BAB II
LANDASAN TEORI

2.1 Penelitian Terdahulu

Tabel 2.1 menyajikan ringkasan penelitian terdahulu yang relevan dengan topik penelitian ini. Penelitian-penelitian tersebut mencakup pendeteksian penyakit pada berbagai jenis tanaman serta pendeteksian penyakit yang secara spesifik menyerang tanaman pisang. Kajian terhadap penelitian terdahulu ini digunakan sebagai landasan untuk mengidentifikasi perkembangan penelitian, membandingkan pendekatan yang telah digunakan, serta menentukan kontribusi penelitian yang dilakukan.

Tabel 2.1 Penelitian Terdahulu

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
[25]	11 kelas dari *dataset* PlantVillage yang terkait dengan buah tomat. Total data adalah 7.700 gambar. Augmentasi data yang dilakukan termasuk rotasi, *flip*, *shear*, *zoom*, dan *shift*, serta	Model EfficientNetB0 yang dilatih mencapai skor akurasi 91,4%.	Tidak ada.	Rata-rata kecepatan inferensi pada telepon genggam Realme GT Neo2 adalah 149,2 ms, namun tidak diperjelas bagaimana kecepatan inferensi ini diukur. Tidak diperjelas arti 149,2 ms dalam konteks telepon genggam.	*Optimizer* Adam *Learning rate* 0,001 *Dropout rate* 0,5 Tidak ada *hyperparameter tuning* yang sistematis.	Model di-*deploy* pada sebuah prototipe aplikasi Android. Fitur meliputi deteksi *real-time* dan deteksi menggunakan gambar statis.	-

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
	perubahan kecerahan gambar.						
[39]	11 kelas gambar nematoda yang ditemukan pada tanaman di Indonesia. Total data adalah 957 gambar. Augmentasi data yang dilakukan termasuk mengubah kecerahan dan kontras gambar, menambahkan *noise*, dan menambahkan *blur*.	Metrik yang digunakan adalah skor akurasi. Tiga model dengan kinerja terbaik adalah: 1. EfficientNetV2B0: 97,94% 2. EfficientNetV2M: 97,94% 3. CoAtNet-0: 96,91%	Tidak ada.	Tidak diukur.	*Hyperparameter* yang terbaik diuji secara sistematis, namun masih berupa *grid search* untuk *optimizer*. *Hyperparameter* numerik statis dan tidak dioptimasi. *Optimizer* Adam, SGD, RMSprop *Learning rate* 0,001 untuk Adam dan RMSprop, 0,01 untuk SGD	Model di-*deploy* pada sebuah aplikasi berbasis web. Fitur meliputi deteksi menggunakan gambar statis.	-
[20]	4 kelas gambar tanaman pisang. Total data adalah 937 gambar.	Metrik yang digunakan adalah skor akurasi. 1. BananaSqueezeNet (arsitektur CNN yang dibuat pada penelitian ini,	Generalisasi model diuji pada *dataset* terpisah, namun model sudah	Waktu inferensi model diukur dan dibandingkan, namun tidak dijelaskan bagaimana dan di	*Hyperparameter* yang terbaik diuji secara sistematis melalui *Bayesian optimization*.	Model di-*deploy* pada sebuah aplikasi berbasis web. Fitur meliputi deteksi	Penelitian ini mengembangkan arsitektur CNN bernama BananaSqueezeNet yang bertujuan untuk mengidentifikasi penyakit

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
		berbasis SqueezeNet): 96,25% 2. VGG16: 95% 3. InceptionV3: 90% 4. ResNet101: 90% 5. EfficientNetB0: 87,5% 6. MobileNetV3: 86,25% 7. ResNet50: 86,25%	diberikan kesempatan terlebih dahulu untuk beradaptasi ke distribusi *dataset* yang baru sehingga belum terlihat kinerja prediksi model ketika menghadapi data OOD. Tidak ada pengujian *robustness* terhadap korupsi gambar.	mana waktu inferensi ini dukur.	*Optimizer* Adadelta, Adam, RMSprop, SGD *Learning rate* 0,00001-0,01.	menggunakan gambar statis.	tanaman pisang dengan kompleksitas model yang sederhana. Model ini menunjukkan kinerja yang lebih baik daripada tiga arsitektur populer lainnya. Model dengan kompleksitas yang lebih sederhana (parameter yang lebih sedikit) dapat menunjukkan kinerja yang mirip dengan model dengan parameter yang lebih banyak. Penelitian ini menekankan kebutuhan akan model CNN yang lebih sederhana untuk diimplementasikan dalam alat komputasi sederhana seperti Raspberry Pi 4. Penelitian ini juga mengimplementasikan Grad-CAM.
[21]	3 kelas gambar tanaman pisang. Total data adalah 16.092 gambar. Augmentasi	Metrik yang digunakan adalah skor akurasi. 1. SqueezeNet: 97,12% 2. ShuffleNetV2: 96,39%	Tidak ada.	Ukuran file model, waktu pelatihan model dan jumlah parameter dalam model dievaluasi, namun tidak ada	*Optimizer* Adam, RMSprop *Learning rate* 0,00005-0,001 Tidak ada *hyperparameter*	Tidak ada.	-

