BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian terdahulu merupakan salah satu bagian penting dalam penelitian yang digunakan untuk memahami perkembangan metode, pendekatan, serta hasil penelitian yang telah dilakukan sebelumnya terkait topik yang diteliti [1]. Melalui penelitian terdahulu, peneliti dapat mengetahui kelebihan, kekurangan, serta celah penelitian (*research gap*) yang masih dapat dikembangkan lebih lanjut [25]. Selain itu, penelitian terdahulu juga berfungsi sebagai referensi dalam menentukan metode, model, maupun pendekatan yang sesuai dengan tujuan penelitian yang dilakukan. Ringkasan penelitian terdahulu yang berkaitan dengan deteksi berita *hoax* berbasis teks, gambar, maupun *multimodal* dapat dilihat pada Tabel 2.1.

Penerapan IndoBERT dalam deteksi berita *hoax* berbahasa Indonesia menggunakan metode *fine-tuning* IndoBERT pada *Mafindo API* dan *Indonesia News Dataset* untuk mendeteksi *hoax* berbasis teks. Hasil penelitian menunjukkan bahwa IndoBERT memperoleh *accuracy* sebesar 98,51%, *precision* sebesar 98,23%, dan *F1-score* sebesar 98,44%, sehingga efektif digunakan dalam mendukung proses deteksi *hoax* berbahasa Indonesia [15]. Namun, penelitian tersebut masih menggunakan pendekatan *unimodal* berbasis teks sehingga belum memanfaatkan informasi visual dari gambar berita. Penelitian lain membandingkan performa IndoBERT dan mBERT pada klasifikasi berita *hoax* politik berbahasa Indonesia. Penelitian tersebut menunjukkan bahwa IndoBERT memperoleh performa lebih baik dibandingkan mBERT dengan *accuracy* sebesar 95,60%, sedangkan mBERT memperoleh *accuracy* sebesar 91,97%. Hasil tersebut menunjukkan bahwa model yang dilatih secara khusus menggunakan korpus Bahasa Indonesia memiliki kemampuan yang lebih optimal dalam memahami konteks berita *hoax* dibandingkan model *multilingual* [13]. Namun, penelitian tersebut hanya memanfaatkan informasi tekstual sehingga belum mampu menganalisis informasi visual yang sering muncul pada konten *hoax multimodal*.

Selain pendekatan berbasis teks, penelitian juga banyak memanfaatkan arsitektur *Convolutional Neural Network* (CNN) dan ResNet-50 dalam proses analisis visual. CNN berbasis ResNet-50 untuk mendeteksi manipulasi visual pada gambar *deepfake*. Hasil penelitian menunjukkan bahwa model memperoleh *accuracy* sebesar 85%, *precision* sebesar 100%, dan *recall* sebesar 65%. Penelitian tersebut menunjukkan bahwa ResNet-50 efektif dalam melakukan ekstraksi fitur visual pada gambar. Namun, performa model mengalami penurunan pada beberapa kondisi visual kompleks dan variasi gambar tertentu [26]. Penelitian lain juga menggunakan ResNet-50 dengan pendekatan *transfer learning* dan *fine-tuning* untuk mendeteksi gambar wajah *deepfake*. Hasil penelitian menunjukkan bahwa model memperoleh *precision* sebesar 98%, *recall* sebesar 96%, dan *F1-score* sebesar 97%. Penelitian tersebut menunjukkan bahwa ResNet-50 mampu menghasilkan performa yang baik dalam mendeteksi manipulasi visual pada gambar wajah [17]. Namun, penelitian tersebut berfokus pada analisis visual sehingga belum memanfaatkan informasi tekstual yang terdapat pada gambar maupun narasi berita.

Pendekatan OCR 2.5, *BERT-base uncased*, dan ResNet-18 pada *Dataset fake news* berbasis teks dan gambar yang mengandung teks visual memiliki penelitian menunjukkan bahwa pendekatan OCR *multimodal* menghasilkan *accuracy* teks sebesar 95,12% dan *accuracy* gambar sebesar 97,56%. Penelitian tersebut menunjukkan bahwa pemanfaatan OCR mampu meningkatkan analisis *multimodal* melalui ekstraksi teks pada gambar [20]. Namun, penelitian tersebut masih menggunakan *BERT-base uncased* yang belum secara khusus dilatih untuk Bahasa Indonesia. CNN, BiLSTM, IndoBERT, OCR, dan *weighted soft-voting ensemble* pada *Dataset multimodal* media sosial Indonesia memiliki hasil penilitian menunjukkan bahwa model memperoleh *accuracy* sebesar 95,42%, *precision* sebesar 95,00%, *recall* sebesar 95,62%, dan *F1-score* sebesar 95,31% [27]. Penelitian tersebut menunjukkan bahwa OCR dan pendekatan *multimodal* mampu menangani variasi visual dan *noisy OCR* pada konten media sosial. Meskipun penelitian tersebut berfokus pada deteksi iklan judi online, karakteristik *Dataset* yang digunakan memiliki kemiripan dengan penelitian ini karena sama-sama menggunakan gambar media sosial yang mengandung teks visual, poster *digital*, serta variasi kualitas gambar. Penelitian terkait *Hyperparameter-Tuned BERT*

(HTBERT) dan ResNet-110 untuk mendeteksi *fake news multimodal* berbasis teks dan gambar. Hasil penelitian menunjukkan bahwa model memperoleh *accuracy* sebesar 93,1%, *precision* sebesar 94,4%, *recall* sebesar 94,2%, dan *F1-score* sebesar 94,6%. Penelitian tersebut menunjukkan bahwa integrasi BERT dan ResNet mampu meningkatkan performa deteksi *fake news multimodal* [24]. Namun, penggunaan ResNet-110 memiliki kompleksitas komputasi yang cukup tinggi.

Penelitian yang membahas terkait *attention fusion* pada model *multimodal* untuk mendeteksi *misinformation* berbasis teks dan gambar pada *dataset Gossipcop* dan *Politifact*. Penelitian tersebut menunjukkan bahwa model memperoleh *accuracy* sebesar 88,55% pada *dataset Gossipcop* dan 88.46% pada *dataset Politifact*. Pendekatan *attention fusion* digunakan untuk meningkatkan hubungan antar modalitas dalam proses klasifikasi [28]. Namun, arsitektur *fusion* yang digunakan memiliki kompleksitas komputasi yang cukup tinggi. Pada Penelitian yang membahas BERT dan ResNet-50 dalam pendekatan *multimodal* berbasis *Feature concatenation*. Penelitian tersebut juga menerapkan Adam *optimizer*, *dropout*, dan *early stopping* pada proses pelatihan model. Hasil penelitian menunjukkan bahwa model memperoleh *accuracy* sebesar 93,4%, *precision* sebesar 92,7%, *recall* sebesar 93,1%, dan *F1-score* sebesar 93,4% [29]. Penelitian tersebut menunjukkan bahwa integrasi BERT dan ResNet-50 mampu meningkatkan performa deteksi *multimodal* dibandingkan pendekatan *unimodal*. Meskipun demikian, penelitian tersebut masih menggunakan *dataset* berbahasa Inggris dan berfokus pada *fake review*, bukan *fake news*.

Berdasarkan penelitian terdahulu, berbagai pendekatan berbasis *Transformer*, OCR, dan *multimodal* telah menunjukkan performa yang baik dalam mendeteksi informasi palsu. Namun, integrasi OCR, IndoBERT, dan ResNet-50 dalam kerangka *multimodal* untuk deteksi berita hoax berbahasa Indonesia masih belum banyak dieksplorasi. Selain itu, kajian yang membandingkan secara langsung performa model berbasis teks, gambar, dan *multimodal* pada *dataset* berita hoax berbahasa Indonesia yang sama juga masih terbatas. Oleh karena itu, penelitian ini mengusulkan pendekatan *multimodal* berbasis OCR, IndoBERT, dan ResNet-50 serta melakukan perbandingan performa dengan model *unimodal* untuk mengevaluasi efektivitas pendekatan yang diusulkan.

