BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian terdahulu memiliki peran penting dalam memberikan landasan, arah, dan pemahaman yang lebih mendalam terhadap topik yang diteliti. Melalui telaah terhadap penelitian sebelumnya, peneliti dapat mengetahui perkembangan teori yang sudah ada serta menemukan kesenjangan pengetahuan yang masih perlu dikaji lebih lanjut. Kajian ini juga membantu peneliti dalam menilai metode dan hasil dari penelitian lain, sehingga penelitian yang dilakukan dapat memiliki hubungan yang jelas dengan penelitian sebelumnya serta memberikan kontribusi yang lebih relevan. Tabel 2.1 memuat daftar penelitian terdahulu yang dijadikan acuan dalam penelitian ini.

Tabel 2.1 Penelitian Terdahulu

Metode	***Dataset***	**Kelebihan**	**Kekurangan**	**Hasil**
Perbandingan BERT (*fine-tuned*) dan Naïve Bayes, BERT dilatih menggunakan Adam *optimizer* (*learning rate* 1e-5) [25].	*Dataset* berisi total 44.800 tweet bertopik COVID-19 yang terbagi menjadi 41.000 data latih dan 3.800 data uji dengan label sentimen mulai dari sangat negatif hingga sangat positif.	Model BERT unggul dalam menangkap nuansa emosi kompleks serta konteks kalimat dibandingkan metode Naive Bayes.	Model Naive Bayes kurang efektif karena performanya rendah akibat ketidakmampuan memahami hubungan antar kata dalam kalimat, sedangkan BERT masih butuh data lebih besar untuk optimalisasi.	BERT mencapai akurasi 94% dengan F1-score tinggi pada semua kelas, sedangkan *Naïve Bayes* hanya mencapai akurasi sekitar 70.2%.
Perbandingan performa BERT-LSTM dan BERT-FF dengan algoritma tradisional (SVM, Logistic Regression, Naïve Bayes, XGBoost, dll.) [26].	*Dataset* kecil berisi 1.009 tweet yang dikumpulkan manual terkait gempa Turki Februari 2023, dibagi menjadi data latih (70%), evaluasi (15%), dan uji (15%) dengan label biner bencana dan non-bencana.	Model BERT-LSTM dan BERT-FF sangat efektif karena menggabungkan pemahaman konteks mendalam dan pola urutan teks, sehingga mengungguli metode tradisional.	Model tradisional kurang efektif dibanding *Deep Learning*, *dataset* kecil memicu *overfitting* dan membatasi penggunaan BERT-*Large*.	BERT-LSTM mencapai akurasi tertinggi sebesar 85,43%, unggul signifikan dibandingkan model tradisional terbaik SVM yang hanya mencapai 76,80%, serta jauh melampaui Naïve Bayes (72,84%) dan Light-GBM (66,88%).

Metode	*Dataset*	Kelebihan	Kekurangan	Hasil
Perbandingan RoBERTa dengan pendekatan tradisional (Naïve Bayes, Logistic Regression) dan berbasis leksikon (VADER) [27].	Amazon *Fine Food Reviews* yang berisi lebih dari 4.500 ulasan produk.	Model RoBERTa sangat efektif karena teknik berbasis *transformer* yang mampu menangani nuansa kontekstual secara *real-time* untuk mendukung keputusan bisnis strategis.	Model tradisional kurang efektif dibandingkan *deep learning*, di mana Naïve Bayes, VADER, dan Logistic Regression menunjukkan performa di bawah model *transformer*.	RoBERTa mencapai akurasi tertinggi sebesar 87.0%. Hasil ini menunjukkan keunggulan signifikan model berbasis *Transformer* dibandingkan model tradisional seperti Logistic Regression (83.58%), VADER (82.8%), dan Naïve Bayes yang hanya mencapai 80.91%.
BERT dan IndoBERT (indobenchmark/indobert-*base*-p1), *fine-tuned* menggunakan Adam *optimizer* (*learning rate* 2e-5)**,** *batch size* 16, dan 10 *epochs* [28].	Tweet Status COVID-19 berjumlah 1.000 pesan dari media sosial X yang berisi laporan mandiri terkait status terinfeksi virus, terdiri dari 500 data kelas positif (terinfeksi/gejala) dan 500 data kelas negatif.	Model IndoBERT sangat efektif karena mampu menangani kekurangan algoritma sebelumnya dan lebih optimal dalam mendeteksi status laporan mandiri berbahasa Indonesia dibandingkan model BERT standar.	Model BERT kurang efektif, sementara algoritma lama seperti CNN dan LSTM memiliki keterbatasan dalam memahami konteks global teks serta membutuhkan sumber daya data yang besar.	IndoBERT: Akurasi 94%, Spesifisitas 92%, Sensitivitas 96%, lebih baik dari BERT (82%) dan signifikan secara statistik (p=0.0488).
IndoBERT (*indobenchmark/*IndoBERT-*base-p1*)*, AdamW optimizer* (*lr=2e-5*)*, 3 epochs, batch size 16-128, max_len=128* [14].	Ulasan Produk Shopee berjumlah 1.926 data ulasan (periode Januari-Juni) yang dibagi menjadi 1.541 data latih dan 385 data uji dengan pelabelan sentimen positif dan negatif.	Model IndoBERT sangat efektif dengan kemampuan membaca konteks dua arah (bidireksional) yang unggul dalam menangani nuansa bahasa Indonesia.	Konfigurasi *batch size* kecil (16 dan 32) kurang optimal dibandingkan *batch size* besar, serta model masih memiliki keterbatasan dalam menangani *noise* data yang memerlukan optimasi lebih lanjut.	IndoBERT: Akurasi tertinggi 93% (pada *batch* 128), stabil pada 91% untuk *batch* 16-32.
IndoBERT *pretrained model* (*Indo4B*)*, AdamW optimizer* (*lr=1e-5, weight decay=0.01*)*, 5 epochs, batch size=8, max_len=72* [29].	Ulasan Aplikasi BRImo yang bersumber dari Kaggle (Google Play Store) berjumlah total 17.219 data ulasan pengguna.	Model IndoBERT sangat efektif karena dirancang khusus untuk memahami karakteristik bahasa Indonesia dan konteks kalimat.	Naive Bayes dan SVM kurang efektif menangkap konteks bahasa kompleks dibandingkan IndoBERT.	IndoBERT: Akurasi 90%, F1-*Score* = Negatif 0,89 / Netral 0,91 / Positif 0,90.

Metode	*Dataset*	Kelebihan	Kekurangan	Hasil
CNN dengan fitur TF-IDF dan Word2Vec, serta optimisasi hiperparameter menggunakan *Particle Swarm Optimization* (PSO) [15].	Tweet Calon Presiden 2024 yang bersumber dari Twitter berjumlah 37.391 data dengan 5 kata kunci utama terkait kandidat presiden tahun 2024.	Model CNN dengan optimasi PSO sangat efektif karena meningkatkan performa melalui seleksi parameter yang optimal serta penggabungan fitur TF-IDF dan Word2Vec.	Model CNN *baseline*, SVM, dan Naïve Bayes kurang efektif dalam menangani kompleksitas fitur dibandingkan *deep learning* teroptimasi.	Akurasi 78,2%, naik ±10,07% dari *baseline*; PSO meningkatkan akurasi dan F1 dibanding konfigurasi *default.*
ResNet-50 dengan channel attention dan *modified sigmoid cross-entropy loss*, dilatih selama 125 *epoch* dengan *batch size* 32 [17].	AffectNet *Dataset* dengan total sampel 2.000 gambar wajah berlabel emosi yang dibagi menjadi data latih (80%) dan uji (20%).	Model ResNet-50 yang ditingkatkan sangat efektif karena penambahan mekanisme atensi dan fungsi *loss* khusus yang mempertajam deteksi fitur emosi.	Model R-CNN dan BERT kurang efektif dibandingkan ResNet-50 karena menghasilkan nilai akurasi dan *recall* yang lebih rendah dalam mengenali respon emosional visual pengguna.	ResNet-50: F1 0,946, *Recall* 0,938, lebih tinggi dari R-CNN (0,862) dan BERT (0,845). Model dinilai ≥95/100 oleh pakar industri, menunjukkan efektivitas tinggi untuk analisis emosi visual.
YOLOv8 untuk deteksi dan segmentasi, dikombinasikan dengan attention model berbasis ResNet-101 untuk ekstraksi fitur serta *cosine similarity* untuk pengukuran kemiripan [30].	Total 131.774 gambar pakaian yang terdiri dari 99.621 data latih dan 32.153 data validasi, mencakup 13 kategori berbeda	Model Usulan (YOLOv8 + Attention) sangat efektif karena mampu mendeteksi dan mengisolasi objek pakaian dari latar belakang dengan presisi tinggi serta bekerja secara *real-time*.	Model GRNet, DLCBIR, dan BiCBIR kurang efektif; metode tradisional (SIFT, SURF, HOG) kurang presisi, dan segmentasi murni membutuhkan komputasi besar.	Akurasi 94% (Top-3) dan 96% (Top-5) dengan kecepatan *real-time.* Integrasi YOLOv8 meningkatkan segmentasi dan presisi pencarian gambar *fashion.*
Stable Diffusion + Multi-LoRA (*Color, Simple, Full*) untuk *captioning* dan generasi gambar *fashion*. Evaluasi menggunakan ViT *Loss* dan *Fashion Loss* [24].	Terdiri dari 3 set data: 4.000 gambar (*caption* warna), 3.000 gambar (*caption* simpel), dan 1.500 gambar (*caption* detail), ditambah 100 gambar validasi.	Model "LLM-Full LoRA" sangat efektif (*loss* terendah) karena kombinasi Stable Diffusion dan multi-LoRA sukses menghasilkan gambar realistis sesuai domain *fashion* (gaya, bahan, pola) dibanding model dasar.	Model *Base* LLM & Stable Diffusion kurang efektif (*loss* tertinggi) karena gagal menangkap detail pakaian, serta metode pelatihan gabungan (*end-to-end*) terkendala masalah teknis gradien.	Full LoRA: ViT *Loss* 0.1053, *Fashion Loss* 0.1913, error turun ±30% dari *baseline*; menghasilkan gambar pakaian lebih realistis dan sesuai deskripsi teks.