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
	data dilakukan secara geometris, tetapi tidak disebutkan secara spesifik. Total data setelah augmentasi menjadi 17.149 gambar.	3. MobileNetV3S: 93,59% 4. MobileNetV2: 73%		evaluasi efisiensi pada telepon genggam asli. MobileNetV3S memiliki waktu pelatihan paling pendek dari semua model dan menunjukkan skor akurasi yang lebih tinggi daripada MobileNetV2 yang membutuhkan sekitar 2x lebih waktu pelatihan. SqueezeNet menunjukkan skor akurasi terbaik dari semua model. MobileNetV3S memiliki jumlah parameter terbanyak dengan ukuran model terbesar.	*tuning* yang sistematis.		
[22]	4 kelas gambar tanaman pisang. Total data adalah 27.767 gambar.	Metrik yang digunakan adalah skor akurasi. Model yang dibuat mencapai skor akurasi 91,17%.	Tidak ada.	Tidak diukur.	*Optimizer* Adam *Learning rate* 0,001 Tidak ada *hyperparameter tuning* yang sistematis.	Model di-*deploy* pada sebuah prototipe aplikasi Android. Fitur meliputi deteksi	Penelitian ini membangun arsitektur CNN yang terdiri dari 4 lapisan konvolusi dengan fungsi *activation* ReLU serta 4 lapisan *pooling* dan *dropout*.

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
						menggunakan gambar statis dan foto baru dari kamera. Persentase keyakinan model terhadap suatu kelas ditunjukkan. Aplikasi memberi tahu informasi tambahan mengenai penyakit seperti cara mitigasi.	
[23]	4 kelas gambar tanaman pisang. Total data adalah 937 gambar.	Metrik yang digunakan adalah skor akurasi. Model yang dibuat mencapai skor akurasi 39%.	Tidak ada.	Tidak diukur.	Tidak ada.	Model di-*deploy* pada sebuah prototipe aplikasi Android. Fitur meliputi deteksi menggunakan gambar statis.	Penelitian ini membangun arsitektur CNN yang terdiri dari 6 lapisan konvolusi dengan fungsi *activation* ReLU serta 6 lapisan *pooling*.
[19]	3 kelas gambar tanaman pisang. Total data adalah	Metrik yang digunakan adalah skor akurasi. 1. ResNet50: 88,90%	Tidak ada.	Tidak diukur.	*Optimizer* Adam *Learning rate* 0,001	Model di-*deploy* pada sebuah prototipe	-

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
	900 gambar. Augmentasi yang dilakukan termasuk rotasi, *translation*, *crop*, *zoom, flip*, dan perubahan kecerahan. Total data setelah augmentasi menjadi 9.000 gambar.	2. EfficientNetB0: 88,33% 3. VGG19: 87,22%			Tidak ada *hyperparameter tuning* yang sistematis.	aplikasi Android. Fitur meliputi deteksi menggunakan foto baru dari kamera. Aplikasi memberi tahu informasi tambahan mengenai penyakit seperti cara mitigasi.	
[24]	3 kelas gambar tanaman pisang. Total data dan augmentasi tidak diberi tahu.	Metrik yang digunakan adalah skor F1. 1. DenseNet121: 91,4% 2. MobileNetV2: 89,9% 3. Xception: 89,9% 4. InceptionV3: 88,5%	Tidak ada.	Waktu pelatihan model dan jumlah parameter model diukur. Namun, tidak ada pengujian efisiensi pada telepon genggam asli. InceptionV3 mencapai akurasi terendah, namun merupakan model dengan jumlah parameter terbanyak dan memakan waktu	*Learning rate* 0,0001-0,001 *Dropout rate* 0,2-0,25 Tidak ada *hyperparameter tuning* yang sistematis.	Tidak ada.	-

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
				paling lama untuk dilatih. Sebaliknya, MobileNetV2 merupakan model dengan jumlah parameter terendah dan paling cepat untuk dilatih dan pada akhirnya mencapai akurasi yang kompetitif.			
[40]	10 kelas gambar tanaman beras. Total data adalah 10.080 gambar. Augmentasi data yang dilakukan termasuk *translation* dan *shear*.	Metrik yang digunakan adalah skor akurasi. 1. InceptionV3: 99,64% 2. InceptionresnetV2: 99,62% 3. DenseNet201: 99,61% 4. Xception: 99,61% 5. EfficientNetB0: 99,56% 6. ResNet101: 99,55% 7. MobileNetV2: 99,53% 8. NASNetMobile: 99,48% 9. ResNet50: 99,46%	Tidak ada.	Tidak diukur.	*Optimizer* Adam *Learning rate* 0,0001 Tidak ada *hyperparameter tuning* yang sistematis.	Tidak ada.	Penelitian ini melakukan komparasi antar 15 arsitektur CNN dan menemukan bahwa secara umum, InceptionV3 memiliki kinerja yang terbaik dalam memprediksi kelas gambar tanaman beras. Untuk arsitektur yang didesain dengan beban komputasi rendah, arsitektur EfficientNetB0 memiliki kinerja yang terbaik, diikuti dengan MobileNetV2 dan NASNetMobile.