Tabel 2.1 Tabel Penelitian Terdahulu

Penulis	Judul Artikel	*Dataset*	Metode	Masalah	Hasil	Kelebihan	Kekurangan
S. Y. Yakub, R. J. Wowor, D. Agustriawan, Irmawati, D. A. Kristiyanti, dan P. M. Winarno [15]	*Deep Learning for Cyber Security: Deephoax Detection* (2025)	*Mafindo API dan Indonesia News Dataset*	Fine-tuning IndoBERT untuk deteksi hoax berbasis teks	Penyebaran *hoax* berbahasa Indonesia membutuhkan model NLP yang lebih akurat	*Accuracy* 98,51%, *precision* 98,23%, dan *F1-score* 98.44%	IndoBERT sangat efektif untuk deteksi *hoax* Bahasa Indonesia	Hanya menggunakan modalitas teks
Charlotte Jocelynne L. Tobing, I. G. N. Lanang Wijayakusuma, dan L. P. I. Harini [13]	Perbandingan Kinerja IndoBERT dan MBERT untuk Deteksi Berita Hoaks Politik dalam Bahasa Indonesia (2025)	*Dataset berita hoax politik berbahasa Indonesia*	Fine-tuning IndoBERT dan MBERT	Model *multilingual* belum tentu optimal pada konteks Bahasa Indonesia	IndoBERT memperoleh *Accuracy* 95,60% sedangkan MBERT 91,97%	Membandingkan performa IndoBERT dan MBERT secara langsung	Penelitian masih berbasis teks *unimodal*
A. Awalina, J. Fawaid, R. Y. Krisnabayu, dan N. Yudistira [30].	Indonesia's Fake News Detection using Transformer Network (2021)	TurnBackHoax Dataset dan dataset berita Indonesia	*BERT with Transformer Network*	Penyebaran berita hoax Bahasa Indonesia	*Accuracy* hingga 90%	BERT mampu memberikan performa lebih baik dibanding CNN dan BiLSTM pada deteksi hoax Bahasa Indonesia	Hanya menggunakan analisis teks dan belum memanfaatkan informasi *multimodal*
A. Adhicitta [26]	*Application of Deep Learning for Image Deepfake Detector Using Convolutional Neural Network Algorithm* (2023)	*Dataset gambar deepfake*)	CNN berbasis ResNet-50	Kesulitan mendeteksi manipulasi visual pada gambar *deepfake*	*Accuracy* 85%, *precision* 100%, dan *recall* 65%	ResNet-50 efektif dalam ekstraksi fitur visual	Performa menurun pada beberapa kondisi visual kompleks dan variasi gambar tertentu

Penulis	Judul Artikel	*Dataset*	Metode	Masalah	Hasil	Kelebihan	Kekurangan
L. Kroiß dan J. Reschke [17]	*Deepfake Detection of Face Images Based on a Convolutional Neural Network* (2025)	*Dataset gambar wajah deepfake*	ResNet-50 dengan transfer learning dan fine-tuning	*Deepfake* sulit dibedakan dengan gambar asli	*Precision* 98%, *recall* 96%, dan *F1-score* 97%	ResNet-50 efektif mendeteksi manipulasi visual	Tidak menggunakan pendekatan *multimodal*
J. Vennapu [20]	*Multimodal Fake News Detection: Integrating OCR and Deep Learning Models for Text and Image Analysis* (2024)	*Dataset fake news berbasis teks dan gambar yang mengandung teks visual*	OCR 2.5 + BERT-base uncased + ResNet-18	Informasi teks pada gambar sering tidak dianalisis pada model *fake news detection*	Pendekatan OCR *multimodal* menghasilkan *accuracy* teks 95,12% dan *accuracy* gambar 97,56%	Memanfaatkan OCR untuk mengekstraksi teks dari gambar sehingga meningkatkan analisis *multimodal*	Menggunakan *BERT-base uncased* yang belum spesifik untuk Bahasa Indonesia
M. I. Alfiansyah dan A. Muzakir [27]	*Enhanced Detection of Indonesian Online Gambling Advertisements Using Multimodal Ensemble Deep Learning* (2025)	*Dataset multimodal media sosial Indonesia*	CNN, BiLSTM, IndoBERT, OCR, dan *weighted soft-voting ensemble*	Konten media sosial *multimodal* sulit dianalisis karena *noisy OCR* dan variasi visual	*Accuracy* 95,42%, *precision* 95,00%, *recall* 95,62%, dan *F1-score* 95,31%	Mendukung OCR dan *multimodal* berbahasa Indonesia	Fokus penelitian pada deteksi judi online, bukan *fake news*
C. Agrawal, A. Pandey, dan S. Goyal [24]	*Multimodal Fake News Detection using Hyperparameter-Tuned* BERT *and* ResNet110 (2024)	*Dataset fake news multimodal berbasis teks dan gambar*	*Hyperparameter-*Tuned BERT (HTBERT) untuk teks dan ResNet-110 untuk gambar	Model *fake news* sebelumnya masih kurang optimal dalam memahami hubungan antara teks dan gambar secara bersamaan	Model HTBERT + ResNet-110 memperoleh *Accuracy* 93,1%, *precision* 94,4%, *recall* 94,2%, dan *F1-score* 94,6%	Mampu meningkatkan performa deteksi *fake news multimodal* dengan optimasi *hyperparameter*	Menggunakan ResNet-110 yang memiliki kompleksitas komputasi tinggi

Penulis	Judul Artikel	*Dataset*	Metode	Masalah	Hasil	Kelebihan	Kekurangan
S. Mehreen, A. Bhuiyan, dan E. Hossain [28]	*An Attentive Fusion Framework For Multi-modal Misinformation Detection* (2024)	*Dataset Gossipcop dan Politifact*	Model multimodal sebelumnya belum optimal dalam melakukan *Feature fusion*	*Model multimodal sebelumnya belum optimal dalam melakukan Feature fusion*	*Accuracy* mencapai 88,55% pada *Gossipcop* dan 88,46% pada *Politifact*	Menggunakan mekanisme *attention fusion* untuk meningkatkan hubungan antar modalitas	Arsitektur *fusion* cukup kompleks dan membutuhkan komputasi tinggi
S. R. Veluru, S. T. Erukude, dan V. C. Marella [29]	*Multimodal Detection of Fake Reviews Using* BERT *and* ResNet-50	*20.144 data fake review multimodal teks dan gambar*	BERT, ResNet-50, Feature concatenation, Adam optimizer, dropout, dan early stopping	Pendekatan *unimodal* kurang mampu memahami ketidaksesuaian antara teks dan gambar	*Accuracy* 93,4%, *precision* 92,7%, *recall* 93,1%, dan *F1-score* 93,4%	92.7%, *recall* 93.1%, dan *F1-score* 93.4% Mengintegrasikan BERT dan ResNet-50 dalam pendekatan *multimodal*	*Dataset* menggunakan Bahasa Inggris dan fokus pada *fake review*, bukan *fake news*

2.2 Tinjauan Teori

2.2.1 *Hoax*

Hoax merupakan informasi yang tidak benar atau menyesatkan yang sengaja dibuat dan disebarkan untuk mempengaruhi opini publik. Istilah *hoax* sering dikaitkan dengan *fake news*, yaitu berita palsu yang disajikan seolah-olah sebagai informasi yang valid. Dalam konteks komunikasi *digital*, *hoax* tidak hanya berbentuk teks, tetapi juga dapat berupa gambar, video, maupun kombinasi berbagai media [31]. Perkembangan teknologi informasi, khususnya internet dan media sosial, telah mempercepat penyebaran *hoax* secara signifikan karena informasi dapat tersebar luas tanpa melalui proses verifikasi yang memadai. Hal ini meningkatkan risiko penyebaran *misinformation* dan *disinformation*, di mana *hoax* sering dirancang dengan judul provokatif, narasi emosional, serta visual yang menarik agar mudah dipercaya dan dibagikan oleh pengguna [32].

Secara umum, *hoax* dapat diklasifikasikan ke dalam beberapa jenis. *Fabricated content* merupakan informasi yang sepenuhnya dibuat tanpa dasar fakta. *Manipulated content* adalah informasi yang dimodifikasi dari fakta yang ada sehingga menjadi menyesatkan. *Misleading content* adalah informasi yang digunakan secara tidak tepat untuk menggiring opini tertentu. *False context* merujuk pada informasi yang sebenarnya benar namun ditempatkan dalam konteks yang salah. Sementara itu, *satire* atau *parody* adalah konten yang bersifat humor tetapi sering disalahartikan sebagai fakta. Penyebaran *hoax* memiliki dampak yang luas, mulai dari menurunnya kepercayaan publik terhadap media, munculnya konflik sosial, hingga terganggunya stabilitas Masyarakat [33].