Metode	*Dataset*	Kelebihan	Kekurangan	Hasil
Framework multimodal dengan *GPT-3.5-Turbo* untuk *pseudo image,* Stable Diffusion untuk gambar sintetis, dan *CLIP* (ViT) untuk ekstraksi fitur; fusi melalui *self-adaptive adapter* (*Siamese*) [31].	*Dataset* terdiri dari 12.061 postingan *multimodal* (9.649 data latih**,** 1.206 data validasi, dan 1.206 data uji), mencakup teks, gambar.	Model Usulan efektif karena data sintetis dari Stable Diffusion memperkaya *dataset*, menjembatani kesenjangan modalitas, dan meningkatkan kinerja sentimen.	Model *Unimodal* (ResNet, ViT, BERT) dan *Multimodal* standar (BLIP-2) kurang efektif dibandingkan model berbasis data sintetis.	Akurasi 76,64%, F1 75,07%, lebih tinggi dari CLMLF (74,02%) dan DeBERTa (71,50%); metode efektif mengurangi *modality gap*.
BERT untuk teks dan ResNet-50 untuk citra, dengan metode fusi CMAC, HSTEC, OTE, NativeCat, dan NativeCombine [19].	Total 4.512 data pasangan gambar-teks yang terdiri dari 3.200 data latih, 800 validasi, dan 512 data uji dengan label emosi positif, negatif, dan netral.	Model OTEModel sangat efektif karena menggabungkan fitur teks BERT dan fitur visual ResNet melalui mekanisme attention, sehingga mampu memanfaatkan informasi lintas modalitas secara komplementer.	Model *unimodal* dan fusi sederhana kurang efektif karena tidak mampu menangkap konteks lintas modalitas secara optimal, serta masih memiliki keterbatasan pada emosi ekstrem dan latar belakang gambar yang kompleks.	BERT-*only*: 70.5% Acc. ResNet-*only*: 65.75% Acc. OTEModel *fusion*: 74.5% Acc, F1 = 73% (*best*).
Hybrid BERT, RoBERTa, dan FNet untuk analisis teks, dipadukan dengan CNN untuk analisis citra, serta Stable Diffusion v2-*Base* untuk generasi citra sintetis [20].	Total 5.227 data yang terdiri dari 3.529 ulasan teks saja dan 1.698 ulasan bergambar. Kekurangan data visual pada ulasan teks diatasi dengan gambar sintetis yang dibuat menggunakan *Stable Diffusion v2-base*.	Model Hybrid CNN-BERT unggul karena integrasi teks dan visual, sementara BERT (*text-only*) menegaskan dominasi informasi tekstual.	Model CNN (*image-only*) kurang efektif karena gambar sintetis dari Stable Diffusion standar tidak cukup merepresentasikan kompleksitas visual dunia nyata untuk klasifikasi sentimen mandiri.	*Text-only*: BERT 90.97%, RoBERTa 90.33%, FNet 89.69%. *Image-only*: CNN 54.25%. *Multimodal*: CNN-BERT 90.60%, CNN-RoBERTa 89.40%, CNN-FNet 86.20%.
BERT, ALBERT, XLNet untuk teks; ViT, DeiT untuk citra; fusi dengan *concatenation*. dengan citra sintetis dihasilkan oleh Stable Diffusion v2-*Base* [18].	Total 5.227 data yang terdiri dari 3.529 ulasan teks saja dan 1.698 ulasan bergambar. Kekurangan data visual pada ulasan teks diatasi dengan gambar sintetis yang dibuat menggunakan *Stable Diffusion v2-base*.	Model *Multimodal* (BERT–DeiT) efektif menggabungkan teks dan gambar; BERT unggul untuk teks.	Model *Image-only* (DeiT & ViT) kurang efektif karena kualitas gambar sintetis belum sepenuhnya merepresentasikan ulasan nyata. Model XLNet kurang efisien akibat komputasi lambat, sementara ALBERT menunjukkan performa terendah pada teks.	*Text-only*: BERT 92.16%, XLNet 91.39%, ALBERT 90.39%. *Image-only*: DeiT 69.60%, ViT 67.30%. *Multimodal* (BERT+DeiT): 93.49% (*best*).

Berdasarkan Tabel 2.1 penelitian terdahulu menunjukkan perkembangan signifikan dalam efektivitas analisis sentimen, khususnya melalui penerapan model berbasis *Deep Learning* dan *Transformer* yang secara konsisten mengungguli pendekatan tradisional. Pada studi klasifikasi sentimen terkait COVID-19 yang menggunakan 44.800 data *tweet*, model BERT yang di-*fine-tune* terbukti sangat dominan dengan mencapai akurasi 94% dan F1-score tinggi di seluruh kelas, jauh melampaui algoritma *Naïve Bayes* yang hanya mencatatkan akurasi sekitar 70,2% [25]. Keunggulan ini juga terkonfirmasi dalam konteks *dataset* terbatas untuk respons bencana alam, di mana model hibrida BERT-LSTM mencatatkan akurasi tertinggi sebesar 85,43%. Hasil ini secara signifikan mengungguli model tradisional terbaik, SVM, yang hanya mencapai 76,80%, serta memperlihatkan kesenjangan performa yang jauh dibandingkan *Naïve Bayes* dan Light-GBM [26].

Selain itu, penelitian mengeksplorasi penerapan model *Deep Learning* berbasis *Transformer* pada analisis sentimen ulasan pelanggan *e-commerce* menggunakan *dataset* Amazon *Fine Food Reviews*. Hasil eksperimen memperlihatkan bahwa model RoBERTa mencatatkan kinerja paling akurat dengan akurasi mencapai 87,0%. Capaian ini menunjukkan keunggulan signifikan arsitektur *Transformer* dibandingkan pendekatan tradisional maupun berbasis leksikon, di mana *Logistic Regression* memperoleh akurasi 83,58%, VADER sebesar 82,8%, dan *Naïve Bayes* tertinggal di angka 80,91% [27]. Temuan ini mengindikasikan bahwa metode berbasis *Transformer* lebih efektif dalam menangkap nuansa kontekstual ulasan pelanggan dibandingkan model-model klasik.

Penelitian membandingkan performa antara BERT dan IndoBERT dalam klasifikasi teks laporan mandiri COVID-19 pada *media* sosial [28]. IndoBERT menunjukkan peningkatan signifikan dengan akurasi 94% dan sensitivitas 96%, berkat korpus pelatihan yang disesuaikan dengan bahasa Indonesia [28]. Sementara itu, penelitian menerapkan IndoBERT pada ulasan produk Shopee dan memperoleh akurasi hingga 93%, mengindikasikan kemampuan model tersebut dalam memahami konteks ulasan *e-commerce* lokal [14]. Penelitian juga menggunakan

IndoBERT untuk ulasan aplikasi BRImo dan mencapai akurasi 90% dengan F1-*score* yang seimbang antar kelas sentimen [29].

Jalur *unimodal* menempatkan keluarga CNN sebagai *baseline* yang layak sebelum pembahasan ResNet. Studi CNN untuk sentimen berbasis gambar menunjukkan kinerja validasi pada kisaran menengah dan sensitif terhadap pilihan arsitektur serta konfigurasi pelatihan, sehingga berguna sebagai tolok ukur awal sebelum memakai model yang lebih dalam [15]. Di sisi lain, pada *domain* teks, varian CNN yang dioptimasi *Particle Swarm Optimization* (PSO) melaporkan kenaikan akurasi hingga 78,2% dibanding *baseline* melalui rekayasa fitur (TF-IDF/Word2Vec) dan penyetelan hiperparameter, menguatkan bahwa CNN tetap kompetitif ketika didukung skema optimisasi yang tepat. Temuan-temuan ini memperkuat posisi CNN sebagai pijakan evaluasi pada ekosistem *e-commerce* dengan YOLOv8 untuk penyaringan gambar dan SDXL-LoRA untuk pemenuhan modalitas sebelum beralih ke arsitektur yang lebih kompleks dan ke skema *multimodal*.

Sisi *unimodal* visual ditunjukkan melalui penelitian yang mengembangkan model ResNet-50 dengan mekanisme *channel attention* untuk analisis emosi visual pada desain iklan digital [17]. Pendekatan ini memungkinkan model menyoroti bagian gambar yang paling berpengaruh terhadap pengenalan emosi, sehingga meningkatkan ketepatan dalam memahami ekspresi visual yang kompleks. Hasil pengujian menunjukkan bahwa model mencapai F1-*score* 0,946 dan *recall* 0,938, melampaui kinerja R-CNN (0,862) dan BERT (0,845) [17]. Hal ini mencerminkan keunggulan arsitektur *deep convolutional neural network* dalam menangkap fitur emosional yang tidak dapat ditangkap oleh model berbasis teks. Selain menunjukkan stabilitas pelatihan hingga 125 *epochs*, model juga memperoleh nilai evaluasi ahli ≥95/100 [17].

Dalam ranah *computer vision* dan *fashion*, penelitian memperkenalkan algoritma YOLOv8 (*You Only Look Once*) yang dikombinasikan dengan *attention* model berbasis ResNet-101 untuk sistem pencarian gambar pakaian [30]. Model tersebut mencapai akurasi 94% pada Top-3 dan 96% pada Top-5, dengan kecepatan inferensi *real-time* menggunakan GPU A100 [30]. Kombinasi ini meningkatkan

ketepatan *segment*asi objek pakaian dan memperkuat efisiensi pencarian gambar pada *platform e-commerce fashion*. Selanjutnya, penelitian mengusulkan pendekatan Multi-LoRA *fine-tuning* pada model Stable Diffusion untuk meningkatkan representasi digital pakaian [24]. Model ini menggunakan tiga varian LoRA (*Color*, *Simple*, dan *Full*) yang dilatih secara terpisah dan digabungkan selama inferensi, menghasilkan penurunan *Fashion Loss* hingga 30% dibanding *baseline* [24]. Pendekatan ini membuktikan bahwa *low-rank adaptation* efektif dalam menghasilkan gambar pakaian yang lebih realistis dan semantik [24].

Penelitian mengembangkan *framework multimodal* yang menggabungkan LLM (GPT-3.5-Turbo), Stable Dif*fusion*, dan CLIP (ViT) untuk mengatasi kesenjangan antar-modalitas (*modality gap*) pada unggahan *fashion* di media sosial [31]. Sistem ini menghasilkan *pseudo-image* dari teks ulasan, kemudian melakukan fusi adaptif menggunakan *self-adaptive* adapter berbasis *Siamese network* guna menyesuaikan bobot representasi teks dan gambar secara dinamis. Proses *fine-tuning* dilakukan dengan metode LoRA pada GPU RTX-4090 untuk efisiensi parameter dan peningkatan kualitas fusi modalitas. Hasil pengujian menunjukkan akurasi sebesar 76,64% dan F1-*score* 75,07%, menjadi bukti bahwa pendekatan berbasis *pseudo-data fusion* efektif meningkatkan konsistensi semantik antara informasi visual dan tekstual pada data *multimodal fashion* [31].

Penelitian menggabungkan BERT dan ResNet-50 dalam satu kerangka *multimodal* dengan lima metode fusi berbeda [19]. Model terbaik (OTE) mencapai akurasi 74,5% dan F1-*score* 73%, menunjukkan keunggulan pendekatan *multimodal* dibandingkan model *unimodal* [19]. Penelitian menggunakan kombinasi BERT, RoBERTa, dan FNet untuk teks serta CNN untuk gambar yang dihasilkan melalui Stable Diffusion v2-*Base* [20]. Model *multimodal* ini mencapai akurasi 90,6%, memperlihatkan kontribusi data sintetis terhadap peningkatan kinerja visual [20]. Di sisi lain, penelitian menggabungkan BERT dan DeiT (Vision *Transformer*) melalui fusi *concatenation*, dengan hasil akurasi tertinggi 93,49%, mengonfirmasi efektivitas *transformer fusion* dalam meningkatkan hasil analisis *multimodal* [18].