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
		10. ShuffleNet: 99,41% 11. GoogleNet: 99,40% 12. Darknet-53: 99,35% 13. SqueezeNet: 99,09% 14. VGG16: 98,47% 15. AlexNet: 97,35%					
[26]	4 kelas gambar tanaman anggur. Total data adalah 9.027 gambar. Augmentasi data yang dilakukan termasuk rotasi, *flip*, *zoom*, dan *shift*. Total data setelah augmentasi adalah 27.122 gambar.	Metrik yang digunakan adalah skor akurasi. 1. MobileNetV3L: 99,42% 2. DenseNet121: 99,1% 3. EfficientNetV2B2: 98,78% 4. EfficientNetV2B1: 98,72% 5. MobileNetV3S: 98,65% 6. NASNetMobile: 97,09%	Tidak ada.	Waktu inferensi untuk setiap model diukur dalam satuan detik untuk memprediksi 500 gambar. 1. MobileNetV3S: 14,6 detik 2. MobileNetV3L: 45,6 detik 3. NASNetMobile: 133,9 detik 4. EfficientNetV2B1: 228,7 detik 5. EfficientNetV2B2: 236,7 detik 6. DenseNet121: 342,5 detik	*Optimizer* Adam *Learning rate* 0,0001 *Dropout rate* 0,45 Tidak ada *hyperparameter tuning* yang sistematis.	Tidak ada.	Penelitian ini melakukan komparasi antar 6 arsitektur CNN dan menemukan bahwa MobileNetV3L memiliki kinerja terbaik dalam memprediksi kelas gambar tanaman anggur. Untuk arsitektur yang didesain dengan beban komputasi rendah, arsitektur yang terbaik setelah itu adalah EfficientNetV2B2 dan EfficientNetV2B1. Walaupun DenseNet121 adalah arsitektur yang lebih kompleks, tetapi tidak mencapai skor

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
							akurasi tertinggi dan arsitektur lainnya dengan jumlah parameter yang lebih sedikit mencapai skor akurasi yang mirip atau lebih baik daripada DenseNet121. Penelitian ini juga mengimplementasikan Grad-CAM untuk mengevaluasi model secara kualitatif dan menemukan bahwa MobileNetV3L, DenseNet121, dan EfficientNetV2B1 adalah yang paling baik dalam menemukan fitur yang relevan untuk setiap kelas. MobileNetV3S memiliki latensi inferensi paling cepat, diikuti dengan MobileNetV3L dan NASNetMobile.
[41]	4 kelas gambar tanaman jagung. Total data adalah 4.188 gambar. Augmentasi data yang	Model ResNet152 yang dilatih mencapai skor akurasi 98,34%.	Tidak ada.	Tidak diukur.	*Optimizer* Adam *Learning rate* 0,0001 Tidak ada *hyperparameter*	Tidak ada.	Penelitian ini membandingkan kinerja model ResNet152 yang dilatih terhadap beberapa arsitektur CNN lainnya, termasuk MobileNetV3 dan EfficientNet yang

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
	dilakukan termasuk rotasi, *flip*, dan *scaling*.				*tuning* yang sistematis.		memiliki skor akurasi yang mirip dengan model pada penelitian ini (~98%). Penelitian ini juga memasangkan Grad-CAM pada model yang dilatih untuk verifikasi kemampuan model dalam melokalisasi fitur penyakit.
[27]	4 kelas gambar tanaman anggur. Total data adalah 9.027 gambar. Augmentasi data yang dilakukan termasuk rotasi, *flip*, *zoom*, dan perubahan kontras.	Model EfficientNetV2L yang dilatih mencapai skor akurasi 98%.	Tidak ada.	Tidak diukur.	*Optimizer* Adam *Learning rate* 0,00001 Tidak ada *hyperparameter tuning* yang sistematis.	Tidak ada.	Penelitian ini mengimplementasikan Grad-CAM pada model yang dibuat untuk verifikasi kemampuan model dalam melokalisasi fitur penyakit.
[42]	14 kelas gambar tanaman jeruk. Total data adalah 7.516 gambar.	Model YOLOv5L yang dilatih mencapai skor F1 mikro 85,19%.	Tidak ada.	Tidak diukur.	*Learning rate* 0,001 Tidak ada *hyperparameter tuning* yang sistematis.	Model di-*deploy* pada sebuah prototipe aplikasi Android. Fitur aplikasi meliputi	-