2.2.2 *Web* Berita

Web berita adalah platform *digital* yang digunakan untuk menyajikan informasi dan peristiwa terkini kepada masyarakat melalui jaringan internet. Berbeda dengan media cetak konvensional, *web* berita memungkinkan distribusi informasi secara cepat, luas, dan *real-time*, sehingga pengguna dapat mengakses berita kapan saja dan di mana saja [34]. Konten dalam *web* berita umumnya disusun secara terstruktur, terdiri dari judul, isi berita, gambar pendukung, serta informasi tambahan seperti waktu publikasi dan sumber. Dalam perkembangannya, *web*

berita tidak hanya dimanfaatkan oleh media resmi, tetapi juga oleh berbagai pihak yang tidak selalu memiliki kredibilitas jurnalistik. Hal ini menyebabkan munculnya *web* berita tidak terpercaya yang menyebarkan informasi *hoax* dengan tampilan yang menyerupai media professional [35].

2.2.3 Instagram

Instagram merupakan platform media sosial berbasis visual yang memungkinkan pengguna untuk membagikan konten berupa gambar, video, serta teks dalam bentuk *caption*. Platform ini memiliki karakteristik utama berupa penyajian informasi yang menarik secara visual, sehingga mampu meningkatkan perhatian dan interaksi pengguna secara signifikan [36]. Fitur seperti *likes*, komentar, dan *sharing* memungkinkan informasi tersebar dengan cepat dalam jaringan pengguna yang luas. Banyak konten *hoax* dikemas dalam bentuk poster, infografis, atau tangkapan layar yang terlihat meyakinkan, sehingga sulit dibedakan dari informasi yang valid [37].

2.2.4 *Web Scraping*

Web scraping merupakan teknik pengumpulan data secara otomatis dari berbagai *website* menggunakan program atau *script* tertentu. Teknik ini memungkinkan peneliti untuk mengekstraksi informasi dalam jumlah besar, seperti teks, gambar, judul berita, tanggal publikasi, tautan, maupun metadata lainnya yang kemudian dapat digunakan sebagai *dataset* untuk analisis lebih lanjut [38]. *Web scraping* biasanya dilakukan dengan mengakses struktur HTML dari suatu halaman *web*, kemudian mengambil elemen data yang dibutuhkan secara sistematis berdasarkan tag, class, atau atribut tertentu pada halaman tersebut [39].

2.2.5 *Deep Learning*

Deep learning merupakan pengembangan dari *machine learning* yang berfokus pada algoritma yang terinspirasi dari struktur dan fungsi otak manusia melalui *artificial neural networks*. Pendekatan ini menjadi sangat populer karena kemampuannya dalam menangani data dalam jumlah besar dan kompleks [40]. Berbeda dengan *machine learning* tradisional yang sering memerlukan intervensi manual dalam proses *Feature engineering* untuk menentukan fitur yang relevan,

deep learning mampu secara otomatis mengekstraksi dan mempelajari fitur penting dari data, sehingga meningkatkan efisiensi terutama pada tahap *modeling*. *Deep learning* terbukti mampu meningkatkan akurasi secara signifikan, terutama pada data dengan dimensi tinggi dan tingkat kompleksitas yang besar. Kemampuan ini menjadikan *deep learning* sangat sesuai untuk digunakan pada data yang berasal dari media sosial, yang umumnya bersifat tidak terstruktur dan beragam, seperti teks dan gambar yang diperoleh melalui teknik *web scraping* [41].

2.2.6 *Data pre-processing*

Data pre-processing merupakan tahapan awal dalam *machine learning* dan *deep learning* yang bertujuan untuk mempersiapkan data agar siap digunakan dalam proses pelatihan model. Tahapan ini mencakup proses pembersihan, transformasi, dan normalisasi data sehingga memiliki kualitas yang lebih baik, terstruktur, dan sesuai dengan kebutuhan model. *Data pre-processing* berperan penting dalam mengurangi *noise*, menangani data yang tidak konsisten, serta meningkatkan akurasi model [42]. Pada data teks, tahapan *data pre-processing* secara umum meliputi:

1. *Case folding*

   Case folding adalah proses normalisasi teks dengan mengubah seluruh huruf dalam suatu teks menjadi huruf kecil (*lowercase*). Proses ini bertujuan untuk menghilangkan perbedaan penulisan akibat penggunaan huruf kapital sehingga kata yang sama tidak dianggap berbeda oleh sistem. Dengan adanya *case folding*, data teks menjadi lebih konsisten dan memudahkan proses analisis serta pemodelan.
2. *Tokenization*

   Tokenization adalah proses pemecahan teks menjadi unit-unit kecil seperti kata, frasa, atau subkata yang disebut token. Proses ini bertujuan untuk mempermudah analisis teks dengan mengubah data teks menjadi bentuk yang lebih terstruktur dan dapat diproses oleh model. Dengan *tokenization*, setiap kata dalam teks dapat dianalisis secara individual sehingga membantu dalam proses pembelajaran model.
3. Penghapusan *stopwords*

Penghapusan *stopwords* adalah proses menghilangkan kata-kata umum dalam teks yang tidak memiliki makna signifikan terhadap analisis, seperti “dan”, “yang”, atau “di”. Proses ini bertujuan untuk mengurangi *noise* dan fokus pada kata-kata penting yang lebih relevan dalam menentukan makna suatu teks. Dengan menghapus stopwords, kualitas fitur yang digunakan dalam pemodelan dapat meningkat sehingga membantu meningkatkan kinerja model.

4. *Cleaning*

   Cleaning adalah proses pembersihan teks dengan menghapus karakter khusus, angka, atau simbol yang tidak relevan. Proses ini bertujuan untuk mengurangi *noise* dalam data sehingga teks menjadi lebih bersih dan mudah diproses. Dengan melakukan *cleaning*, kualitas data meningkat dan membantu model dalam memahami pola teks secara lebih akurat.

5. *Padding* dan *truncation*

   Padding dan truncation adalah proses penyesuaian panjang teks agar sesuai dengan ukuran input yang dibutuhkan oleh model. Padding dilakukan dengan menambahkan token khusus pada teks yang lebih pendek, sedangkan truncation dilakukan dengan memotong teks yang terlalu panjang. Proses ini bertujuan untuk memastikan seluruh data memiliki panjang yang seragam sehingga dapat diproses secara efisien oleh model.

Sedangkan pada data gambar, tahapan *pre-processing* meliputi:

1. *Resizing*

   Resizing adalah proses penyesuaian ukuran gambar agar memiliki dimensi yang seragam sesuai dengan kebutuhan model. Proses ini bertujuan untuk memastikan setiap gambar memiliki ukuran yang konsisten sehingga dapat diproses secara optimal oleh model. Dengan melakukan resizing, gambar dapat diolah lebih efisien dan membantu meningkatkan kinerja model dalam mengenali pola visual.

2. Normalisasi

   Normalisasi adalah proses transformasi nilai piksel gambar ke dalam rentang tertentu, biasanya antara 0-1 atau dengan distribusi tertentu. Proses ini

bertujuan untuk menyamakan skala nilai piksel sehingga model dapat belajar dengan lebih stabil dan cepat. Dengan melakukan normalisasi, perbedaan intensitas yang ekstrem dapat dikurangi sehingga membantu meningkatkan performa model.

3. *Data augmentation*

   Data augmentation adalah teknik untuk meningkatkan variasi data dengan melakukan transformasi pada data asli, seperti rotasi, *flipping*, atau perubahan skala. Proses ini bertujuan untuk memperkaya *dataset* tanpa menambah data baru secara langsung. Dengan data *augmentation*, model menjadi lebih *robust* dan mampu melakukan generalisasi dengan lebih baik terhadap data yang belum pernah dilihat.

2.2.7 *Data Labeling*

Data labeling merupakan proses pemberian label atau anotasi pada data yang digunakan dalam *machine learning* dan *deep learning*. Label ini berfungsi sebagai acuan (*ground truth*) bagi model dalam proses pembelajaran, sehingga model dapat memahami hubungan antara input dan output yang diharapkan [43]. Kualitas *data labeling* sangat berpengaruh terhadap performa model, karena kesalahan dalam pemberian label dapat menyebabkan penurunan akurasi dan menghasilkan prediksi yang tidak tepat [44].