Hasil-hasil penelitian pada Tabel 2.1 memperlihatkan evolusi dari pendekatan *unimodal* menuju *multimodal* dengan integrasi teknologi *generative* AI. Model berbasis IndoBERT dan ResNet terbukti efektif untuk masing-masing modalitas, sementara kombinasi metode seperti YOLOv8, Stable Dif*fusion*, dan LoRA membuka peluang bagi sistem *multimodal* sentiment analysis yang lebih akurat, efisien, serta kontekstual. Penelitian ini memperluas arah tersebut dengan mengintegrasikan IndoBERT untuk teks, ResNet-18 untuk gambar, YOLOv8 untuk filtrasi otomatis gambar, dan Stable Diffusion XL (SDXL) dengan LoRA *domain fashion*, guna menghasilkan model *multimodal* yang optimal untuk konteks ulasan *e-commerce* di Indonesia.

2.2 Tinjauan Teori

2.2.1 *E-commerce*

Electronic commerce (*e-commerce*) merupakan ekosistem transaksi digital yang memfasilitasi pertukaran barang atau jasa melalui jaringan internet, dengan *platform* daring sebagai penghubung utama antara penjual dan pembeli [32]. Aktivitasnya mencakup pencarian informasi produk, evaluasi alternatif, pemesanan, pembayaran, hingga pemenuhan pesanan yang didukung sistem informasi seperti manajemen katalog, pengelolaan pesanan, serta layanan pelanggan pada website atau aplikasi [33]. Keberadaan *platform* ini berperan sebagai etalase sekaligus infrastruktur transaksi, sehingga proses jual beli dapat berlangsung lebih cepat, terdokumentasi, dan menjangkau konsumen lintas wilayah dibandingkan pola perdagangan [34]. Dalam konteks pasar digital, *e-commerce* juga membuka peluang bagi usaha mikro, kecil, dan menengah untuk memperluas kanal pemasaran, membangun kepercayaan melalui ulasan, serta mengoptimalkan proses operasional berbasis data.

Model transaksi *e-commerce* dapat dipetakan berdasarkan relasi pelaku, seperti *Business-to-Consumer* (B2C), *Business-to-Business* (B2B), *Consumer-to-Consumer* (C2C), dan *Consumer-to-Business* (C2B) [35]. Pada praktiknya, B2C cenderung dominan pada *marketplace* karena menyediakan kemudahan pencarian produk, promosi, serta mekanisme pembayaran dan pengiriman yang terintegrasi.

Akselerasi adopsi *e-commerce* menguat saat *Coronavirus Disease* 2019 (COVID-19) mendorong pergeseran perilaku konsumsi ke kanal daring dan meningkatkan ketergantungan pada layanan pengantaran[36]. Pertumbuhan ini semakin diperkuat oleh peningkatan akses internet dan perangkat *mobile*, adopsi pembayaran digital seperti *electronic wallet* (e-wallet), serta dukungan logistik termasuk *last-mile delivery* yang mempercepat pemenuhan pesanan dan meningkatkan pengalaman pelanggan[37].

2.2.2 Ulasan Produk

Ulasan produk pada *platform e-commerce* merupakan *user-generated content* (UGC) yang merekam evaluasi pascapembelian dalam bentuk narasi teks, *rating* bintang, dan kadang bukti visual, sehingga calon pembeli memperoleh sinyal kualitas, kesesuaian, dan pengalaman penggunaan yang tidak selalu tercakup pada deskripsi penjual [38]. Dalam kerangka *electronic word-of-mouth* (e-WOM), ulasan menyebarkan informasi berbasis pengalaman yang memperkuat persepsi kredibilitas, menurunkan ketidakpastian, dan membentuk sosial *proof* yang dapat menggeser niat beli serta preferensi merek [39]. Dampak ulasan tidak hanya ditentukan oleh valensi positif atau negatif, tetapi juga oleh kekonkretan atribut yang dibahas, kedalaman alasan, dan indikator kegunaan seperti penilaian *helpfulness*, yang memengaruhi seberapa diagnostik informasi bagi pembaca [40]. Dari sisi penjual dan *platform*, agregasi volume ulasan, distribusi *rating*, serta tema keluhan dan pujian dapat dianalisis untuk memantau reputasi, mengidentifikasi area perbaikan produk atau layanan, dan mengestimasi keterkaitan ulasan dengan performa penjualan serta efektivitas pemasaran digital [41].

2.2.3 *Web Scraping*

Web scraping adalah teknik pengambilan data dari laman web secara terprogram yang mengubah informasi tidak terstruktur menjadi *dataset* terstruktur yang mudah dianalisis dan dijaga konsistensinya antarsumber dan antarwaktu [42]. Dalam riset dan praktik industri teknik ini dipakai ketika data primer tidak tersedia atau terbatas pada antarmuka resmi sehingga ulasan produk harga dan metadata perilaku pengguna tetap dapat dihimpun dalam skala besar. Pendekatan dasarnya

bertumpu pada otomatisasi penjelajahan untuk memuat konten dinamis dan pemrosesan dokumen untuk mengekstrak elemen relevan dari struktur HTML atau DOM sehingga hasil siap masuk ke tahap pembersihan penggabungan dan analitik lanjutan [42]. Implementasi umum memanfaatkan Python dengan Selenium untuk merender konten yang dibangkitkan JavaScript *BeautifulSoup* untuk parsing dan pandas untuk pengelolaan data serta alternatif JavaScript langsung di konsol *browser* untuk membaca dan memanen elemen DOM ketika struktur halaman stabil [43]. Tantangan yang sering muncul mencakup paginasi pemuatan bertahap perbedaan struktur antarlaman deteksi bot dan ketidaklengkapan data sehingga diperlukan pengaturan laju permintaan penjadwalan yang wajar st*rate*gi pengulangan saat gagal muat dan validasi kualitas sebelum data digunakan pada model dan pelaporan [44].

2.2.4 *Sentiment analysis*

Sentiment analysis atau *opinion mining* memanfaatkan pemrosesan bahasa untuk mengekstrak informasi subjektif dari teks dan memetakan kecenderungan sikap menjadi polaritas seperti positif, negatif, atau netral, sehingga ulasan pelanggan, percakapan layanan, dan konten media sosial dapat diringkas menjadi indikator persepsi yang terukur [45]. Analisis ini dapat disusun pada beberapa tingkat granularitas, misalnya level dokumen untuk label keseluruhan, level kalimat untuk opini per pernyataan, atau level aspek untuk menangkap sentimen terhadap atribut tertentu dari produk [46]. Pendekatan pemodelan mencakup metode pembelajaran mesin seperti *Logistic Regression*, *Naïve Bayes*, dan *Support Vector Machine* (SVM), serta *deep learning* seperti *Long Short Term Memory* (LSTM) dan *Convolutional Neural Network* (CNN), sementara keluarga *transformer* banyak digunakan melalui *Bidirectional Encoder Re*presentations *from Transformers* (BERT) yang dilatih sebelumnya lalu diadaptasi untuk klasifikasi sentimen [47][48]. Beberapa langkah *Pre-processing* adalah:

1. Konversi Huruf Kecil (*Lowercasing*)

   Proses ini melakukan konversi seluruh karakter dalam teks ke format huruf kecil. Tujuannya adalah untuk menyeragamkan data, memastikan bahwa

sistem tidak salah menginterpretasikan kata yang identik sebagai dua entitas berbeda akibat perbedaan kapitalisasi. Sebagai contoh, tanpa lowercasing, 'Bagus' dan 'bagus' akan dianggap sebagai dua kata yang terpisah, padahal keduanya merujuk pada makna yang sama.

2. Tokenisasi (*Tokenization*)

   Tokenisasi merupakan tahap *segment*asi teks menjadi unit-unit yang lebih kecil, yang disebut sebagai token. Umumnya, token ini merepresentasikan kata-kata individual. Proses ini esensial agar setiap komponen bahasa dapat dianalisis secara terpisah oleh model. Sebagai ilustrasi, frasa 'Kualitas produknya sangat baik' akan dipecah menjadi empat token, 'Kualitas', 'produknya', 'sangat', dan 'baik'.

3. Pembersihan Tanda Baca dan Karakter Khusus

   Tahap ini berfokus pada eliminasi tanda baca (seperti titik, koma, tanda tanya) dan karakter khusus (seperti simbol @, #, atau emotikon) dari teks. Elemen-elemen ini sering dianggap sebagai '*noise*' atau derau dalam analisis semantik karena umumnya tidak berkontribusi pada makna inti sentimen. Sebagai contoh, ulasan 'Barangnya oke!!) Harganya juga murah... #promo' akan dibersihkan menjadi 'Barangnya oke Harganya juga murah promo'.

4. Penghapusan *Stop Word*

   Proses ini melibatkan penghapusan kata-kata umum yang memiliki frekuensi kemunculan tinggi namun minim kontribusi makna (dikenal sebagai stop words), seperti 'yang', 'dan', 'di', atau 'dari'. Tujuan dari stopword removal adalah untuk mereduksi dimensi data dan memfokuskan analisis hanya pada kata-kata yang kaya akan informasi. Misalnya, kalimat 'Produk ini sangat bagus dan bermanfaat' akan disederhanakan menjadi 'Produk sangat bagus bermanfaat'.

5. *Stemming*

   Proses reduksi kata ke bentuk dasarnya dengan cara menghapus imbuhan (afiks) secara algoritmik, tanpa mengandalkan kamus. Misalnya, kata-kata seperti 'belajar', 'mempelajari', dan 'pembelajaran' dapat dipangkas menjadi satu bentuk dasar yang sama, yakni 'ajar'. Meskipun cepat, metode ini

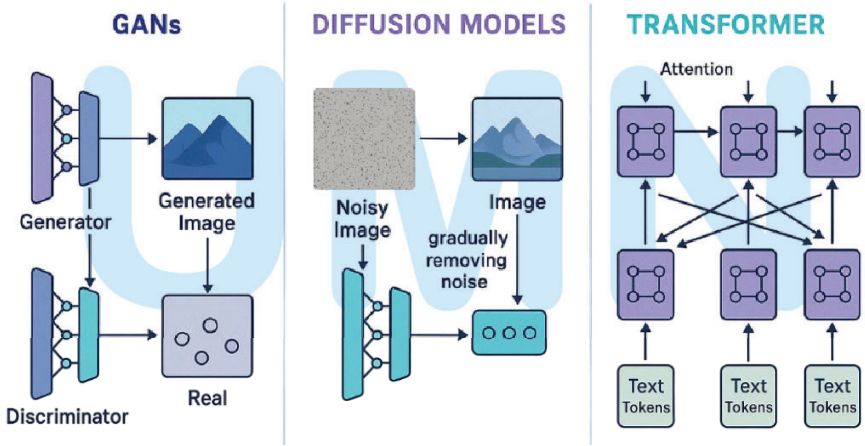
terkadang menghasilkan kata dasar yang tidak valid secara linguistik karena hanya berbasis pada aturan pemotongan.

6. Lematisasi (*Lemmatization*)

   Lematisasi juga bertujuan menemukan bentuk dasar kata, namun dengan pendekatan yang lebih canggih secara linguistik. Berbeda dari stemming, lematisasi menggunakan kamus (l*eksikon*) untuk mengonversi kata ke bentuk dasarnya yang valid (disebut *lemma*). Sebagai contoh, kata 'melihat' akan diubah menjadi 'lihat', dan kata 'terbaik' (yang merupakan bentuk superlatif) akan dikembalikan ke bentuk dasar adjektivanya, 'baik'.