Ref.	*Dataset*	Hasil	Uji *Robustness* atau *Generalisasi*	Evaluasi Efisiensi	Seleksi *Hyperparameter*	*Deployment*	Catatan
						deteksi menggunakan gambar statis.	
[43]	3 kelas gambar tanaman jagung. Total data adalah 2.675 gambar. Augmentasi yang dilakukan termasuk rotasi, *flip*, *zoom*, dan *crop*.	Metrik yang digunakan adalah *mean average precision* (mAP): 1. YOLOv8n: 99,04% 2. YOLOv4: 97,50% 3. YOLOv7s: 93,30% 4. YOLOv5s: 88,23% 5. YOLOv3-tiny: 69,40%	Tidak ada.	Tidak diukur.	*Optimizer* SGD, Adam *Learning rate* 0,1 Tidak ada *hyperparameter tuning* yang sistematis.	Model di-*deploy* pada sebuah prototipe aplikasi Android. Fitur aplikasi meliputi deteksi menggunakan gambar statis dan siaran langsung dari kamera.	-

Berdasarkan 12 penelitian terdahulu yang tertera pada Tabel 2.1, dapat dilihat bahwa *convolutional neural network* (CNN) menunjukkan kinerja yang sangat memuaskan dalam mengklasifikasi gambar penyakit pada berbagai macam tanaman seperti pisang, anggur, padi, jeruk dan tomat, mencapai akurasi di atas 90% untuk semua tanaman tersebut. Hal ini menunjukkan menunjukkan bahwa CNN mampu mempelajari fitur visual penyakit tanaman secara efektif dari citra daun.

Pada klasifikasi penyakit daun tanaman pisang, beberapa penelitian menunjukkan bahwa arsitektur CNN ringan dapat menunjukkan kinerja prediksi yang kompetitif dengan arsitektur CNN yang lebih berat. Hal ini disoroti oleh penelitian [20], [21], [40] dan [26] yang melaporkan bahwa arsitektur seperti EfficientNet, MobileNet, dan SqueezeNet memiliki skor akurasi yang sebanding dengan arsitektur lebih berat seperti DenseNet, ResNet maupun Inception. Pada penelitian [20], BananaSqueezeNet bahkan mencapai skor akurasi yang lebih tinggi dibandingkan VGG16, InceptionV3, ResNet50, dan EfficientNetB0 meskipun dirancang dengan kompleksitas yang relatif lebih sederhana. Temuan-temuan tersebut menunjukkan bahwa peningkatan jumlah parameter dan kompleksitas model tidak selalu menghasilkan peningkatan kinerja prediksi yang signifikan.

Selain mengevaluasi akurasi, beberapa penelitian juga mulai mempertimbangkan efisiensi model. Penelitian [25] dan [26] mengevaluasi waktu inferensi dan menunjukkan bahwa arsitektur CNN ringan dapat memberikan latensi yang jauh lebih rendah dibandingkan arsitektur yang lebih kompleks. Sebagai contoh, penelitian [26] melaporkan bahwa MobileNetV3S hanya membutuhkan 14,6 detik untuk mengklasifikasi 500 gambar, sedangkan DenseNet121 memerlukan 342,5 detik untuk tugas yang sama, sehingga ada perbedaan sebesar ~23x. Hasil tersebut menunjukkan bahwa perbedaan efisiensi antar arsitektur dapat sangat signifikan, sehingga menjadi faktor yang perlu dipertimbangkan ketika model akan diimplementasikan pada perangkat dengan sumber daya komputasi terbatas seperti telepon genggam [28].

Beberapa penelitian juga telah membahas interpretabilitas model melalui penerapan Grad-CAM. Penelitian [26], [27], [41] menggunakan Grad-CAM untuk menghasilkan visualisasi *heatmap* yang menunjukkan bagian citra yang paling berpengaruh pada keputusan model. Visualisasi tersebut memungkinkan evaluasi kualitatif terhadap perilaku model sehingga dapat diketahui apakah model benar-benar memanfaatkan gejala penyakit pada daun sebagai dasar prediksi atau justru bergantung pada fitur lain yang tidak relevan seperti latar belakang lahan pertanian.

Ada juga penelitian yang melewati fase pelatihan dan evaluasi model, mencapai fase di mana model diimplementasikan pada perangkat keras yang sebenarnya menjadi tujuan akhir model. Penelitian [25], [39], [22] dan [23] melakukan implementasi model yang dibuat pada aplikasi berbasis web serta berbasis sistem operasi Android, sehingga mendemonstrasikan bagaimana model dapat digunakan dalam dunia asli oleh tenaga kerja agrikultur. Prototipe yang dibuat bervariasi dalam rangkaian fitur, dari deteksi sederhana menggunakan gambar statis sampai dengan deteksi menggunakan siaran langsung dari kamera dan pemaparan informasi cara memitigasi penyakit.

Meskipun penelitian terdahulu menunjukkan bahwa arsitektur CNN ringan mampu mencapai kinerja yang tinggi dalam klasifikasi penyakit tanaman secara umum dan pada penyakit daun tanaman pisang secara spesifik, namun masih terdapat kekurangan dalam penelitian terdahulu terkait penyakit daun tanaman pisang yang perlu diteliti lebih lanjut. Penelitian terdahulu umumnya berfokus pada peningkatan kinerja prediksi model yang diukur melalui akurasi klasifikasi atau berfokus pada implementasi model ke dalam prototipe aplikasi. Pendekatan tersebut mendemonstrasikan kemampuan model dalam mendeteksi penyakit daun tanaman pisang, namun belum memberikan gambaran yang menyeluruh mengenai kelayakan model untuk digunakan pada perangkat dengan sumber daya terbatas.