2.2.8 *Data Splitting*

Data splitting merupakan proses pembagian *dataset* menjadi beberapa bagian yang digunakan dalam tahapan pengembangan model *machine learning* dan *deep learning*. Pembagian ini bertujuan untuk memastikan bahwa model dapat dilatih, divalidasi, dan diuji secara objektif tanpa mengalami *overfitting*. Dengan melakukan *data splitting*, model tidak hanya belajar dari data yang sama, tetapi juga diuji kemampuannya dalam melakukan generalisasi terhadap data baru [45]. Secara umum, *dataset* dibagi menjadi tiga bagian utama, yaitu *training set*, *validation set*, dan *test set*. *Training set* digunakan untuk melatih model dan mempelajari pola dari data. *Validation set* digunakan untuk melakukan evaluasi sementara serta penyesuaian parameter model selama proses pelatihan. Sementara itu, *test set* digunakan untuk mengukur performa akhir model terhadap data yang belum pernah

dilihat sebelumnya. Proporsi pembagian data dapat bervariasi, namun umumnya menggunakan rasio seperti 70:15:15 atau 80:10:10. Rasio ini dipilih untuk memberikan porsi data yang lebih besar pada *training set* agar model dapat mempelajari pola secara optimal, sementara *validation set* dan *test set* tetap disediakan dalam jumlah yang cukup untuk melakukan evaluasi dan mengukur kemampuan generalisasi model secara objektif [46].

2.2.9 AdamW Optimizer

AdamW Optimizer merupakan pengembangan dari algoritma Adam (*Adaptive Moment Estimation*) yang menerapkan mekanisme *weight decay* secara terpisah dari proses pembaruan gradien [47]. Pendekatan ini memungkinkan regularisasi yang lebih efektif sehingga dapat meningkatkan kemampuan generalisasi model dan mengurangi risiko *overfitting*. AdamW banyak digunakan pada model berbasis *deep learning*, khususnya arsitektur Transformer seperti BERT dan IndoBERT, karena mampu menghasilkan proses pelatihan yang lebih stabil serta konvergensi yang lebih baik dibandingkan optimizer Adam konvensional [48].

2.2.10 Metrik Evaluasi

Metrik evaluasi merupakan ukuran yang digunakan untuk menilai performa model dalam menyelesaikan suatu tugas klasifikasi [49]. Dalam konteks *machine learning* dan *deep learning*, metrik evaluasi berperan penting untuk mengetahui seberapa baik model dalam memprediksi data secara akurat serta dalam membedakan antara kelas yang berbeda [50]. Beberapa metrik evaluasi yang umum digunakan dalam klasifikasi adalah sebagai berikut:

1) *Accuracy*

Accuracy merupakan ukuran yang menunjukkan proporsi prediksi yang benar dibandingkan dengan seluruh jumlah data. Nilai *accuracy* diperoleh dari perbandingan jumlah prediksi benar, yaitu *True Positive* (TP) dan *True Negative* (TN), terhadap total data yang terdiri dari *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) Seperti pada Rumus 2.1.

Oleh karena itu, *accuracy* dirumuskan sebagai (TP + TN) / (TP + TN + FP + FN).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.1)$$

Rumus 2.1 Rumus *Accuracy* [50]

Pada Rumus 2.1 *True Positive* (TP) menunjukkan jumlah data pada kelas positif yang berhasil diprediksi dengan benar, sedangkan *True Negative* (TN) menunjukkan jumlah data pada kelas negatif yang diprediksi dengan benar. *False Positive* (FP) merupakan jumlah data kelas negatif yang salah diprediksi sebagai kelas positif, sementara *False Negative* (FN) merupakan jumlah data kelas positif yang salah diprediksi sebagai kelas negatif.

2) *Precision*

Precision menunjukkan seberapa besar proporsi prediksi positif yang benar dibandingkan dengan seluruh prediksi positif yang dihasilkan oleh model. Nilai *precision* dihitung dengan membandingkan jumlah *True Positive* (TP) terhadap total prediksi positif yang terdiri dari *True Positive* (TP) dan *False Positive* (FP). *Precision* dirumuskan sebagai TP / (TP + FP) seperti pada Rumus 2.2.Metrik ini penting untuk mengurangi kesalahan klasifikasi positif (*false positive*), sehingga model tidak terlalu sering mengklasifikasikan data negatif sebagai positif.

$$Precision = \frac{TP}{TP+FP} \quad (2.2)$$

Rumus 2.2 Rumus *Precision* [50]

3) *Recall*

Recall menunjukkan kemampuan model dalam mendeteksi seluruh data positif yang sebenarnya. Nilai *recall* dihitung dengan membandingkan jumlah *True Positive* (TP) terhadap total data positif yang terdiri dari *True Positive* (TP) dan *False Negative* (FN). Oleh karena itu, *recall* dirumuskan pada Rumus 2.3.Metrik ini berfokus pada meminimalkan kesalahan negatif (*false negative*), sehingga model tidak melewatkan data positif yang seharusnya terdeteksi.

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

Rumus 2.3 Rumus *Recall* [50]

4) *F1-score*

F1-score merupakan rata-rata harmonis dari *precision* dan *recall* yang digunakan untuk menyeimbangkan kedua metrik tersebut. Nilai *F1-score* dihitung berdasarkan kombinasi antara *precision* dan *recall*, sehingga memberikan gambaran yang lebih seimbang terhadap kinerja model. Oleh karena itu, *F1-score* dirumuskan pada Rumus 2.4 Metrik ini sangat berguna terutama ketika terdapat ketidakseimbangan data antar kelas, karena mampu memberikan evaluasi yang lebih representatif dibandingkan hanya menggunakan salah satu metrik saja.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall} \quad (2.4)$$

Rumus 2.4 Rumus *F1-score* [50]

2.2.11 User Acceptance Testing (UAT)

User Acceptance Testing (*UAT*) merupakan metode pengujian perangkat lunak yang dilakukan oleh pengguna akhir (*end-user*) untuk memastikan bahwa sistem yang dikembangkan telah sesuai dengan kebutuhan pengguna dan dapat digunakan dengan baik dalam kondisi nyata [51]. UAT berfokus pada aspek fungsionalitas, kemudahan penggunaan (*usability*), efektivitas sistem, serta tingkat kepuasan pengguna terhadap aplikasi yang dibangun. Pengujian ini dilakukan setelah sistem selesai dikembangkan dan sebelum sistem diimplementasikan secara penuh kepada pengguna akhir [52]. Menurut Roger S. Pressman dalam buku *Software Engineering: A Practitioner's Approach*, UAT merupakan tahap validasi akhir yang bertujuan untuk memastikan bahwa perangkat lunak telah memenuhi kebutuhan bisnis dan harapan pengguna. Pengujian UAT dilakukan menggunakan skala Likert 1–5, dengan ketentuan nilai 1 menunjukkan sangat tidak setuju, nilai 2 menunjukkan tidak setuju, nilai 3 menunjukkan cukup, nilai 4 menunjukkan setuju, dan nilai 5 menunjukkan sangat setuju. Hasil penilaian kemudian dikonversi ke dalam bentuk persentase untuk mengetahui tingkat kelayakan dan penerimaan

sistem berdasarkan pengalaman pengguna secara langsung. Nilai hasil pengujian kemudian dikonversi ke dalam bentuk persentase untuk mengetahui tingkat penerimaan pengguna terhadap system [53]. Interpretasi persentase UAT dapat mengacu pada Tabel 2.2.