2.2.5 *Image Generative Artificial Intelligence*

Image Generative Artificial Intelligence (*Image Generative AI*) mempelajari cara menghasilkan gambar baru dengan memodelkan distribusi data visual dari kumpulan contoh, sehingga sistem mampu menyintesis visual yang konsisten dengan pola statistik pada data pelatihan [49]. Pendekatan generatif modern umumnya tidak mengandalkan aturan eksplisit, melainkan belajar representasi laten dan mekanisme kondisi seperti prompt teks, label kelas, atau petunjuk lain, agar keluaran dapat diarahkan sesuai kebutuhan [50]. Pemanfaatannya mencakup pembuatan konten kreatif, prototyping desain, pengayaan data untuk pelatihan model, dan pelebaran cakupan data pada *domain* yang sulit dikumpulkan.

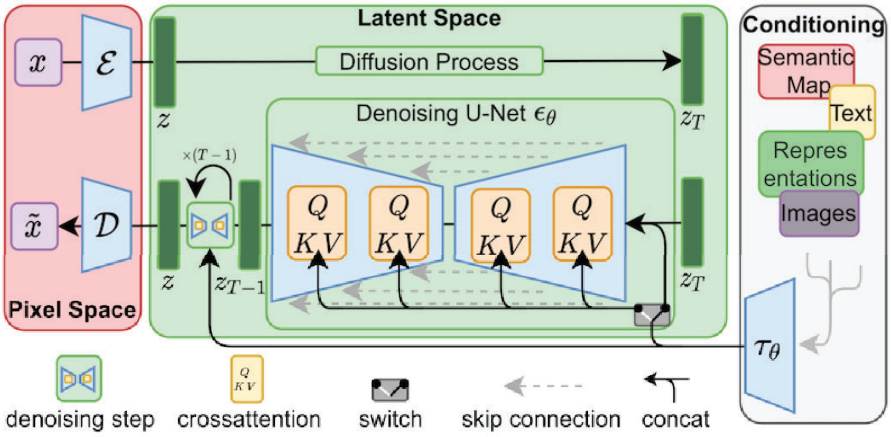


Gambar 2.1 *Generative AI Architecture*
Sumber: [51]

Tiga keluarga arsitektur yang sering dibahas dalam *Image Generative AI* ditunjukkan pada Gambar 2.1**,** yaitu *Diffusion Model*, *Generative Adversarial Network* (GAN), dan model berbasis *Transformer*. *Diffusion Model* seperti *Denoising Diffusion Probabilistic Model* (DDPM) membangkitkan gambar melalui proses *denoising* bertahap dari *noise* menuju citra, yang dikenal stabil dalam pelatihan dan menghasilkan fidelitas tinggi, termasuk varian *Latent Diffusion Model* (LDM) yang menjalankan difusi di ruang laten agar lebih efisien [52]. GAN melatih generator dan *discriminator* secara kompetitif untuk mendorong generator menghasilkan citra yang tidak mudah dibedakan dari data asli, dengan pengembangan praktik seperti StyleGAN2 yang meningkatkan kualitas visual dan mengurangi artefak [53]. Sementara itu, model berbasis *Transformer* memformulasikan generasi gambar sebagai prediksi token secara autoregresif, misalnya melalui *Generative Pretraining from Pixels* (iGPT) dan pendekatan *text-to-image* berbasis token seperti DALL·E, sehingga hubungan antara kondisi teks dan struktur visual dapat dipelajari melalui mekanisme *self-attention* [54].

2.2.6 *Low-Rank Adaptation* (LoRA)

Low-Rank Adaptation adalah pendekatan penyetelan ulang yang hemat parameter untuk menyesuaikan model generatif besar tanpa mengganti seluruh bobot model dasar. Pada Stable Diffusion adaptor LoRA ditempatkan di lapisan *cross-attention* dalam U-Net yang menjadi titik temu antara representasi teks dan gambar [55]. Ukuran berkas LoRA relatif kecil sehingga mudah disimpan dan digabungkan serta cocok bagi pengguna yang ingin mengoleksi banyak gaya atau *domain* pada satu checkpoint dasar. Ketika checkpoint dasar seperti SDXL dipasangkan dengan adaptor LoRA yang dilatih menggunakan data spesifik *domain*, kombinasi tersebut membentuk *domain-adapted generative* AI yang tetap mempertahankan kemampuan umum model dasar sekaligus menyesuaikan karakter visual *domain* target [56].



Gambar 2.2 *LoRA Architecture* 2
Sumber: [57]

Pada Gambar 2.2, LoRA melakukan penyetelan pada lapisan *cross-attention* di U-Net tepatnya pada proyeksi Q, K, dan V. Penambahan adaptor berderajat rendah pada bagian ini menyisipkan pengetahuan gaya atau *domain* baru tanpa mengganti bobot penuh model dasar sehingga perubahan bersifat ringan, terfokus, dan mudah digabungkan dengan *checkpoint* utama [57]. Kombinasi SDXL dan LoRA yang dilatih pada data *domain* tertentu berfungsi sebagai model *domain-adapted generative* AI yang lebih peka terhadap ciri visual yang diinginkan.

2.2.7 *Data Labeling*

Data labeling merupakan proses mengubah data mentah menjadi data beranotasi yang siap digunakan untuk pelatihan dan evaluasi model pembelajaran mesin (*machine learning*), karena anotasi berperan sebagai *ground truth* untuk membandingkan keluaran model dengan kondisi acuan. Pada *semantic segmentation*, anotasi umumnya direpresentasikan sebagai peta label atau *mask* yang menetapkan kelas pada setiap piksel sehingga informasi lokasi dan batas area objek dapat dipelajari secara rinci [58]. Ketepatan dan konsistensi anotasi sangat menentukan kualitas sinyal pembelajaran, sehingga praktik seperti pedoman pelabelan, pengecekan ulang, dan konsistensi antar anotator penting untuk menekan *noise* pada label [59]. Pendekatan *supervised learning* biasanya memberikan kinerja lebih baik ketika *dataset* berlabel tersedia, karena model mempelajari pemetaan langsung dari *input* ke label piksel. Sebaliknya, pendekatan *unsupervised*

learning berusaha membentuk *segment*asi tanpa label manusia, misalnya melalui pengelompokan berbasis kemiripan fitur yang dipelajari, namun hasilnya sering lebih menantang untuk disejajarkan dengan kategori semantik terutama pada kasus kompleks [60].

2.2.8 *Data Splitting*

Data splitting adalah proses mempartisi *dataset* menjadi *training* set untuk pembelajaran parameter, *validation* set untuk pemilihan konfigurasi dan penyetelan hiperparameter selama eksperimen, serta *test* set sebagai evaluasi final yang tetap "steril" sampai model ditetapkan, sehingga pengukuran kinerja lebih merefleksikan kemampuan generalisasi pada data baru [61]. Skema *hold-out* seperti 80/20 sering dipilih karena memberi porsi besar pada data latih sekaligus menyediakan sampel uji yang memadai, tetapi rasio terbaik tidak selalu sama untuk semua studi dan dapat dipengaruhi oleh ukuran korpus, keragaman data, serta kompleksitas model, sehingga rasio alternatif seperti 70/30 atau 90/10 dapat dipertimbangkan bila ada alasan metodologis yang jelas. Pada masalah klasifikasi dengan distribusi kelas yang tidak seimbang, pemisahan terstratifikasi membantu menjaga proporsi label tetap konsisten di setiap subset agar evaluasi tidak bias [62]. Untuk estimasi yang lebih stabil dibanding satu kali pemisahan, *k-fold cross-validation* mengulang proses pelatihan dan validasi dengan membagi data latih menjadi k lipatan dan merata-ratakan metrik di seluruh lipatan, sementara *test* set tetap dipertahankan terpisah untuk penilaian akhir agar mengurangi kebocoran informasi dari proses *tuning* [63].

2.2.9 Metrik Evaluasi

Dalam konteks pembelajaran mesin, metrik evaluasi berperan sebagai indikator untuk menilai performa suatu model pada berbagai tugas, seperti klasifikasi, regresi, dan segmentasi gambar [64]. Penggunaan metrik ini bertujuan untuk mengukur tingkat ketepatan prediksi yang dihasilkan model serta kemampuannya dalam melakukan generalisasi terhadap data baru yang belum pernah dilihat sebelumnya. Adapun beberapa metrik evaluasi yang sering diterapkan meliputi:

1. Akurasi

Akurasi merupakan salah satu indikator evaluasi yang digunakan untuk menilai tingkat ketepatan prediksi model. Metrik ini menunjukkan proporsi prediksi yang sesuai dengan label sebenarnya terhadap keseluruhan prediksi yang dihasilkan, mencakup prediksi benar pada kelas positif maupun negatif. Nilai akurasi yang tinggi mencerminkan kemampuan model dalam melakukan klasifikasi secara umum dengan baik. Perhitungan akurasi pada penelitian ini mengacu pada Rumus (2.1) [65].

$$\text{Akurasi} = \frac{TP+TN}{TP+FP+TN+FN} \qquad (2.1)$$

Rumus 2.1. Akurasi

Keterangan:

a. True Positive (TP): Banyaknya data berlabel positif yang berhasil diprediksi secara tepat oleh model.
b. True Negative (TN): Banyaknya data berlabel negatif yang diprediksi dengan benar oleh model.
c. False Positive (FP): Banyaknya data berlabel negatif yang keliru diprediksi sebagai positif.
d. False Negative (FN): Banyaknya data berlabel positif yang keliru diprediksi sebagai negatif.