Pertama, implementasi model *deep learning* pada telepon genggam menuntut kebutuhan akan efisiensi karena keterbatasan sumber daya komputasi [30], [31]. Tanpa pertimbangan ini, model dengan akurasi yang tinggi belum tentu menjadi pilihan yang paling praktis apabila membutuhkan waktu inferensi yang lama,

ukuran model yang besar, atau penggunaan RAM yang tinggi [44], [45]. Dalam konteks *deployment* telepon genggam, model juga perlu memiliki ketahanan (*robustness*) terhadap variasi kualitas citra dan perbedaan distribusi data mengingat bahwa kondisi lapangan, bentuk daun dan gejala visual penyakit, serta kualitas kamera bisa berbeda [32]. Aspek ini belum terlihat di penelitian terdahulu.

Kedua, penggunaan Grad-CAM pada penelitian klasifikasi penyakit daun tanaman pisang masih relatif terbatas. Sebagian besar penelitian hanya melaporkan nilai akurasi tanpa memverifikasi apakah model benar-benar melokalisasi daun atau justru fitur lain yang tidak relevan. Akibatnya, model dengan akurasi tinggi tetap berpotensi mengambil keputusan berdasarkan pola yang tidak berkaitan dengan gejala penyakit, seperti latar belakang citra, pencahayaan, atau objek lain yang secara kebetulan muncul pada data pelatihan. Oleh karena itu, analisis interpretabilitas menjadi penting untuk meningkatkan kepercayaan terhadap hasil prediksi model [33].

Ketiga, penerapan *hyperparameter tuning* secara sistematis masih belum menjadi praktik umum dalam penelitian klasifikasi penyakit daun tanaman pisang. Sebagian besar penelitian menggunakan kombinasi *hyperparameter* yang telah ditentukan di awal, sedangkan hanya sedikit penelitian yang melakukan optimasi secara sistematis. Kondisi ini dapat menyebabkan perbandingan antararsitektur menjadi kurang adil karena setiap model belum tentu dievaluasi pada konfigurasi terbaiknya. Oleh sebab itu, optimasi *hyperparameter* diperlukan agar setiap arsitektur memperoleh kesempatan yang setara untuk mencapai kinerja optimal sebelum dilakukan perbandingan [34].

Untuk mengatasi kekurangan dari penelitian-penelitian sebelumnya, penelitian ini tidak hanya membandingkan tiga arsitektur CNN ringan, namun memberikan evaluasi yang lebih menyeluruh terhadap kelayakan model klasifikasi penyakit daun tanaman pisang untuk implementasi pada telepon genggam. Penelitian ini menganalisis *tradeoffs* antara kinerja prediksi dan efisiensi model dengan mempertimbangkan beberapa aspek secara bersamaan, yaitu akurasi, *robustness* terhadap korupsi citra, kemampuan generalisasi, ukuran model, waktu inferensi,

dan penggunaan RAM. Pendekatan ini memungkinkan evaluasi yang lebih realistis terhadap kebutuhan implementasi di telepon genggam yang memiliki keterbatasan sumber daya komputasi dan variasi lingkungan *deployment*.

Mengingat bahwa aspek akurasi, *robustness*, kemampuan generalisasi, dan efisiensi merupakan kriteria yang dapat saling berlawanan seperti yang telah dibuktikan pada penelitian terdahulu, maka pada penelitian ini, pemilihan model yang paling layak untuk *deployment* pada telepon genggam mengadopsi metode CRITIC dan TOPSIS. CRITIC berguna untuk menentukan bobot setiap kriteria berdasarkan informasi yang terkandung dalam data, sehingga metode ini tidak membutuhkan opini manusia untuk mendapatkan bobot, melainkan hanya *data-driven*. TOPSIS digunakan setelah CRITIC untuk menghasilkan *ranking* dari semua model, di mana model dengan peringkat tertinggi adalah model dengan *tradeoffs* yang paling minim antara keempat aspek yang dipertimbangkan, sehingga merepresentasikan model yang paling layak untuk *deployment* pada telepon genggam dari semua model yang dievaluasi.

Untuk memverifikasi kualitas prediksi model, maka penelitian ini mengintegrasikan Grad-CAM untuk menambah interpretabilitas model sehingga dapat diketahui jika model mampu melokalisasi daun dalam gambar. Untuk menjamin perbandingan antara arsitektur yang lebih adil, maka penelitian ini juga melakukan *hyperparameter tuning* untuk setiap model agar setiap model mendapatkan kesempatan untuk mencapai kinerja maksimalnya. Melalui pendekatan ini, penelitian ini menghasilkan rekomendasi model yang tidak hanya memiliki kinerja prediksi yang tinggi, tetapi juga layak untuk diimplementasikan pada telepon genggam dengan faktor-faktor yang relevan dalam konteksnya.