Tabel 2.2 Interpretasi persentase UAT [54]

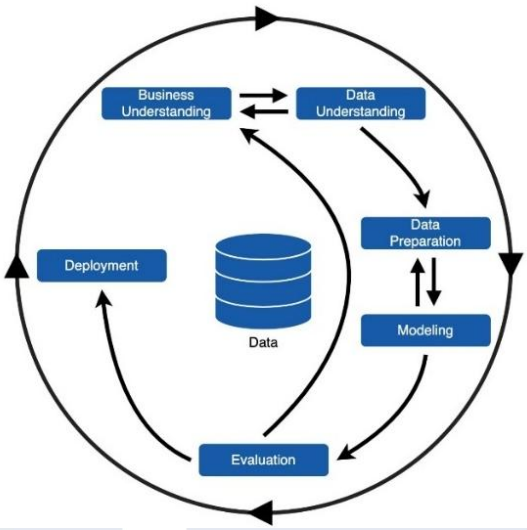
Persentase	Kategori
0% – 20%	Sangat Tidak Baik
21% – 40%	Tidak Baik
41% – 60%	Cukup
61% – 80%	Baik
81% – 100%	Sangat Baik

Pada Tabel 2.2 dijelaskan bahwa hasil persentase *User Acceptance Testing* (*UAT*) berada pada rentang 0%-100% yang dibagi ke dalam beberapa kategori penilaian. Persentase sebesar 81%-100% termasuk dalam kategori *sangat baik*, sehingga sistem dapat dikatakan memiliki tingkat penerimaan pengguna yang sangat baik apabila memperoleh nilai persentase di atas 81%. Semakin tinggi persentase yang diperoleh, maka semakin baik tingkat usability, efektivitas, dan kepuasan pengguna terhadap sistem yang dikembangkan [54].

2.3 *Framework*/Algoritma yang digunakan

2.3.1 *CRISP-DM (Cross Industry Standard Process for Data Mining)*

CRISP-DM (Cross Industry Standard Process for Data Mining) merupakan *framework* yang digunakan sebagai panduan dalam proses pengembangan proyek *data mining* dan *machine learning*. *Framework* ini menyediakan pendekatan sistematis yang membantu dalam mengelola alur kerja penelitian seperti pada Gambar 2.1, mulai dari pemahaman masalah hingga evaluasi hasil [55]. *CRISP-DM* banyak digunakan karena bersifat fleksibel dan dapat diterapkan pada berbagai jenis permasalahan, termasuk dalam penelitian *hoax detection*.



Gambar 2.1 Tahapan CRISP-DM

Sumber : [55]

Berdasarkan Gambar 2.1, CRISP-DM memiliki enam tahapan utama, yaitu *business understanding*, *data understanding*, *data preparation*, *modeling*, *evaluation*, dan *deployment* [56]. Setiap tahapan saling berhubungan dan dapat dilakukan secara iteratif sesuai dengan kebutuhan dalam proses pengembangan model. Berikut Adalah penjelasannya:

1. *Business Understanding*

   Business Understanding merupakan tahap awal dalam metodologi CRISP-DM yang bertujuan untuk memahami tujuan penelitian serta permasalahan yang ingin diselesaikan. Pada tahap ini dilakukan identifikasi kebutuhan, penentuan tujuan analisis, serta perumusan masalah yang relevan dengan konteks penelitian. Tahap ini juga mencakup penentuan strategi dan pendekatan yang akan digunakan untuk mencapai tujuan yang telah ditetapkan.

2. *Data understanding*

   Data understanding merupakan tahap dalam metodologi *CRISP-DM* yang berfokus pada pengumpulan dan eksplorasi data untuk memahami karakteristik *dataset* yang digunakan. Pada tahap ini dilakukan proses analisis awal terhadap data, seperti melihat distribusi, kualitas, serta potensi permasalahan yang terdapat dalam *dataset*. Tahap ini bertujuan untuk memastikan bahwa data yang

digunakan sesuai dengan kebutuhan penelitian dan dapat mendukung proses pemodelan secara optimal.

3. *Data Preparation*

   Data Preparation merupakan tahap dalam metodologi *CRISP-DM* yang berfokus pada pembersihan dan transformasi data agar siap digunakan dalam proses pemodelan. Pada tahap ini dilakukan berbagai proses *pre-processing*, seperti *cleaning*, *tokenization*, normalisasi, serta *data labeling* untuk memastikan kualitas data. Selain itu, tahap ini bertujuan untuk menghasilkan *dataset* yang terstruktur, konsisten, dan sesuai dengan kebutuhan model sehingga dapat meningkatkan performa pemodelan.

4. *Modeling*

   Modeling merupakan tahap dalam metodologi *CRISP-DM* yang berfokus pada pembangunan model menggunakan algoritma *machine learning* atau *deep learning*. Pada tahap ini dilakukan pemilihan algoritma, penentuan parameter, serta proses pelatihan model menggunakan data yang telah dipersiapkan. Selain itu, tahap ini bertujuan untuk menghasilkan model yang mampu mempelajari pola dari data dan memberikan prediksi yang akurat sesuai dengan tujuan penelitian.

5. *Evaluation*

   Evaluation merupakan tahap dalam metodologi CRISP-DM yang berfokus pada penilaian performa model untuk memastikan bahwa model yang dihasilkan telah sesuai dengan tujuan penelitian. Pada tahap ini dilakukan pengukuran kinerja model menggunakan berbagai metrik evaluasi, seperti *accuracy*, *precision*, *recall*, dan *F1-score*. Tahap ini bertujuan untuk memastikan bahwa model tidak hanya memiliki performa yang baik secara teknis, tetapi juga relevan dan dapat digunakan dalam konteks permasalahan yang diteliti.

6. *Deployment*

   Deployment merupakan tahap dalam metodologi CRISP-DM yang berfokus pada implementasi model ke dalam sistem atau aplikasi yang dapat digunakan secara nyata. Pada tahap ini, model yang telah melalui proses evaluasi akan diintegrasikan ke dalam lingkungan operasional sehingga dapat digunakan untuk menghasilkan prediksi secara langsung. Tahap ini juga mencakup proses

pemantauan dan pemeliharaan model guna memastikan kinerja tetap optimal seiring dengan perubahan data atau kebutuhan sistem.

2.3.2 BERT

BERT (*Bidirectional Encoder Representations from Transformers*) merupakan model *deep learning* berbasis arsitektur *Transformer* yang dikembangkan oleh Google pada tahun 2018 [57]. BERT dirancang untuk memahami konteks kata dalam suatu kalimat secara dua arah (*bidirectional*), sehingga model dapat memahami hubungan antar kata dengan lebih baik dibandingkan metode sebelumnya yang hanya membaca teks dari satu arah. Kemampuan tersebut membuat BERT banyak digunakan pada berbagai tugas *Natural Language Processing* (NLP), seperti klasifikasi teks, analisis sentimen, *question answering*, dan deteksi berita *hoax* [58].

Meskipun metode traditional machine learning seperti Naive Bayes dan SVM telah banyak digunakan dalam klasifikasi teks, metode tersebut umumnya memerlukan proses feature engineering secara manual. Sementara itu, model deep learning seperti CNN dan LSTM mampu mempelajari fitur secara otomatis, namun masih memiliki keterbatasan dalam menangkap hubungan kontekstual yang panjang dan memproses data secara berurutan. Untuk mengatasi keterbatasan tersebut, Transformer diperkenalkan dengan mekanisme self-attention yang memungkinkan model memahami konteks secara lebih efektif dan melakukan komputasi secara parallel [59].

Arsitektur BERT menggunakan bagian *encoder* pada model *Transformer* yang terdiri dari beberapa lapisan *multi-head self-attention* dan *feed forward neural network*. Mekanisme *self-attention* memungkinkan model memahami hubungan antar kata dalam suatu kalimat dengan mempertimbangkan konteks dari seluruh kalimat secara bersamaan. Selain itu, BERT menggunakan proses *token embedding*, *segment embedding*, dan *position embedding* untuk merepresentasikan input teks sebelum diproses pada *Transformer encoder*. Proses pelatihan BERT dilakukan menggunakan metode *Masked Language Modeling* (MLM) dan *Next Sentence Prediction* (NSP) untuk meningkatkan pemahaman konteks bahasa. Mekanisme utama pada BERT menggunakan *self-attention* yang berfungsi untuk menghitung

hubungan antar token dalam kalimat [60]. Perhitungan *self-attention* dapat dilihat pada Rumus 2.5

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.5)$$

Rumus 2.5 Rumus *Self-attention* [60]

Keterangan:

1. Q= *query*
2. K= *key*
3. V= *value*
4. d_k= dimensi vektor *key*

Pada Rumus 2.5 menunjukkan bahwa nilai *attention* diperoleh dari hasil perkalian antara *query* dan *key* yang kemudian dinormalisasi menggunakan fungsi *softmax*. Hasil tersebut selanjutnya dikalikan dengan *value* untuk menghasilkan representasi konteks dari setiap token. Proses kerja BERT dapat dilihat pada Tabel 2.3.