2. Presisi

Presisi adalah metrik yang menilai ketepatan model dalam memprediksi kelas positif. Dengan kata lain, presisi menunjukkan proporsi prediksi positif yang benar dari seluruh prediksi positif yang dihasilkan model. Metrik ini menjadi sangat penting ketika kesalahan dalam memprediksi positif palsu (*false positive*) memiliki dampak yang signifikan. Perhitungan presisi dapat dilakukan menggunakan rumus (2.2) [66].

$$\text{presisi} = \frac{TP}{TP+FP} \qquad (2.2)$$

Rumus 2.2. Presisi

3. *Recall*

Recall adalah metrik yang mengukur kemampuan model dalam mendeteksi seluruh contoh positif yang ada pada *dataset*. Dengan kata lain, *recall* menunjukkan proporsi kasus positif yang berhasil diidentifikasi dengan benar oleh model. Metrik ini menjadi krusial ketika kesalahan dalam melewatkan kasus positif (*false negative*) memiliki konsekuensi yang besar, seperti pada deteksi kanker. Perhitungan *recall* dilakukan menggunakan rumus (2.3) [66].

$$Recall = \frac{TP}{TP+FN} \quad (2.3)$$

Rumus 2.3. Recall

4. F1-*Score*

F1-*Score* merupakan metrik yang menggabungkan presisi dan *recall* dalam satu ukuran harmonis. Metrik ini memberikan penilaian yang seimbang antara presisi dan *recall*, sehingga sangat berguna ketika terdapat ketidakseimbangan antara keduanya. F1-*Score* sering digunakan untuk menilai kinerja model secara keseluruhan ketika baik presisi maupun *recall* sama-sama penting. Perhitungan F1-*Score* dilakukan menggunakan rumus (2.4) [67].

$$F1 - Score = 2 \times \frac{presisi \times Recall}{presisi + Recall} \quad (2.4)$$

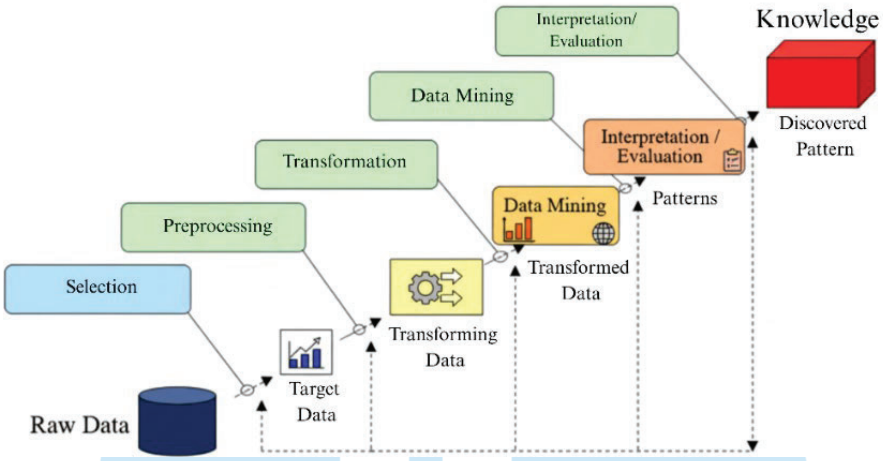
Rumus 2.4. F1-*Score*

2.3 Algoritma dan *Framework*

2.3.1 KDD (*Knowledge Discovery DataBase*)

Knowledge Discovery in Databases atau KDD adalah kerangka kerja sistematis untuk mengekstrak pengetahuan yang bernilai dari data[68]. KDD memadukan praktik statistik, pembelajaran mesin, dan rekayasa data dalam rangkaian langkah yang saling terkait sehingga peneliti tidak sekadar "menjalankan algoritma", melainkan merancang alur yang memastikan data layak pakai, model teruji, dan temuan dapat ditafsirkan. Sifatnya iterative [69]. Ketika evaluasi menunjukkan celah, peneliti kembali ke tahap lebih awal untuk memperbaiki

pemilihan data, pembersihan, atau fitur [70]. Diagram alur KDD dapat dilihat pada Gambar 2.3.



Gambar 2.3 Diagram Proses KDD 3
Sumber: [71]

Gambar 2.3 menunjukkan alur tahapan dalam *framework* KDD yang terdiri dari lima langkah utama, yaitu:

1. *Data Selection*

   Menentukan sumber, cakupan, dan kriteria inklusi-eksklusi data agar representatif terhadap pertanyaan penelitian. Hasilnya adalah himpunan data mentah yang terdokumentasi asal-usul dan batasannya.

2. *Pre-processing* (*Data cleaning*)

   Meningkatkan kualitas data dengan menangani ketidakkonsistenan, nilai hilang, duplikasi, dan *noise*. Pada tahap ini lazim dilakukan normalisasi format, standarisasi satuan, serta penetapan skema pelabelan yang konsisten.

3. *Data Transformation*

   Membentuk ulang data agar lebih informatif dan siap dianalisis. Kegiatan umum meliputi penskalaan, pengodean variabel kategorikal, rekayasa fitur, reduksi dimensi, atau penggabungan beberapa sumber data menjadi satu kerangka yang seragam.

4. *Data Mining*

   Menerapkan teknik komputasional untuk menemukan pola, hubungan, atau struktur tersembunyi. Metode yang digunakan dapat berupa klasifikasi,

klasterisasi, asosiasi, deteksi anomali, atau peramalan, dipilih sesuai tujuan dan karakteristik data.

5. *Interpretation/Evaluation*

   Menafsirkan hasil agar bermakna dan dapat ditindaklanjuti. Tahap ini mencakup evaluasi metrik, pemeriksaan keterjelasan temuan, validasi dengan pengetahuan *domain*, serta perumusan implikasi dan batasan. Jika diperlukan, peneliti kembali ke tahap sebelumnya untuk penyempurnaan.

2.3.2 *Diffusion Model*

Diffusion Model adalah arsitektur generatif berbasis *deep learning* yang bekerja melalui dua tahap utama. Tahap pertama, *forward diffusion*, menambahkan *noise* secara bertahap pada data hingga berubah menjadi *white noise*. Tahap kedua, *reverse denoising*, melatih model untuk menghapus *noise* secara bertahap sehingga data kembali mendekati bentuk aslinya [52]. Proses ini biasanya menggunakan arsitektur U-Net dengan fungsi *loss* berbasis *mean squared error* (MSE). Berbeda dengan GAN yang memerlukan *discriminator*, *Diffusion Model* cenderung lebih stabil saat proses pelatihan.

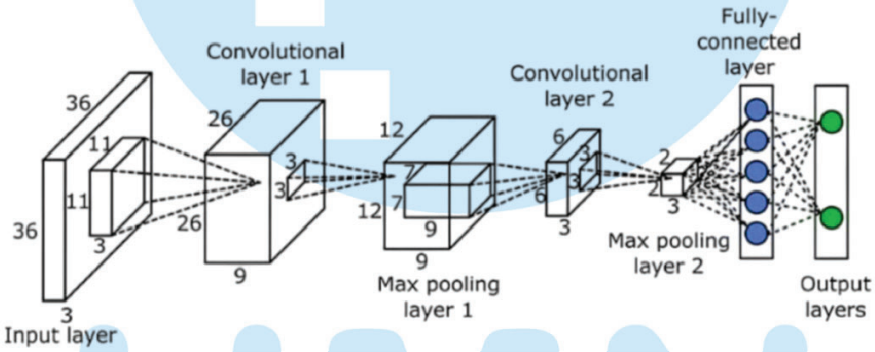
2.3.3 YOLOv8

YOLOv8 merupakan generasi terbaru dari keluarga algoritma *You Only Look Once* yang dirancang sebagai *single-stage object detector* dengan fokus pada keseimbangan antara akurasi dan kecepatan inferensi. Arsitektur YOLOv8 mengadopsi *backbone* ringan dengan modul C2f untuk memperkaya aliran gradien, *neck* berbasis *feature pyramid* untuk fusi fitur multi-skala, serta *decoupled head* yang memisahkan tugas klasifikasi dan regresi *bounding box*, sehingga lebih stabil dalam pelatihan dan lebih peka terhadap objek berukuran kecil [72]. Dalam konteks *fashion* dan pakaian, berbagai penelitian menunjukkan bahwa varian YOLOv8 yang telah diimprovisasi mampu mendeteksi objek pakaian secara lebih andal pada kondisi sulit, misalnya pakaian yang tertutup sebagian (*occlusion*) maupun dengan variasi skala yang lebar, dengan peningkatan akurasi signifikan pada *dataset* seperti ModaNet [73]. Pada sisi inspeksi kualitas, YOLOv8 juga digunakan untuk mendeteksi cacat jahitan garmen, di mana penambahan mekanisme *attention* dan

kepala deteksi khusus objek kecil terbukti meningkatkan *mean Average* Presisi (*mAP*) dan *recall* untuk mendeteksi cacat halus pada pakaian [74]. Karakteristik ini menjadikan YOLOv8 relevan sebagai komponen penyaring awal (*filtering*) gambar ulasan produk *fashion* sebelum digunakan dalam kerangka *multimodal sentiment analysis*.

2.3.4 *Convolutional Neural Network* (CNN)

Convolutional Neural Network adalah arsitektur *deep learning* untuk data berbentuk *grid* seperti gambar yang mengekstraksi fitur secara otomatis melalui konvolusi, *pooling*, aktivasi, dan *fully connected* sehingga belajar hierarkis dari tepi hingga bentuk objek [75]. Pendekatan ini menghapus kebutuhan ekstraksi fitur manual, efisien pada dimensi tinggi, dan umum dipakai untuk klasifikasi serta deteksi, dengan bobot dilatih menggunakan *backpropagation* agar galat minimum [76]. Alur dari gambar masukan menjadi peta fitur lalu keluaran kelas ditunjukkan pada Gambar 2.4.



Gambar 2.4 Diagram *Convolutional Neural Network*
Sumber: [77]

1. *Convolutional Layer*

   Lapisan ini menggunakan filter (kernel) untuk mendeteksi fitur lokal pada gambar, seperti tepi, warna, atau tekstur. Proses konvolusi dilakukan dengan menggeser filter ke seluruh gambar untuk menghasilkan peta fitur. Definisi matematis dari operasi konvolusi dapat dilihat pada rumus (2.5) [78].

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m] \qquad (2.5)$$

Rumus 2.5. Operasi Konvolusi

Keterangan:

a. $f[m]$: Filter atau kernel yang digunakan untuk mengekstrak pola dari data.
b. $g[n - m]$: Data masukan yang diproses oleh filter.
c. n: Posisi saat ini di data masukan di mana hasil konvolusi sedang dihitung.
d. $\sum$: Penjumlahan hasil perkalian elemen-elemen filter dan data masukan pada setiap posisi.
e. $(f * g)[n]$: Hasil akhir konvolusi di posisi nnn, berupa fitur baru yang dihasilkan.

Tabel 2.2 *Pseudocode Convolutional Layer* 2

No.	Tahapan
1	Menyiapkan data masukan *g[n]* dan filter *f[m]* yang akan digunakan untuk konvolusi.
2	Untuk setiap posisi *n* di data masukan, tentukan posisi filter *m* yang sesuai.
3	Hitung hasil perkalian antara filter *f[m]* dan data masukan *g[n−m]* pada setiap posisi *n*.
4	Jumlahkan hasil perkalian $\sum m=-\infty\infty f[m] \cdot g[n-m]$ untuk mendapatkan hasil konvolusi pada posisi n.
5	Simpan hasil konvolusi *(f∗g)[n]* sebagai fitur baru yang dihasilkan pada posisi *n*.
6	Ulangi langkah 3 sampai 5 untuk setiap posisi n dalam data masukan hingga seluruh proses selesai.
7	Akhiri proses setelah seluruh data masukan telah diproses dan fitur konvolusi dihasilkan.

2. *Pooling Layer*

Lapisan pooling bertujuan untuk mengurangi dimensi peta fitur tanpa kehilangan informasi penting. Dua jenis pooling yang umum adalah max pooling (memilih nilai maksimum) dan average pooling (mengambil nilai rata-rata).

3. *Activation Layer* (*ReLU*)

Setelah proses konvolusi atau pooling, fungsi aktivasi seperti ReLU (*Rectified Linear Unit*) diterapkan untuk memperkenalkan non-linearitas. Ini memungkinkan model mempelajari pola-pola yang lebih kompleks dalam data. Fungsi ini mengubah nilai negatif menjadi nol, yang membuat jaringan lebih efisien dalam memproses data. Definisi operasi secara matematis ReLU dapat dilihat pada Rumus (2.6) [78]:

$$f(x) = max(0, x) \qquad (2.6)$$

Rumus 2.6. *Activation Layer* (ReLU)

Keterangan:

a. $f(x)$: Fungsi aktivasi yang akan diterapkan pada nilai *input* x.

b. x: Nilai *input* yang diterima oleh fungsi aktivasi ReLU. Biasanya, ini adalah hasil dari operasi konvolusi atau proses lapisan sebelumnya dalam jaringan.

c. $max(0, x)$: Fungsi ReLU itu sendiri, yang menghasilkan nilai *input* x jika x lebih besar dari 0, dan menghasilkan 0 jika x kurang dari atau sama dengan 0.