2.2 Tinjauan Teori

Subbab ini menjelaskan teori-teori yang terkait dengan penelitian ini. Secara garis besar, subbab ini membahas pentingnya ketahanan pangan dan ancaman penyakit tanaman. Subbab ini juga menjelaskan hal-hal terkait *convolutional neural network* dan cara evaluasi kinerja dan efisiensinya. Terakhir, subbab ini juga membahas teori *user testing* dan *multi-criteria decision making*.

2.2.1 Ketahanan Pangan

Ketahanan pangan dapat didefinisikan sebagai situasi di mana semua penduduk suatu negara memiliki akses terhadap pangan yang berkecukupan dan bernutrisi pada setiap saat agar dapat memiliki kehidupan yang sehat dan aktif. Ketahanan pangan menjadi salah satu prioritas dalam setiap negara dan secara internasional karena efeknya langsung berdampak pada kesejahteraan penduduk [46]. Kondisi ketahanan pangan yang buruk dapat meningkatkan jumlah penduduk yang mengalami kekurangan vitamin dan nutrisi sehingga membuat kelangsungan hidup semakin sulit, serta memiliki dampak lain seperti meningkatkan risiko terhadap penyakit jantung, diabetes, dan tekanan darah tinggi [47].

Ketahanan pangan dapat diukur dalam beberapa metrik seperti jumlah produksi pangan, harga umum pangan, serta kerugian pangan. Hal-hal tersebut menentukan seberapa mudah pangan yang sehat dapat diakses oleh semua kalangan populasi. Salah satu pemain utama dalam memastikan ketahanan pangan adalah sektor agrikultur yang menjadi produsen pangan organik yang memiliki banyak vitamin dan nutrisi, sehingga usaha-usaha sektor agrikultur dalam meningkatkan dan mempertahankan produksi pangan organik perlu didukung [46]. Namun, produktivitas di sektor ini terus terancam dengan adanya serangan penyakit pada berbagai tanaman yang dapat menyebabkan kerugian panen dan secara langsung mengurangi ketersediaan pangan bagi masyarakat, sehingga solusi terhadap identifikasi dan penanganan dini penyakit tanaman menjadi kunci untuk memastikan tidak terjadi infeksi pada skala besar [48].

2.2.2 Buah Pisang

Pisang merupakan buah yang dihasilkan dari tanaman genus *Musa*, dalam keluarga *Musaceae*. Dari segi gizi, pisang sangat bernilai karena komposisinya yang padat dengan nutrisi. Buah pisang mengandung banyak karbohidrat, potasium, vitamin B6 dan vitamin C. Kombinasi nutrisi ini menjadikan pisang sebagai sumber energi cepat yang juga mendukung fungsi saraf, tekanan darah sehat, dan kesehatan pencernaan [7].

Berdasarkan data dari *Food and Agriculture Organization*, lebih dari 50 juta ton buah pisang secara konsisten diproduksi setiap tahun. Lebih dari 18 juta ton buah pisang yang diekspor setiap tahunnya, sebagian besar berasal dari Ekuador, Filipina, dan Costa Rica. Sedangkan, negara yang paling banyak mengimpor buah pisang termasuk Amerika Serikat, Republik Rakyat Tiongkok, dan negara-negara dalam Uni Eropa seperti Perancis dan Jerman [49]. Secara global, lebih dari 400 juta penduduk bergantung pada buah pisang sebagai pangan pokok atau sebagai sumber penghasilan [13]. Hal ini menunjukkan bahwa buah pisang adalah salah satu komoditas strategis yang menghubungkan negara-negara tropis dengan pasar global yang berkontribusi terhadap pendapatan ekspor dan ketenagakerjaan di negara-negara produsen.

Secara botanis, jenis pisang yang diperdagangkan secara global didominasi oleh kultivar *Cavendish*. Kultivar ini diutamakan karena jumlah panen yang tinggi, umur simpan yang relatif panjang, dan rasa yang konsisten. Namun, dominasi monokultur ini menimbulkan kerentanan terhadap serangan penyakit. Tanpa variasi genetik yang memadai, suatu patogen yang berhasil menginfeksi satu tanaman pisang *Cavendish* berpotensi menyebar dengan cepat ke tanaman pisang *Cavendish* lainnya dan menyebabkan kerugian besar [13].

2.2.3 Penyakit Tanaman dan Tanaman Pisang

Subbab ini menjelaskan penyakit tanaman secara umum serta penyakit yang secara spesifik menyerang tanaman pisang. Pembahasan diawali dengan konsep dasar mengenai penyakit tanaman sebagai landasan untuk memahami karakteristik penyakit pada tanaman pisang. Selanjutnya, dijelaskan berbagai penyakit yang umum menyerang tanaman pisang beserta karakteristiknya.