Tabel 2.3 *Pseudocode* Model BERT [61]

(1)	Melakukan tokenisasi pada teks input menggunakan tokenizer BERT
(2)	Mengubah token menjadi *embedding* dan menambahkan *positional embedding*
(3)	Membentuk matriks *Query* (Q), *Key* (K), dan *Value* (V)
(4)	Menghitung skor *self-attention* menggunakan perkalian QK^T
(5)	Melakukan normalisasi skor menggunakan fungsi *softmax*
(6)	Mengalikan hasil *softmax* dengan matriks V untuk menghasilkan representasi konteks
(7)	Memproses hasil *attention* pada *Transformer encoder*
(8)	Menghasilkan representasi akhir untuk proses klasifikasi teks

Pada proses kerjanya, BERT melakukan tokenisasi terhadap teks input menggunakan *tokenizer* BERT, kemudian setiap token diubah menjadi representasi *embedding* yang dikombinasikan dengan *positional embedding*. Selanjutnya, model membentuk matriks *Query* (Q), *Key* (K), dan *Value* (V) untuk menghitung mekanisme *self-attention*. Skor perhatian dihitung menggunakan perkalian QK^T dan dinormalisasi menggunakan fungsi *softmax*. Hasil tersebut kemudian dikalikan dengan matriks *Value* untuk menghasilkan representasi konteks yang selanjutnya diproses pada *Transformer encoder* [61]. Representasi akhir yang dihasilkan

digunakan untuk proses klasifikasi teks. Proses kerja BERT dapat dilihat pada Tabel 2.3.

2.3.3 mBERT

mBERT (*Multilingual Bidirectional Encoder Representations from Transformers*) merupakan pengembangan dari BERT yang dirancang untuk mendukung banyak bahasa dalam satu model *Transformer*. mBERT dikembangkan oleh Google dengan menggunakan korpus multibahasa dari berbagai bahasa yang terdapat pada *Wikipedia* [62]. Berbeda dengan BERT yang umumnya dilatih menggunakan satu bahasa, mBERT mampu memahami representasi teks dari berbagai bahasa secara bersamaan sehingga banyak digunakan pada tugas *Natural Language Processing* (NLP) multibahasa, seperti klasifikasi teks, analisis sentimen, dan deteksi *hoax* [13].

Arsitektur mBERT menggunakan *Transformer encoder* yang sama seperti BERT dengan mekanisme *multi-head self-attention* untuk memahami hubungan antar token dalam suatu kalimat. mBERT memanfaatkan proses tokenisasi berbasis *WordPiece* dan menghasilkan representasi kontekstual melalui proses *self-attention*. Mekanisme *attention* pada mBERT dapat dilihat pada Rumus (2.5). Tabel 2.4 merupakan *Pseudocode* dari proses model mBERT.

Tabel 2.4 *Pseudocode* model mBERT [63]

(1)	Melakukan tokenisasi pada teks input menggunakan tokenizer BERT
(2)	Mengubah token menjadi *embedding* dan menambahkan *positional embedding*
(3)	Membentuk matriks *Query* (Q), *Key* (K), dan *Value* (V)
(4)	Menghitung skor *self-attention* menggunakan perkalian QK^T
(5)	Melakukan normalisasi skor menggunakan fungsi *softmax*
(6)	Mengalikan hasil *softmax* dengan matriks V untuk menghasilkan representasi konteks
(7)	Memproses hasil *attention* pada *Transformer encoder* multilingual
(8)	Menghasilkan representasi akhir untuk proses klasifikasi teks

Pada proses kerjanya, *mBERT* melakukan tokenisasi pada teks input menggunakan *tokenizer BERT* kemudian mengubah token menjadi representasi *embedding* yang dikombinasikan dengan *positional embedding*. Selanjutnya model membentuk matriks *Query* (*Q*), *Key* (*K*), dan *Value* (*V*) untuk menghitung mekanisme *self-attention*. Skor perhatian dihitung menggunakan perkalian QK^T dan dinormalisasi menggunakan fungsi *softmax*. Hasil tersebut kemudian dikalikan

dengan matriks *Value* untuk menghasilkan representasi konteks yang selanjutnya diproses pada *Transformer encoder multilingual* [63]. Representasi akhir yang dihasilkan digunakan untuk proses klasifikasi teks. Proses model *mBERT* dapat dilihat pada Tabel 2.4.

2.3.4 IndoBERT

IndoBERT merupakan model *Bidirectional Encoder Representations from Transformers* (BERT) yang telah dilatih menggunakan korpus Bahasa Indonesia sehingga mampu memahami konteks bahasa Indonesia dengan lebih baik. IndoBERT dikembangkan untuk mendukung berbagai tugas *Natural Language Processing* (NLP) berbahasa Indonesia, seperti klasifikasi teks, analisis sentimen, *question answering*, dan deteksi berita *hoax* [64]. Arsitektur IndoBERT menggunakan *Transformer encoder* yang sama seperti BERT dengan mekanisme *multi-head self-attention* untuk memahami hubungan antar token dalam suatu kalimat. IndoBERT memanfaatkan proses tokenisasi berbasis *WordPiece* dan menghasilkan representasi kontekstual melalui proses *self-attention* [65]. Mekanisme *attention* pada IndoBERT dapat dilihat pada Rumus (2.5). Tabel 2.5 merupakan *Pseudocode* dari proses model IndoBERT.

Tabel 2.5 *Pseudocode* Model Indobert [66]

(1)	Melakukan tokenisasi pada teks input menggunakan tokenizer BERT
(2)	Mengubah token menjadi *embedding* dan menambahkan *positional embedding*
(3)	Membentuk matriks *Query* (Q), *Key* (K), dan *Value* (V)
(4)	Menghitung skor *self-attention* menggunakan perkalian QK^T
(5)	Melakukan normalisasi skor menggunakan fungsi *softmax*
(6)	Mengalikan hasil *softmax* dengan matriks V untuk menghasilkan representasi konteks
(7)	Memproses hasil *attention* pada *Transformer encoder* IndoBERT
(8)	Menghasilkan representasi akhir untuk proses klasifikasi teks

Selanjutnya dilakukan perhitungan skor perhatian melalui operasi perkalian antara matriks *Query* dan *Key* (QK^T) yang kemudian dinormalisasi menggunakan fungsi *softmax*. Hasil normalisasi tersebut dikombinasikan dengan matriks *Value* untuk menghasilkan representasi konteks teks yang lebih mendalam. Representasi tersebut kemudian diproses pada *Transformer encoder* khusus IndoBERT sehingga model dapat memahami karakteristik bahasa Indonesia secara lebih baik [66]. Pada tahap akhir, model menghasilkan representasi fitur yang digunakan untuk proses klasifikasi teks. Proses kerja model IndoBERT dapat dilihat pada Tabel 2.5.

2.3.5 CNN

Convolutional Neural Network (CNN) merupakan salah satu metode *deep learning* yang banyak digunakan pada bidang *computer vision* untuk melakukan pengolahan dan klasifikasi citra. CNN dirancang untuk mengekstraksi fitur visual secara otomatis melalui proses konvolusi sehingga mampu mengenali pola seperti tepi, tekstur, bentuk, maupun objek pada gambar. Dibandingkan metode tradisional, CNN memiliki kemampuan yang lebih baik dalam mempelajari representasi fitur dari data citra karena proses ekstraksi fitur dilakukan secara otomatis oleh jaringan [67].