Tabel 2.3 *Pseudocode Activation Layer (ReLU)*[3]

No.	Tahapan
1	*Input*: Ambil nilai *input x* dari lapisan sebelumnya (*output* dari konvolusi atau lapisan lain).
2	Periksa nilai *input*: Tentukan apakah *x* lebih besar dari 0 atau tidak.
3	Penerapan ReLU: Jika $x>0$, maka *output* $f(x)=x$. Jika $x\leq 0$, maka *output* $f(x)=0$.
4	*Output*: Simpan hasil *output f(x)* yang telah diproses menggunakan fungsi ReLU.
5	Ulangi langkah 2-4 untuk setiap elemen *x* dalam lapisan jaringan hingga seluruh data diproses.
6	Akhiri: Proses selesai setelah seluruh *input* telah diproses dan hasil aktivasi ReLU dihasilkan.

4. *Fully Connected Layer*

Lapisan ini menghubungkan semua neuron di lapisan sebelumnya untuk menghasilkan *output* akhir, seperti klasifikasi kelas. Fungsi aktivasi softmax sering digunakan pada lapisan ini untuk menghasilkan probabilitas antar kelas.

2.3.5 Resnet-18

Algoritma *Residual Network* (ResNet) merupakan salah satu arsitektur *deep learning* yang dikembangkan untuk mengatasi masalah *vanishing gradient* pada jaringan saraf konvolusional dengan kedalaman tinggi [79]. ResNet-18 adalah versi yang memiliki 18 lapisan, lebih ringan dibandingkan ResNet-50 atau ResNet-101, tetapi tetap mampu melakukan klasifikasi gambar dan ekstraksi fitur secara efektif.

Konsep *residual learning* pada ResNet dijelaskan secara matematis melalui rumus (2.7) [80].

$$H(x) = F(x) + x \qquad (2.7)$$

Rumus 2.7. *Residual Learning*

Keterangan:

1. $H(x)$: *Output* dari blok *residual*,
2. $F(x)$: Transformasi non-linear hasil lapisan konvolusi,
3. x: *Input* asli yang dilewatkan melalui *skip connection*.

Tabel 2.4 Pseudocode Residual *Connection* 4

No.	Tahapan
1	Terima *input* x sebagai masukan blok residual.
2	Hitung $F(x)$ dengan menerapkan transformasi non-linear (mis. rangkaian konvolusi + aktivasi).
3	Jika diperlukan, lakukan penyesuaian dimensi pada x agar sama dengan $F(x)$ (mis. 1×1 *convolution*/*stride*/*padding*).
4	Jumlahkan hasilnya untuk memperoleh *output*: $H(x) = F(x) + x$.
5	Keluarkan $H(x)$ sebagai *output* blok residual ke lapisan berikutnya.

2.3.6 DeiT

Data-Efficient Image Transformer (DeiT) adalah arsitektur turunan dari Vision *Transformer* (ViT) yang dirancang khusus untuk mencapai efisiensi pelatihan pada *dataset* yang terbatas, tanpa memerlukan model *convolutional neural network* (CNN) sebagai pelatih eksternal (*teacher*) [81]. DeiT memperkenalkan mekanisme pelatihan berbasis *knowledge distillation* secara langsung ke dalam arsitektur *transformer*, memungkinkan model belajar dari "*soft target*" tanpa memerlukan CNN sebagai model pembimbing eksplisit. Hal ini menjadikan DeiT lebih praktis dan ringan untuk pelatihan di lingkungan dengan sumber daya terbatas [82]. Secara umum, struktur DeiT mirip dengan ViT, meliput*i embedding patch, positional encoding, transformer encoder,* dan *classification head.* Namun, perbedaan utamanya terletak pada penambahan satu token khusus distilasi yang digunakan untuk menangkap pengetahuan dari model teacher selama pelatihan. DeiT bekerja dalam beberapa tahap utama seperti dijelaskan di bawah ini:

1. *Patch Embedding*

Gambar input dengan ukuran H×W×C dibagi menjadi N *patch*, di mana setiap *patch* berukuran P×P. Setiap *patch* kemudian diratakan dan diproyeksikan ke dalam dimensi D melalui *layer linear*. Proses ini memungkinkan model menangkap representasi lokal dari setiap bagian gambar secara efektif, dan konsep *patch embedding* pada DeiT dijelaskan secara matematis pada Rumus (2.8) [83].

$$x_p = Flatten(\mathrm{x}_{(\mathrm{i})}) \times E \quad (2.8)$$

Rumus 2.8. *Patch Embedding*

Keterangan:

a. $\mathrm{x}_{(\mathrm{i})}$: *patch* ke-i yang telah diratakan menjadi vektor 1 dimensi

b. E: matriks proyeksi linear berdimensi $(P^2C) \times D$

c. x_p: vektor hasil *patch embedding* berdimensi *D*

Tabel 2.5 *Pseudocode Patch Embedding* DeiT [5]

No.	Tahapan
1	Terima gambar *input* X berukuran H × W × C.
2	Tentukan ukuran *patch* P dan hitung jumlah *patch* N = (H/P) × (W/P).
3	Bagi gambar X menjadi N *patch* berukuran P × P × C: $\{x_1, x_2, \ldots, x_N\}$.
4	Untuk setiap *patch* x_i, lakukan Flatten menjadi vektor 1D berukuran ($P^2 \cdot C$): (v_i = Flatten (x_i).
5	Proyeksikan vektor *patch* menggunakan matriks linear E berdimensi (P^2C) × D: $x_{p,i}$ = v_i × E.
6	Kumpulkan seluruh vektor hasil *embedding* menjadi matriks *patch embedding* X_p berukuran N × D.

2. *Positional Encoding* dan Token Tambahan

Untuk mempertahankan informasi posisi di antara *patch*, setiap *patch embedding* diberi tambahan vektor posisi. Pada arsitektur DeiT, rangkaian *patch* diawali oleh dua token khusus, yaitu [CLS] sebagai token klasifikasi seperti pada ViT yang mengagregasi representasi global untuk prediksi kelas, serta [DIST] sebagai token distilasi yang belajar dari keluaran *teacher model* sehingga pengetahuan model guru dapat ditransfer secara efektif ke model

siswa. Konsep *Positional Encoding* pada DeiT dinyatakan secara matematis seperti pada rumus (2.9) [83].

$$z^0 = [x_{cls}; x_{dist}; x_{p1}; x_{p2}; \dots; x_{pN}] + E_{pos} \quad (2.9)$$

Rumus 2.9. *Positional Encoding*

Keterangan:

a. x_{cls}: token klasifikasi.

b. x_{dist}: token distilasi.

c. E_{pos}: positional encoding matrix

d. z^0 : *input* untuk *transformer encoder*

Tabel 2.6 *Pseudocode Positional Encoding* dan Token Tambahan

No.	Tahapan
1	Siapkan *patch embedding* Xp = $[x_{p1}; x_{p2}; \dots; x_{pN}]$ berukuran N ×D.
2	Inisialisasi dua token khusus: x_{cls} *dan* x_{dist} berdimensi D.
3	Gabungkan token: $[x_{cls};\ x_{dist}; x_{p1}; \dots; x_{pN}]$ sehingga panjang urutan menjadi N+2.
4	Siapkan matriks positional encoding E_{pos} berdimensi (N+2)×D.
5	Tambahkan positional encoding untuk memperoleh *input encoder*: $z^0 = [x_{cls};\ x_{dist}; x_{p1}; \dots; x_{pN}] + E_{pos}$

3. *Transformer Encoder*

Komponen ini terdiri dari beberapa *layer* yang masing-masing mencakup multi-head *self-attention* (MHSA) dan feed-forward network (FFN). Definisi matematis *Self-attention* dapat dilihat pada rumus (2.10) [83]:

$$Attention(Q, K, V) = softmax\left(\frac{(Q \times K^t)}{\sqrt{d_k}}\right) \times V \quad (2.10)$$

Rumus 2.10. *Self-attention*

Keterangan:

$Q = XW^Q K = XW^K, V = XW^V$

di mana X adalah *input patch*, dan W^Q, W^K, W^V adalah matriks bobot untuk query, key, dan value.

Tabel 2.7 *Pseudocode Transformer Encoder* (MHSA + FFN)

No.	Tahapan
1	Terima *input encoder* z^0, tentukan jumlah *layer encoder* L.
2	Untuk setiap *layer* $l = 1$ hingga L:lakukan normalisasi $u = LN(z^{l-1})$.
3	Hitung $Q = uW^Q, K = uW^K$, dan $V = uW^V$.

No.	Tahapan
4	Hitung *self-attention*: $A = softmax\left(\frac{QK^T}{\sqrt{dk}}\right)V$ (dalam praktik dilakukan per-head lalu digabung).
5	Tambahkan residual *connection*: $z' = z^{l-1} + A$.
6	Normalisasi $u' = LN(z')$, lalu hitung FFN: $f = FFN(u')$.
7	Tambahkan residual untuk keluaran *layer*: $z^l = z' + f$.
8	Keluaran akhir *encoder* adalah z^L yang memuat representasi token $[CLS]$, $[DIST]$, dan *patch*.

4. *Output Layer* (*Classification Head*)

DeiT menghasilkan dua keluaran utama, yakni representasi dari token [CLS] yang digunakan untuk prediksi kelas akhir dan representasi dari token [DIST] yang berfungsi untuk distilasi pengetahuan. Definisi matematis *Output* dapat dilihat pada rumus (2.11).

$$\hat{y} = softmax(W_o \times h_{cls} + b) \quad (2.11)$$

Rumus 2.11. *Output Layer*

Loss Function Gabungan (Distillation *Loss*):

Untuk menggabungkan pembelajaran dari ground truth dan teacher, DeiT menggunakan *loss* kombinasi:

$$\mathcal{L}_{total} = \alpha \cdot \mathcal{L}_{\mathcal{CE}}(y_{cls}, y_{true}) + (1 - \alpha) \cdot \mathcal{L}_{\mathcal{KL}}(y_{dist}, y_{teacher}) \quad (2.12)$$

Rumus 2.12. *Distillation Loss*

Keterangan:

a. $\mathcal{L}_{\mathcal{CE}}$: *Cross-Entropy loss* terhadap label asli dari token [CLS].
b. $\mathcal{L}_{\mathcal{KL}}$: *Kullback-Leibler Divergence* antara token [DIST] dan *output* teacher.
c. α : bobot penggabungan (misalnya 0.5-0.9).