2.2.3.1 Penyakit Tanaman Secara Umum

Penyakit tanaman dapat didefinisikan sebagai kondisi fisiologis yang abnormal dalam sebuah tanaman yang mengganggu fungsi normal tanaman. Penyakit tanaman dapat menyebabkan tanaman yang terkena penyakit untuk menghasilkan panen yang berkualitas buruk atau

mengurangi jumlah hasil panen. Dalam kasus penyakit tanaman yang parah, penyakit tanaman juga dapat menyebabkan kematian pada tanaman yang terkena [50]. Pada umumnya, penyakit tanaman dapat muncul karena dua faktor, yaitu:

1. Patogen

   Patogen adalah mikroorganisme yang dapat menyebabkan induknya untuk menderita penyakit. Patogen dapat dibagi menjadi empat jenis yaitu fungi, bakteri, virus, dan nematoda [50].

2. Kondisi Lingkungan dan Tanaman

   Beberapa kondisi lingkungan dapat menyebabkan penyakit tanaman. Contohnya, kelembapan udara yang tinggi dapat mendorong pertumbuhan fungi dan bakteri, serta suhu udara yang tinggi dapat menyebabkan tanaman untuk menjadi lebih rentan terhadap infeksi. Kondisi tanaman juga dapat menyebabkan tanaman untuk terkena penyakit, contohnya karena kekurangan nutrisi. Penyakit tanaman dapat menyebar melalui beberapa media seperti angin, hujan, serangga, benih tanaman, serta alat perkebunan yang tercemar. Selain itu, beberapa penyakit tanaman dapat bertahan hidup cukup lama sehingga menyebabkan infeksi untuk terjadi secara periodik [50].

Ketika sebuah penyakit tanaman menyerang tanaman, pada umumnya akan muncul gejala-gejala yang dapat diamati secara visual, seperti bercak-bercak warna pada daun, atau pertumbuhan tanaman mengalami deformasi. Pada kasus yang serius, gejala visual termasuk tanaman yang memiliki bagian yang layu, mengering, atau membusuk [51].

2.2.3.2 Penyakit Daun Tanaman Pisang

Seperti tanaman pada umumnya, tanaman pisang juga dapat menderita beberapa penyakit yang disebabkan oleh faktor-faktor yang disebutkan pada subbab 2.2.3.1. Di antara penyakit daun tanaman pisang, sigatoka hitam dan penyakit panama merupakan salah dua penyakit yang paling berdampak di dunia asli saat ini.

1. Sigatoka Hitam

   Disebabkan oleh fungi *Mycosphaerella fijiensis*. Gejala visual yang muncul saat tanaman pisang terkena penyakit ini adalah bercak gelap pada daun dengan pinggiran menguning di sekitar bercak tersebut (contoh pada Gambar 2.1). Proses infeksi berproses dengan lebih cepat daripada varian sigatoka lainnya yaitu sigatoka kuning, sehingga penyakit ini merupakan varian yang lebih agresif dari keluarga penyakit sigatoka. Penyakit ini membuat fotosintesis tanaman sulit karena mematikan sel-sel daun sehingga membuat pertumbuhan tanaman terhambat. Selain mengganggu daun tanaman itu sendiri, penyakit ini juga mengurangi kualitas buah yang dihasilkan dengan menyebabkan pematangan buah pisang yang prematur sehingga mempersulit transportasi buah dari satu tempat ke tempat lain. [52]. Jika penyakit ini tidak ditangani, kerugian hasil panen dapat mencapai 80% [9].

Gambar 2.1 Contoh Penyakit Sigatoka Hitam
Sumber: [53]

2. Penyakit Panama

Dikenal juga sebagai *fusarium wilt*, penyakit ini disebabkan oleh fungi *Fusarium oxysporum f.sp. cubense*. Gejala visual yang muncul saat tanaman pisang terkena penyakit ini adalah daun yang layu atau menguning yang dimulai dari bawah (contoh pada Gambar 2.2). Penyakit ini membuat fotosintesis tanaman sulit karena mematikan sel-sel daun sehingga membuat pertumbuhan tanaman terhambat. Selain itu, buah yang dihasilkan oleh tanaman yang terinfeksi cenderung berukuran kecil. Penyakit panama tidak hanya menginfeksi tanaman itu sendiri, tetapi juga menular ke lahan yang diduduki oleh tanaman yang terinfeksi, sehingga mengurangi area lahan yang aman untuk menanam tanaman pisang baru [14]. Penyakit panama sudah pernah menyebabkan salah satu kultivar pisang, *gros michel*, menjadi hampir punah pada tahun 1950-an. Saat ini, pisang yang paling banyak diekspor di seluruh dunia juga memiliki vulnerabilitas terhadap penyakit ini dan terancam oleh persebaran penyakit ini [11].