Arsitektur CNN terdiri dari beberapa komponen utama, yaitu *convolution layer*, fungsi aktivasi, *pooling layer*, dan *fully connected layer*. *Convolution layer* berfungsi untuk mengekstraksi fitur pada citra menggunakan kernel atau filter tertentu. Hasil dari proses konvolusi berupa *Feature map* yang kemudian diproses menggunakan fungsi aktivasi seperti ReLU (*Rectified Linear Unit*) untuk menambahkan non-linearitas pada model. Setelah itu dilakukan proses *pooling* untuk mengurangi dimensi data dan mempertahankan fitur penting sebelum diproses pada *fully connected layer* untuk menghasilkan output klasifikasi Operasi utama pada CNN adalah proses konvolusi yang digunakan untuk mengekstraksi fitur visual dari citra. Operasi konvolusi dapat dirumuskan sebagai Rumus 2.6.

$$(I * K)(x, y) = \sum_m \sum_n I\,(x - m, y - n) K(m, n) \tag{2.6}$$

Rumus 2.6 Rumus Operasi Konvolusi [68]

Keterangan:

1. I= citra input
2. K= kernel/filter
3. x, y= posisi piksel pada citra

Pada Rumus 2.6 menunjukkan bahwa kernel atau filter akan digeser pada citra input untuk menghasilkan *Feature map* yang merepresentasikan pola tertentu pada gambar. Setelah proses konvolusi, hasil *Feature map* diproses menggunakan fungsi aktivasi ReLU yang dapat dirumuskan sebagai Rumus 2.7.

$$\text{ReLU}(x) = \max(0, x) \quad (2.7)$$

Rumus 2.7 Aktivasi ReLU [69]

Keterangan: x= input neuron

Fungsi aktivasi ReLU pada Rumus 2.7 digunakan untuk mengubah nilai negatif menjadi nol dan mempertahankan nilai positif sehingga membantu model mempelajari pola non-linear pada data. Proses kerja *Convolutional Neural Network* (CNN) dapat dilihat pada Tabel 2.6.

Tabel 2.6 Pseudocode Convolutional Neural Network [70]

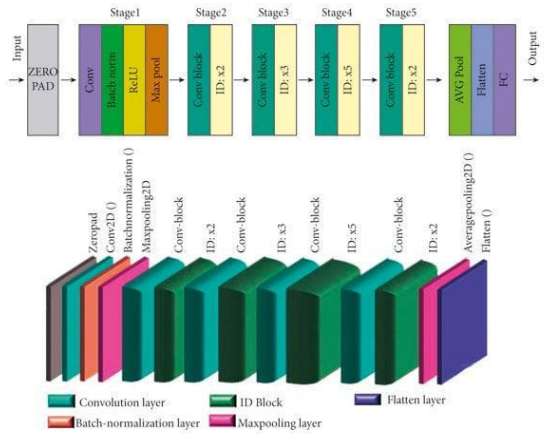
(1)	Menerima input berupa citra/gambar
(2)	Melakukan operasi konvolusi menggunakan kernel/filter
(3)	Menghasilkan *Feature map* dari hasil konvolusi
(4)	Menerapkan fungsi aktivasi ReLU
(5)	Melakukan proses *pooling* untuk mengurangi dimensi *Feature map*
(6)	Mengulangi proses konvolusi dan *pooling* pada beberapa layer
(7)	Melakukan proses *flatten* pada *Feature map*
(8)	Memproses hasil *flatten* menggunakan *fully connected layer*
(9)	Menghasilkan output klasifikasi

Pada proses kerjanya, CNN menerima input berupa citra atau gambar yang kemudian diproses menggunakan operasi konvolusi dengan kernel atau filter tertentu untuk menghasilkan *feature map*. Hasil konvolusi tersebut selanjutnya diproses menggunakan fungsi aktivasi *ReLU* untuk meningkatkan kemampuan model dalam mempelajari pola non-linear. Setelah itu dilakukan proses *pooling* untuk mengurangi dimensi *feature map* sehingga jumlah parameter menjadi lebih efisien dan risiko *overfitting* dapat dikurangi. Proses konvolusi dan *pooling* dapat dilakukan berulang pada beberapa lapisan untuk mengekstraksi fitur visual yang lebih mendalam. Hasil *feature map* diubah menjadi bentuk vektor satu dimensi melalui proses *flatten* dan diproses menggunakan *fully connected layer* untuk menghasilkan output klasifikasi [70]. Proses kerja CNN dapat dilihat pada Tabel 2.6.

2.3.6 ResNet-50

ResNet-50 (*Residual Network-50*) merupakan salah satu arsitektur *Convolutional Neural Network* (CNN) yang terdiri dari 50 lapisan dan dirancang untuk mengatasi permasalahan *vanishing gradient* pada jaringan yang sangat dalam

(*deep neural network*). Arsitektur ini diperkenalkan dengan konsep *Residual learning*, yang memungkinkan model untuk mempelajari perbedaan (*Residual*) antara input dan output [71]. Dengan pendekatan tersebut, proses pelatihan menjadi lebih stabil dan efektif karena informasi dapat mengalir lebih baik melalui jaringan yang dalam.



Gambar 2.2 Arsitektur ResNet-50

Sumber : [72]

Gambar 2.3 menunjukkan arsitektur ResNet-50 (*Residual Network-50*) yang terdiri dari beberapa blok *Residual* (*Residual block*) dengan mekanisme *skip connection*. Arsitektur ini menggunakan kombinasi lapisan konvolusi berukuran 1×1, 3×3, dan 1×1 dalam setiap blok untuk mengekstraksi fitur dari citra. Setiap *Residual block* memungkinkan input dilewatkan langsung ke lapisan berikutnya melalui *shortcut connection*, sehingga membantu mengatasi permasalahan *vanishing gradient*. Mekanisme ini memungkinkan model yang sangat dalam tetap dapat dilatih secara efektif. Pada dasarnya, ResNet-50 merupakan pengembangan dari *Convolutional Neural Network* (CNN) yang bekerja dengan mengekstraksi fitur dari citra melalui operasi konvolusi Keunggulan utama ResNet-50 juga terletak pada penggunaan *Residual connection* atau *skip connection*, yang memungkinkan input dari suatu lapisan dilewatkan langsung ke lapisan berikutnya. Konsep ini dapat dirumuskan pada Rumus 2.8:

$$y = F(x) + x \tag{2.8}$$

Rumus 2.8 *Residual Connection* [73]

Pada rumus 2.8 $F(x)$ adalah fungsi transformasi yang dipelajari oleh jaringan, dan x adalah input awal. Dengan pendekatan ini, model tidak perlu mempelajari fungsi secara keseluruhan, tetapi cukup mempelajari *Residual*-nya, sehingga dapat mengurangi degradasi performa pada jaringan yang dalam. Setiap *Residual block* dalam ResNet-50 umumnya terdiri dari lapisan konvolusi, *batch normalization*, dan fungsi aktivasi seperti *ReLU* (*Rectified Linear Unit)*. ResNet-50 dimulai dengan menerima input berupa citra yang kemudian diproses menggunakan operasi konvolusi untuk mengekstraksi fitur visual seperti pada Tabel 2.7.

Tabel 2.7 *Pseudocode* ResNet-50 [74]

(1)	Menerima input berupa citra/gambar
(2)	Melakukan operasi konvolusi menggunakan kernel/filter
(3)	Melakukan proses *batch normalization*
(4)	Menerapkan fungsi aktivasi ReLU
(5)	Memproses *Feature* map pada *Residual* block
(6)	Menambahkan *skip connection* antara input dan output *Residual* block
(7)	Mengulangi proses *Residual* block hingga seluruh layer selesai diproses
(8)	Melakukan *global average pooling*
(9)	Menghasilkan representasi fitur untuk proses klasifikasi

Hasil konvolusi selanjutnya diproses menggunakan *batch normalization* dan fungsi aktivasi ReLU untuk meningkatkan kemampuan pembelajaran model. Pada setiap *Residual block* diterapkan mekanisme *skip connection* yang memungkinkan informasi dari layer sebelumnya diteruskan langsung ke layer berikutnya. Mekanisme tersebut membantu mengatasi permasalahan *vanishing gradient* pada jaringan yang dalam. Hasil akhir dari proses ekstraksi fitur kemudian digunakan pada proses klasifikasi gambar. Proses kerja ResNet-50 dapat dilihat pada Tabel 2.7.