Tabel 2.8 *Pseudocode Output Layer (Classification Head)* dan *Distillation Loss*

No.	Tahapan
1	Ambil representasi token dari *encoder*: h_{cls} (token [CLS]) dan h_{dist} (token [DIST]) dari z^L.
2	Hitung prediksi kelas dari token [CLS]: $\hat{y}_{cls} = softmax\ (W_0 h_{cls} + b)$.
3	Hitung prediksi distilasi dari token [DIST] (head distilasi): $\hat{y}_{dist} = softmax\ (W_d h_{dist} + b_d)$.
4	Ambil label ground truth y_{true} dan *output* teacher $y_{teacher}$ (soft target).
5	Hitung *loss* klasifikasi: $\mathcal{L}_{\mathcal{CE}} = CE(\hat{y}_{cls}, y_{true})$.
6	Hitung distillation *loss*: $\mathcal{L}_{\mathcal{KL}} = KL(\hat{y}_{dist}, y_{teacher})$.

No.	Tahapan
7	Gabungkan *loss*: $L_{total} = \alpha . \mathcal{L}_{\mathcal{CE}} + (1 - \alpha) . \mathcal{L}_{\mathcal{KL}}$.
8	(*Training*) Lakukan *backpropagation* terhadap $\mathcal{L}_{total}$ untuk memperbarui parameter model.

2.3.7 BERT

Bidirectional Encoder Representations from Transformers (BERT) adalah model *Natural Language Processing* (NLP) berbasis *Transformer Encoder* yang dapat memahami konteks kata secara dua arah (*bidirectional*) [84]. Kelebihan utama BERT terletak pada kemampuannya dalam menangkap hubungan semantik baik di dalam satu kalimat maupun antar kalimat, sehingga efektif untuk berbagai tugas NLP, termasuk *sentiment analysis*.

Dalam tahap *pre-training*, BERT dilatih melalui dua tugas utama: *Masked Language Modeling* (MLM) dan *Next Sentence Prediction* (NSP). Pada MLM, model belajar menebak token yang disembunyikan (*masked tokens*) dengan memanfaatkan konteks dua arah dari kata-kata sekitarnya [85]. Semakin tepat prediksi token yang dilakukan model, semakin kecil nilai *loss* yang dihasilkan, yang menunjukkan pemahaman model terhadap konteks semantik. Secara matematis, fungsi kerugian (*loss function*) untuk MLM ditunjukkan pada Rumus (2.13) [10].

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \mathcal{M}} \log P\left(x_i \mid x_{\setminus i}\right) \quad (2.13)$$

Rumus 2.13. *Masked Language Modeling*

Keterangan:

a. $\mathcal{L}_{\text{MLM}}$: Fungsi *loss* yang digunakan untuk mengukur kesalahan prediksi model pada token yang disembunyikan.
b. M: Himpunan indeks token yang ditutupi (masked tokens).
c. x_i : Token tertentu yang menjadi target prediksi.
d. $x_{\setminus i}$: Semua token lain dalam kalimat yang digunakan sebagai konteks.
e. $P\left(x_i \mid x_{\setminus i}\right)$: Probabilitas model dalam memprediksi token yang benar berdasarkan konteks sekitarnya.

Tabel 2.9 *Pseudocode Masked Language Modeling* (MLM)

No.	Tahapan
1	Ambil kalimat/token *input* $[x_1; x_2; \ldots; x_T]$.
2	Pilih himpunan indeks token yang akan di-mask: $M \subset \{1, \ldots, T\}$.
3	Bentuk *input* ter-mask X' dengan mengganti token pada $i \in M$ menjadi [MASK] (atau variasi mask sesuai aturan pre-*training*).
4	Jalankan *encoder* BERT untuk memperoleh representasi kontekstual $H = BERT(X')$
5	Untuk setiap posisi $i \in M$, hitung probabilitas token asli menggunakan softmax pada head MLM: $P(x_i \mid x'_{\setminus i})$.
6	Hitung *loss* MLM: $L_{MLM} = \sum_{i \in \mathcal{M}} \log P(x_i \mid x'_{\setminus i})$
7	Lakukan *backpropagation* dan *update* parameter untuk meminimalkan L_{MLM}.

Tugas *Next Sentence Prediction* (NSP) dirancang untuk melatih model dalam memahami hubungan semantik antara dua kalimat, sehingga model dapat menangkap konteks yang lebih luas dalam teks. Pada proses ini, BERT diberikan sepasang kalimat (A, B) dan harus menentukan apakah kalimat kedua merupakan kelanjutan logis dari kalimat pertama atau tidak. Evaluasi dilakukan dengan menghitung nilai *loss*, yang mencerminkan seberapa baik model menilai urutan kalimat, dan perhitungan ini ditunjukkan pada Rumus (2.14) [10].

$$\mathcal{L}_{\text{NSP}} = -[y \setminus log\ P(y|A,B) + (1 - y) \setminus log(1 - P(y|A,B))] \quad (2.14)$$

Rumus 2.14. *Next Sentence Prediction* (NSP)

Keterangan:

a. $\mathcal{L}_{\text{NSP}}$: Fungsi *loss* yang digunakan untuk tugas *Next Sentence Prediction* (NSP).

b. y: Label biner, dengan nilai 1 jika kalimat B adalah kelanjutan dari kalimat A, dan 0 jika bukan.

c. $P(y|A,B)$: Probabilitas yang diprediksi model untuk menilai hubungan logis antara dua kalimat.

Tabel 2.10 *Pseudocode Next Sentence Prediction* (NSP)

No.	Tahapan
1	Ambil pasangan kalimat (A, B) dan label biner y (1 = B lanjutan A, 0 = bukan).
2	Bentuk *input* BERT: [CLS] A [SEP] B [SEP]) beserta *segment embedding* untuk A dan B.
3	Jalankan *encoder* BERT dan ambil representasi global token [CLS]: $h_{[\text{CLS}]}$.
4	Hitung probabilitas NSP: $p = P(y = 1 \mid A, B) = \sigma(W_{nsp} h_{[\text{CLS}]} + b_{nsp})$.
5	Hitung *loss* NSP (*binary cross-entropy*): $L_{nsp} = -(y \log p + (1 + y) \log(1 - p))$.
6	Lakukan *backpropagation* dan *update* parameter untuk meminimalkan L_{nsp}.

Tugas ini memungkinkan model untuk menangkap hubungan semantik antar kalimat, sehingga BERT mampu memahami konteks teks yang lebih panjang. Dalam penerapan untuk sentiment analysis, BERT memanfaatkan token khusus [CLS] sebagai representasi global dari keseluruhan teks. Token ini kemudian diproses melalui lapisan klasifikasi linier untuk menghasilkan probabilitas setiap kelas sentimen. Proses klasifikasi tersebut dijelaskan pada Rumus (2.15) [10].

$$\hat{y} = \mathit{softmax}\left(W \cdot h_{[CLS]} + b\right) \quad (2.15)$$

Rumus 2.15. Klasifikasi *Softmax*

Keterangan:

a. $\hat{y}$: Vektor yang berisi probabilitas prediksi untuk setiap kelas.
b. W: Bobot yang digunakan pada lapisan klasifikasi.
c. $h_{[CLS]}$: Vektor representasi global yang diperoleh dari token [CLS].
d. b : Bias pada lapisan klasifikasi.
e. *Softmax*: Fungsi aktivasi yang mengubah nilai logit menjadi distribusi probabilitas untuk tiap kelas.

Fungsi optimasi yang diterapkan dalam model ini adalah *cross-entropy loss*, yang digunakan untuk mengevaluasi sejauh mana distribusi probabilitas prediksi model sesuai dengan label asli. Fungsi ini menghitung perbedaan antara prediksi model dan nilai target, sehingga model dapat menyesuaikan bobotnya untuk meminimalkan kesalahan. Hubungan matematis antara prediksi dan label sebenarnya dijelaskan pada Rumus (2.16) [10].

$$\mathcal{L}_{\text{snie}} = -\sum_{c=1}^{C} y_c \cdot \log(\hat{y_c}) \quad (2.16)$$

Rumus 2.16. Fungsi *Loss Cross-Entropy*

Keterangan:

a. $\mathcal{L}_{\text{snie}}$: Nilai *loss* untuk tugas klasifikasi.
b. C: Jumlah kelas sentimen yang ada.
c. y_c : Label asli untuk kelas ke-c.
d. $\hat{y_c}$: Probabilitas prediksi model untuk kelas ke-c.

Tabel 2.11 *Pseudocode Fine-tuning* BERT untuk *Sentiment Analysis*

No.	Tahapan
1	Ambil teks ulasan dan label sentimen y (kelas 1..C).
2	Tokenisasi dan bentuk *input*: [CLS] *review* [SEP].
3	Jalankan *encoder* BERT dan ambil $h_{[\text{CLS}]}$.
4	Hitung logit kelas: $Wh_{[\text{CLS}]} + b$.
5	Hitung probabilitas prediksi: $\hat{y} = softmax(o)$
6	Hitung *loss* klasifikasi (*cross-entropy*): $L_{sent} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c)$
7	Lakukan *backpropagation* dan *update* parameter untuk meminimalkan L_{sent}.

Model ALBERT (*A Lite* BERT) merupakan pengembangan dari BERT yang menekankan efisiensi komputasi. Dua strategi utama yang diterapkan adalah parameter *sharing* antar lapisan dan *factorized embedding* [86]. Selain itu, ALBERT menggantikan tugas *Next Sentence Prediction* (NSP) dengan *Sentence Order Prediction* (SOP), yang melatih model untuk menentukan apakah dua kalimat disusun dalam urutan yang logis [87]. Fungsi *loss* untuk SOP secara matematis sama dengan NSP, dan perhitungannya ditunjukkan pada Rumus (2.17) [88].

$$\mathcal{L}_{\text{SOP}} = -[y \cdot \log P + (1 - y) \cdot \log(1 - P)] \quad (2.17)$$

Rumus 2.17. Fungsi *Loss Sentence Order Prediction* (SOP)

Keterangan:

a. $\mathcal{L}_{\text{SOP}}$: Fungsi *loss* untuk tugas SOP.

b. y: Label biner (1 jika urutan benar, 0 jika salah).

c. P : Probabilitas prediksi model terhadap kebenaran urutan kalimat.

Tabel 2.12 *Pseudocode Sentence Order Prediction* (SOP) pada *ALBERT*

No.	Tahapan
1	Ambil dua segmen kalimat (S_1, S_2).
2	Bentuk pasangan *input* dengan dua kemungkinan: urutan benar (S_1, S_2) atau ditukar (S_2, S_1).
3	Set label y (1 = urutan benar, 0 = urutan salah).
4	Bentuk *input*: $[CLS]\ S_1[SEP]\ S_2[SEP]$) (sesuai pasangan yang dipilih).
5	Jalankan *encoder* ALBERT dan ambil $h_{[\text{CLS}]}$.
6	Hitung probabilitas urutan benar: $p = \sigma(W_{sop} h_{[\text{CLS}]} + b_{sop})$
7	Hitung *loss* SOP: $L_{sop} = -(y \log p + (1 - y) \log(1 - p))$.
8	Lakukan *backpropagation* dan *update* parameter untuk meminimalkan L_{sop}.

