilkan logit *teacher*: $z_T = T_{\theta_T}(\tilde{I}_j)$.
(12)	Tentukan *hard label teacher*: $y_T = \arg\max_c z_T(c)$.
(13)	Hitung *supervised loss* dari *class token*: $\mathcal{L}_{sup} = \mathcal{L}_{CE}(\psi(z_{cls}), y_j)$.
(14)	Hitung *distillation loss* dari *distillation token*: $\mathcal{L}_{dist} = \mathcal{L}_{CE}(\psi(z_{dist}), y_T)$.
(15)	Hitung total *loss* DeiT: $\mathcal{L}_{DeiT} = \frac{1}{2}\mathcal{L}_{sup} + \frac{1}{2}\mathcal{L}_{dist}$.
(16)	Perbarui parameter *student* DeiT θ_D, *class head* H_{cls}, dan *distillation head* H_{dist}untuk meminimalkan $\mathcal{L}_{DeiT}$.
(17)	Pada tahap inferensi, gabungkan *output class head* dan *distillation head*: $z_{final} = \frac{1}{2}(z_{cls} + z_{dist})$.
(18)	Hitung probabilitas kelas akhir: $P(y \mid I_j) = \psi(z_{final})$.
(19)	Tentukan label prediksi akhir: $\hat{y} = \arg\max_c P(y = c \mid I_j)$.
Output	Probabilitas kelas $P(y \mid I)$, label prediksi $\hat{y}$, dan representasi visual v_j.

Pseudocode pada Tabel 2.6 menjelaskan alur kerja DeiT dengan menekankan komponen khas berupa *distillation token* dan *teacher-student distillation*, dimulai

dari citra input I_j yang distandardisasi menjadi $\tilde{I}_j$, dibagi menjadi *patch* $P_j = \{p_1, p_2, \dots, p_n\}$, lalu diproyeksikan menjadi *patch embedding* X_p. Berbeda dari *vision transformer* standar, DeiT membentuk input $X_0 = [t_{cls}; X_p; t_{dist}] + E_{pos}$, dengan t_{cls} untuk mempelajari label asli dan t_{dist} untuk mempelajari prediksi *teacher model*, kemudian menghasilkan z_{cls} dari *class head* dan z_{dist} dari *distillation head*. *Teacher model* T_{θ_T} menghasilkan logit z_T dan *hard label teacher* $y_T = \arg\max_c z_T(c)$, lalu total *loss* dihitung dari kombinasi L_{sup} dan L_{dist}. Pada tahap inferensi, *output* akhir diperoleh melalui $z_{final} = \frac{1}{2}(z_{cls} + z_{dist})$ dan label $\hat{y} = \arg\max_c P(y = c \mid I_j)$ [101].

2.4 *Tools* Penelitian

2.4.1 Python

Python adalah bahasa pemrograman yang banyak digunakan untuk pengembangan aplikasi, analisis data, komputasi ilmiah, *machine learning*, dan *artificial intelligence* [103]. Python memiliki sintaksis yang sederhana dan mudah dibaca, sehingga memudahkan proses penulisan, pengujian, dan pengembangan program. Bahasa pemrograman ini bersifat *open-source* serta dapat dijalankan pada berbagai sistem operasi, seperti Windows, Linux, dan macOS [104]. Dalam penelitian berbasis data, Python didukung oleh berbagai *library* seperti *NumPy*, *Pandas*, *Matplotlib*, *Scikit-learn*, *TensorFlow*, dan *PyTorch* yang membantu proses pengolahan data, visualisasi, serta pemodelan. Dengan dukungan komunitas yang luas dan dokumentasi yang lengkap, Python menjadi salah satu *tools* yang relevan untuk implementasi analisis data dan pembangunan model dalam penelitian ini.

2.4.2 Google Colab

Google Colab adalah *platform notebook* interaktif berbasis *cloud* yang digunakan untuk menjalankan kode Python melalui *browser* tanpa memerlukan instalasi perangkat lunak secara lokal [105]. *Platform* ini mendukung kebutuhan analisis data, *machine learning*, dan *deep learning* karena menyediakan *runtime environment* dengan pilihan komputasi seperti CPU, GPU, dan TPU. Dukungan

komputasi tersebut membantu proses pengolahan data dan pelatihan model yang kompleks berjalan lebih cepat dibandingkan eksekusi menggunakan Python lokal dengan spesifikasi perangkat terbatas [105]. Selain itu, integrasinya dengan Google Drive membuat *notebook*, *dataset*, dan hasil eksperimen dapat disimpan secara otomatis di *cloud*, dapat diakses kapan saja di berbagai perangkat, serta dapat dibagikan dengan lebih mudah kepada kolaborator lain. Namun, penggunaan sumber daya komputasi pada Google Colab memiliki keterbatasan *compute unit*, sehingga akses GPU atau TPU secara lebih stabil umumnya memerlukan langganan berbayar [106].

2.4.3 Streamlit

Streamlit merupakan *framework* berbasis Python yang digunakan untuk membangun aplikasi web interaktif, khususnya untuk menampilkan hasil pengolahan data, *dashboard*, visualisasi, dan keluaran model [107]. *Framework* ini membuat skrip Python dapat disajikan dalam bentuk antarmuka web tanpa perlu membangun komponen *front-end* secara manual menggunakan HTML, CSS, atau JavaScript [108]. Streamlit menyediakan berbagai elemen seperti tombol, *slider*, tabel, grafik, input *file*, dan komponen visual lain yang dapat langsung dihubungkan dengan proses analisis data atau model *machine learning*. Streamlit dapat dijalankan secara lokal melalui perintah "*streamlit run*", kemudian aplikasi dapat dipublikasikan melalui Streamlit *Community Cloud* dengan menghubungkan *repository* GitHub, memilih *branch* dan *file* utama aplikasi, lalu menjalankan proses *deployment* [109]. Dalam penelitian ini, Streamlit digunakan sebagai *tools* untuk membuat *deployment* web sederhana agar hasil model dan proses prediksi dapat ditampilkan dalam bentuk antarmuka yang lebih menarik.

2.4.4 Visual Studio Code

Visual Studio Code (VSCode) merupakan *editor* kode *open-source* dari Microsoft yang digunakan untuk penulisan, pengeditan, dan pengelolaan program secara terstruktur [110]. VSCode mendukung berbagai bahasa pemrograman serta dilengkapi fitur seperti terminal bawaan, *auto-complete*, *syntax highlighting*, *debugging*, dan integrasi Git dalam satu *workspace*. Dalam konteks *deployment*

pada penelitian ini, VSCode digunakan untuk menyusun struktur proyek, mengelola *file*, menjalankan perintah terminal, dan menguji aplikasi sebelum dipublikasikan. VSCode juga mendukung pengembangan aplikasi Streamlit karena *file* Python dapat dijalankan dengan perintah "*streamlit run"* untuk pengujian lokal sebelum *deployment*. Selain itu, dukungan *extension* dan integrasi GitHub memudahkan pengelolaan kode, pencatatan perubahan, serta penghubungan proyek ke *repository* untuk publikasi aplikasi [111].