2.3.7 *Optical Character Recognition* (OCR)

Optical Character Recognition (OCR) merupakan teknologi yang digunakan untuk mengubah teks yang terdapat dalam gambar atau dokumen visual menjadi teks *digital* yang dapat diproses oleh sistem komputer. OCR banyak digunakan dalam berbagai aplikasi seperti digitalisasi dokumen, pembacaan teks pada gambar, serta ekstraksi informasi dari media visual [75]. Secara umum, proses kerja OCR terdiri dari beberapa tahapan. Tahap pertama adalah *image pre-processing*, yang

bertujuan untuk meningkatkan kualitas gambar melalui proses seperti *grayscale conversion*, *binarization*, dan penghilangan *noise*. Tahap berikutnya adalah *text detection*, yaitu proses untuk mendeteksi area dalam gambar yang mengandung teks [76]. Secara matematis, proses pengenalan karakter dapat dipandang sebagai fungsi klasifikasi dalam Rumus 2.9:

$$y = f(x) \tag{2.9}$$

Rumus 2.9 Fungsi Klasifikasi [77]

Pada Rumus 2.9, *x* merupakan citra karakter sebagai input dan *y* merupakan hasil prediksi berupa karakter atau teks. Fungsi *f(x)* menggambarkan proses pemetaan dari citra input menjadi keluaran yang dapat dikenali oleh sistem. Pada pendekatan modern, OCR sering dikombinasikan dengan model sekuensial untuk mengenali rangkaian karakter dalam satu waktu, sehingga mampu meningkatkan akurasi dalam pengenalan teks yang kompleks [77].

2.3.8 *Feature Fusion*

Feature fusion merupakan teknik dalam *multimodal learning* yang digunakan untuk menggabungkan fitur dari berbagai sumber data atau modalitas, seperti teks dan gambar, menjadi satu representasi terpadu. Tujuan utama dari *Feature fusion* adalah untuk memanfaatkan informasi yang saling melengkapi (*complementary information*) dari masing-masing modalitas sehingga model dapat memperoleh pemahaman yang lebih komprehensif terhadap data. Dengan menggabungkan berbagai jenis fitur, model diharapkan mampu meningkatkan akurasi dalam proses klasifikasi [78].

Secara umum, terdapat beberapa jenis *Feature fusion*, yaitu *early fusion*, *late fusion*, dan *hybrid fusion*. *Early fusion* (*Feature-level fusion*) merupakan metode penggabungan fitur yang dilakukan sebelum proses klasifikasi, di mana fitur dari masing-masing modalitas digabungkan menjadi satu vektor sebagai input model. *Late fusion* (*decision-level fusion*) dilakukan setelah masing-masing modalitas menghasilkan prediksi, kemudian hasil tersebut digabungkan untuk menghasilkan keputusan akhir. Sementara itu, *hybrid fusion* merupakan kombinasi dari kedua pendekatan tersebut untuk memanfaatkan keunggulan masing-masing metode [79].

Salah satu metode yang umum digunakan dalam *Feature fusion* adalah *concatenation*, yaitu penggabungan fitur dengan cara menggabungkan dua vektor fitur menjadi satu vektor baru seperti pada Rumus 2.10.

$$F_{fusion} = [F_{text}; F_{image}] \tag{2.10}$$

Rumus 2.10 *Feature Concatenation* [79]

Pada Rumus 2.10, F_{text} merupakan representasi fitur teks yang dihasilkan dari model pemrosesan bahasa seperti IndoBERT, sedangkan F_{image}merupakan representasi fitur gambar yang dihasilkan dari model ekstraksi visual seperti *ResNet-50*. Kedua representasi fitur tersebut berasal dari modalitas yang berbeda, yaitu modalitas tekstual dan visual. Proses *feature fusion* dilakukan dengan menggabungkan kedua fitur tersebut menjadi satu representasi baru yang lebih komprehensif sehingga model dapat memahami hubungan antara informasi teks dan gambar secara bersamaan. Penggabungan fitur ini bertujuan untuk meningkatkan kemampuan model dalam menangkap konteks informasi yang tidak dapat diperoleh apabila hanya menggunakan satu modalitas saja. Melalui proses *feature fusion*, informasi tekstual seperti kata, kalimat, maupun konteks bahasa dapat dikombinasikan dengan informasi visual seperti pola gambar, bentuk objek, warna, dan struktur visual lainnya. Representasi gabungan tersebut diharapkan mampu memberikan pemahaman yang lebih mendalam terhadap data *multimodal* sehingga performa klasifikasi dapat meningkat dibandingkan pendekatan *unimodal* [80]. Hasil penggabungan fitur kemudian digunakan dalam proses klasifikasi yang dapat dinyatakan pada Rumus 2.11

$$y = \sigma(WF_{fusion} + b) \tag{2.11}$$

Rumus 2.11 Fungsi Klasifikasi *MultimodaI* [80]

Pada Rumus 2.11, *W* merupakan bobot, *b* merupakan bias, dan σ merupakan fungsi aktivasi seperti *sigmoid* atau *softmax*. Fungsi ini digunakan untuk menghasilkan output berupa probabilitas kelas berdasarkan fitur gabungan yang telah diperoleh.

2.4 Tools/software yang digunakan

2.4.1 Python

Python merupakan bahasa pemrograman tingkat tinggi yang banyak digunakan dalam pengembangan *machine learning* dan *deep learning*. Bahasa ini memiliki sintaks yang sederhana dan mudah dipahami, serta didukung oleh berbagai pustaka (*library*) seperti *Pandas*, *NumPy*, *PyTorch*, dan *Transformers* yang memudahkan proses pengolahan data, pembangunan model, hingga evaluasi [81]. Python memiliki komunitas yang besar dan dokumentasi yang lengkap sehingga mendukung pengembangan sistem secara efisien. Dalam penelitian ini, Python digunakan sebagai bahasa utama untuk mengimplementasikan seluruh tahapan, mulai dari *data pre-processing*, pembangunan dan pelatihan model, evaluasi kinerja model, hingga tahap *deployment*. Dengan dukungan berbagai *library* tersebut, proses pengembangan model menjadi lebih terstruktur, fleksibel, dan mudah untuk dikembangkan lebih lanjut [56].

2.1.1 *Google Colab*

Google Colab merupakan platform berbasis *cloud* yang memungkinkan pengguna menjalankan kode Python secara langsung melalui *browser* tanpa perlu melakukan instalasi di perangkat lokal. Platform ini menyediakan dukungan *GPU* dan *TPU* yang sangat membantu dalam mempercepat proses pelatihan model *deep learning* [82]. Selain itu, *Google Colab* juga terintegrasi dengan *Google Drive* sehingga memudahkan penyimpanan dan pengelolaan *dataset*. *Google Colab* digunakan sebagai lingkungan pengembangan utama untuk melakukan eksperimen, pelatihan model, serta pengolahan *dataset* dalam skala besar. Dengan fasilitas yang tersedia, proses komputasi menjadi lebih efisien dan fleksibel tanpa bergantung pada spesifikasi perangkat local [83].

2.1.2 *Visual Studio Code*

Visual Studio Code merupakan *code editor* yang ringan namun memiliki fitur lengkap untuk pengembangan perangkat lunak. Aplikasi ini mendukung berbagai ekstensi (*extensions*) yang memudahkan proses penulisan kode, *debugging*, serta pengelolaan proyek secara efisien [84]. Selain itu, Visual Studio Code juga menyediakan integrasi dengan sistem *version control* seperti *Git* yang membantu dalam manajemen kode. Visual Studio Code digunakan sebagai lingkungan

pengembangan lokal untuk menulis, mengelola, dan mengorganisasi kode program. Dengan fitur yang fleksibel dan dukungan ekstensi yang luas, proses pengembangan menjadi lebih terstruktur dan mudah untuk dikembangkan [85].

2.1.3 Streamlit

Streamlit merupakan *framework* berbasis Python yang digunakan untuk membangun aplikasi *web* interaktif dengan mudah. *Framework* ini memungkinkan pembuatan antarmuka pengguna (*user interface*) serta visualisasi hasil model secara cepat tanpa memerlukan pengetahuan mendalam tentang pengembangan *web* [86]. Selain itu, Streamlit mendukung integrasi dengan berbagai pustaka *machine learning* dan *deep learning*, sehingga memudahkan proses implementasi model ke dalam aplikasi. Dalam penelitian ini, Streamlit digunakan untuk membangun aplikasi deteksi *hoax* yang dapat digunakan secara langsung oleh pengguna. Aplikasi yang dikembangkan mampu menerima input berupa data teks atau gambar, kemudian memprosesnya menggunakan model yang telah dilatih, serta menampilkan hasil prediksi secara interaktif. Dengan demikian, tahap *deployment* dapat dilakukan secara lebih sederhana dan efisien [64].