RoBERTa (*Robustly Optimized* BERT *Approach*) menyederhanakan tahap pra-latih dengan hanya MLM dan menerapkan dynamic masking di setiap *epoch* agar cakupan konteks lebih beragam. Pendekatan ini mendorong pembelajaran pola bahasa yang lebih kaya tanpa ketergantungan pada tugas NSP. Mekanisme klasifikasi tetap bertumpu pada representasi [CLS] sebagaimana pada persamaan (2.15), dan fungsi kerugian yang digunakan tetap *cross-entropy* pada (2.16).

IndoBERT. Varian BERT yang dipra-latih khusus untuk bahasa Indonesia pada korpus berukuran besar sehingga kosakata, distribusi kata, dan fenomena morfosintaksis lokal terwakili lebih baik [89]. Arsitektur dasarnya sama dengan BERT, berfokus pada tujuan MLM saat pra-latih, sementara NSP atau SOP hadir sesuai rilis model, dan kepala klasifikasi tetap menggunakan token [CLS] sesuai persamaan (2.15) dan (2.16). Pada berbagai tugas teks Indonesia termasuk klasifikasi dan *sentiment analysis*, IndoBERT umumnya melampaui BERT berbahasa Inggris yang dipakai lintas bahasa berkat kesesuaian korpus pra-latih dengan *domain* Indonesia [14][28]. Perbandingan ringkas antarmodel disajikan pada Tabel 2.2 Perbedaan Model BERT dan Variannya.

Tabel 2.13 Perbedaan Model BERT

Aspek	**BERT**	**ALBERT**	**RoBERTa**	IndoBERT
Arsitektur	*Transformer Encoder*	Parameter Sharing	Optimasi pelatihan	*Transformer Encoder* berbahasa Indonesia
Tugas Pre-*train*ing	MLM (2.6) + NSP (2.7)	MLM (2.6) + SOP (2.10)	MLM (2.6) saja	MLM dominan (NSP/SOP bergantung varian)
St*rate*gi Masking	Statis	Statis	Dinamis	Umumnya sesuai rilis model
Efisiensi Parameter	Sedang	Tinggi	Rendah	Sedang
Penggunaan Token [CLS]	Ya	Ya	Ya	Ya
Ukuran Model	Besar	Lebih kecil	Besar	Sekelas BERT *base*
Fungsi Klasifikasi	*Softmax* atas [CLS] (2.8)	Sama	Sama	Sama

Aspek	BERT	ALBERT	RoBERTa	IndoBERT
Fungsi *Loss*	*Cross-entropy* (2.9)	Sama	Sama	Sama

BERT menggunakan arsitektur *Transformer Encoder* dengan dua tugas utama, yaitu *Masked Language Modeling* (MLM) (2.13) dan *Next Sentence Prediction* (NSP) (2.14) untuk memahami konteks kata dan hubungan antarkalimat. ALBERT mengoptimalkan efisiensi melalui parameter *sharing* dan *factorized embedding* serta mengganti NSP dengan *Sentence Order Prediction* (SOP) (2.17) agar lebih peka pada urutan kalimat. RoBERTa menyederhanakan pra-latih dengan hanya MLM (2.13) dan menerapkan *dynamic masking* di setiap *epoch* guna memperkaya variasi konteks. IndoBERT dipra-latih pada korpus berbahasa Indonesia berukuran besar sehingga kosakata, morfologi, dan struktur kalimat lokal lebih terwakili, dengan skema pra-latih yang didominasi MLM dan variasi NSP atau SOP mengikuti rilis model. Dari sisi efisiensi, ALBERT paling ringan, BERT moderat, RoBERTa relatif berat, sedangkan IndoBERT berada di kelas BERT *base* namun unggul pada tugas teks Indonesia berkat kesesuaian korpus. Keempatnya tetap memakai token [CLS] sebagai representasi global, klasifikasi softmax pada (2.15), dan *cross-entropy loss* pada (216).

2.3.8 *Wordcloud*

Wordcloud adalah teknik visualisasi teks yang menampilkan kata atau token dari sebuah korpus dengan ukuran yang merepresentasikan frekuensi atau bobot kepentingannya, sehingga istilah dominan dapat terlihat cepat [90]. Visualisasi ini biasanya dibangun dari representasi *bag of words* setelah tahap praproses seperti pembersihan tanda baca, normalisasi huruf, penghapusan *stopwords*, serta opsi stemming atau lemmatisasi, kemudian kata dapat diberi bobot frekuensi atau TF-IDF. Algoritma tata letak menempatkan kata dalam ruang dua dimensi tanpa tumpang tindih, dengan warna dan orientasi sebagai elemen estetika atau pengodean informasi tambahan [91]. Meskipun berguna untuk eksplorasi awal, wordcloud bersifat deskriptif dan tidak memodelkan hubungan semantik maupun urutan kata, sehingga idealnya digunakan sebagai pelengkap ringkasan kuantitatif seperti tabel frekuensi atau analisis topik [92].

2.3.9 *Latent Dirichlet Allocation*

Latent Dirichlet Allocation (LDA) merupakan salah satu model *topic modeling* berbasis probabilistik yang digunakan untuk menemukan topik-topik laten dalam kumpulan dokumen teks tanpa memerlukan label manual. Secara konsep, LDA memodelkan setiap dokumen sebagai campuran beberapa topik, dan setiap topik sebagai campuran beberapa kata [93]. Dengan pendekatan ini, LDA berusaha menangkap pola ko-kemunculan kata di dalam dokumen sehingga dapat membentuk kelompok kata yang merepresentasikan topik tertentu (misalnya topik "kualitas bahan", "ukuran dan *fit*", atau "pengiriman"). Secara matematis, asumsi dasar LDA mengenai distribusi topik pada dokumen dan distribusi kata pada topik dapat dinyatakan melalui persamaan (2.18) [94]:

$$\theta_d \sim Dir(\alpha), \emptyset_k \sim Dir(\beta) \quad (2.18)$$

Rumus 2.18. *Latent Dirichlet Allocation*

Keterangan:

a. θ_d : Vektor distribusi topik untuk dokumen ke-ddd, yang menyatakan proporsi masing-masing topik di dalam dokumen tersebut.
b. $Dir(\alpha)$: Distribusi *Dirichlet* dengan parameter hiper α yang digunakan sebagai prior untuk distribusi topik pada dokumen (θ_d).
c. $\emptyset_k$: Vektor distribusi kata untuk topik ke-kkk, yang menyatakan probabilitas setiap kata muncul dalam topik tersebut.
d. $Dir(\beta)$: Distribusi *Dirichlet* dengan parameter hiper β\betaβ yang digunakan sebagai prior untuk distribusi kata pada topik ($\emptyset_k$).
e. α : Hiperparameter yang mengatur bentuk/sebaran distribusi topik dalam dokumen (seberapa "rata" atau "tajam" campuran topiknya).
f. β: Hiperparameter yang mengatur bentuk/sebaran distribusi kata dalam topik (seberapa banyak kata yang punya probabilitas besar di satu topik).

Tabel 2.14 *Pseudocode Latent Dirichlet Allocation* (LDA)

No.	Tahapan
1	Siapkan korpus dokumen $\{d_1, \dots, d_D\}$, jumlah topik K, serta hiperparameter α dan β.
2	Inisialisasi distribusi topik dokumen dan distribusi kata-topik (sesuai *prior*): $\theta_d \sim Dir(\alpha)$ dan $\emptyset_k \sim Dir(\beta)$.

No.	Tahapan
3	Untuk setiap kata (token) pada setiap dokumen, tetapkan topik awal secara acak: $z_{dn} \in \{1, \dots, K\}$
4	Ulangi untuk sejumlah iterasi (hingga konvergen): perbarui topik setiap token dengan menghitung probabilitas topik k berdasarkan hitungan sementara pada dokumen dan topik.
5	*Sampling* topik baru untuk token tersebut menggunakan distribusi posterior (*Gibbs sampling*), lalu perbarui hitungan kata-topik dan dokumen-topik.
6	Setelah iterasi selesai, estimasi θ_d (proporsi topik per dokumen) dan $\emptyset_k$ (probabilitas kata per topik) dari hitungan akhir.
7	Ambil kata-kata dengan probabilitas tertinggi pada tiap $\emptyset_k$ sebagai representasi topik, lalu interpretasikan topik (mis. "kualitas bahan", "ukuran & fit", "pengiriman").

2.4 *Software* dan *Tool's* yang digunakan

2.4.1 Python

Python adalah bahasa pemrograman tingkat tinggi yang bersifat dinamis dan dijalankan melalui interpreter, dengan fokus pada keterbacaan kode serta dukungan multiparadigma seperti prosedural, berorientasi objek, dan fungsional, sehingga banyak digunakan untuk skrip, pengembangan aplikasi, dan eksperimen komputasi [95][96]. Ekosistem Python didukung oleh pustaka standar dan manajemen paket yang luas, dengan reprodusibilitas penelitian dijaga melalui virtual environment (venv) dan instalasi dependensi menggunakan pip agar versi pustaka tetap konsisten [97]. Dalam analisis data, NumPy menyediakan komputasi numerik berbasis array multidimensi, sementara pandas menawarkan struktur Series dan DataFrame untuk pengolahan data tabular, mendukung pembersihan data, transformasi fitur, dan pengembangan model Machine Learning [98].

2.4.2 Visual Studio Code

Visual Studio Code (*VSCode*) merupakan penyunting kode yang dikembangkan *Microsoft* dan dirancang untuk mendukung proses pengembangan perangkat lunak secara efisien [99]. Walau ringkas, *VSCode* menyediakan fitur esensial seperti *syntax highlighting*, *auto-completion*, fasilitas *debugging*, serta integrasi kendali versi *Git* sehingga penulisan, pengeditan, dan pengelolaan kode menjadi lebih terstruktur. Dukungan bahasa pemrograman diperluas melalui *extensions marketplace* yang memungkinkan penambahan fungsionalitas sesuai kebutuhan misalnya untuk *Python*, *JavaScript*, dan *C++* disertai antarmuka yang

intuitif dan dapat disesuaikan, termasuk *linter* dan *formatter* agar gaya penulisan selaras dengan standar proyek. Ketersediaan lintas sistem operasi *Windows*, *Linux*, dan *macOS* menempatkan *VSCode* sebagai pilihan yang fleksibel dan banyak digunakan pada beragam konteks, mulai dari pengembangan *web* dan aplikasi hingga riset data [100].

2.4.3 Google Collab

Google Colab adalah *platform* notebook interaktif yang menjalankan komputasi Python pada *server cloud* Google, sehingga eksperimen analisis data dan *machine learning* dapat dilakukan langsung dari peramban web dengan proses penyiapan yang minimal, sekaligus mendukung dokumentasi yang rapi karena satu notebook menggabungkan sel kode yang dapat dieksekusi dan narasi berbasis Markdown [101]. Eksekusi notebook berlangsung pada *runtime environment* berbasis mesin virtual yang menyediakan pilihan akselerator, seperti Central Processing Unit (CPU), Graphics Processing Unit (GPU), dan Tensor Processing Unit (TPU), sehingga pelatihan model *deep learning* dan pemrosesan data berskala besar dapat dipercepat tanpa bergantung pada spesifikasi perangkat lokal. Notebook juga dapat disimpan, diakses, dan dibagikan melalui integrasi dengan Google Drive sehingga kolaborasi dan replikasi eksperimen lebih mudah [102].