Gambar 2.2 Contoh Penyakit Panama
Sumber: [54]

2.2.4 *Machine Learning*

Machine learning atau pembelajaran mesin adalah algoritma komputer yang berbasis pada konsep bahwa komputer dapat belajar dari data yang mengandung pola atau tren, kemudian menggunakan pengetahuan yang didapatkan untuk menentukan hal yang perlu dilakukan selanjutnya. *Machine learning* berkontras dengan algoritma konvensional pada komputer yang

menggunakan seperangkat aturan yang ditetapkan secara eksplisit untuk mencapai suatu tujuan. *Machine learning* merupakan salah satu bagian dari payung besar *artificial intelligence* atau kecerdasan buatan [55]. Berdasarkan data yang digunakan untuk melatih sebuah model, *machine learning* dapat dibagi menjadi dua jenis, yaitu:

1. *Supervised Learning*

   Supervised learning adalah jenis pelatihan model *machine learning* yang menggunakan data pelatihan yang sudah memiliki label untuk setiap poin data [56]. Pelatihan sebuah model untuk mengklasifikasi gambar penyakit tanaman termasuk *supervised learning* karena setiap gambar sudah dilabel dengan jenis penyakit.
2. *Unsupervised Learning*

   Unsupervised learning adalah jenis pelatihan model *machine learning* yang menggunakan data pelatihan yang tidak memiliki label, sehingga model harus mempelajari karakteristik data secara mandiri dan mengelompokkan setiap poin-poin data berdasarkan karakteristik yang dipelajari [56].

2.2.5 *Artificial Neural Network* dan *Deep Learning*

Artificial Neural Network (ANN) merupakan model komputasi yang terinspirasi dari struktur dan cara kerja jaringan saraf pada otak manusia. ANN terdiri atas sejumlah neuron buatan yang saling terhubung dan bekerja secara bersama-sama untuk mempelajari pola dari data yang diberikan. Secara umum, ANN memiliki tiga jenis lapisan utama, yaitu sebagai berikut:

1. Lapisan Input

   Lapisan input adalah lapisan paling pertama dalam arsitektur ANN yang menjadi penerima data awal untuk kemudian diproses di lapisan tersembunyi [56].
2. Lapisan Tersembunyi

   Lapisan tersembunyi merupakan lapisan di mana model memproses data input dengan mempelajari karakteristik dari data tersebut seperti mencari

pola dan hubungan dalam data. Sebuah arsitektur ANN biasanya memiliki lebih dari satu lapisan tersembunyi, di mana lapisan selanjutnya menerima hasil kalkulasi dari lapisan sebelumnya sampai mencapai lapisan output [56].

3. Lapisan Output

   Lapisan output adalah lapisan paling terakhir dalam arsitektur ANN yang berfungsi sebagai pemberi hasil akhir dari pemrosesan data yang dilakukan pada lapisan-lapisan sebelumnya [56].

Deep learning merupakan bentuk *machine learning* yang lebih kapabel karena menggunakan konsep *artificial neural network* dengan banyak lapisan (sehingga disebut *neural network* yang "dalam" atau *deep neural network*) untuk mempelajari data yang memiliki hubungan yang lebih kompleks atau non-linier [56].

2.2.6 Fungsi *Activation*

Dalam sebuah *Artificial Neural Network* (ANN), terdapat berbagai fungsi aktivasi (*activation function*) yang berperan dalam menerapkan sifat nonlinier pada jaringan sehingga model mampu mempelajari pola dan hubungan data yang kompleks. Tanpa fungsi aktivasi, jaringan hanya akan melakukan transformasi linear yang membatasi kemampuannya dalam menyelesaikan permasalahan dengan pola yang tidak linear. Oleh karena itu, pemilihan fungsi aktivasi yang tepat menjadi salah satu faktor penting dalam menentukan kinerja model ANN [57].

2.2.6.1 Softmax

Softmax adalah fungsi *activation* yang mengeluarkan output neuron dalam bentuk distribusi probabilitas dari vektor angka. Jumlah dari semua probabilitas akan sama dengan 1. Softmax layak digunakan dalam masalah klasifikasi non-biner (lebih dari 2 kelas) sebagai fungsi *activation* pada lapisan output karena outputnya merupakan probabilitas data input terhadap setiap kelas yang ada [57]. Rumus 2.1 menunjukkan fungsi softmax.

$$f(x_i) = \frac{e^{x_i}}{\sum \frac{n}{j=1} e^{x_i}}$$

Rumus 2.1 Fungsi Softmax
Sumber: [57]

e^{x_i} = Eksponen bilangan Euler dengan nilai input

$\sum \frac{n}{j=1} e^{x_i}$ = Jumlah nilai yang telah dieksponen

2.2.6.2 Rectified Linear Unit (ReLU)

ReLU adalah fungsi *activation* yang mengeluarkan output neuron secara langsung jika output tersebut > 0. Jika output neuron < 0, maka ReLU akan mengeluarkan 0. ReLU mengatasi masalah *vanishing gradient* di mana kecepatan model dalam mempelajari hubungan dalam data menurun drastis karena gradien yang digunakan untuk memberikan umpan balik ke setiap neuron berkurang secara signifikan. Kelebihan yang dimiliki oleh ReLU membuatnya efektif untuk digunakan dalam lapisan tersembunyi [57]. Rumus 2.2 menunjukkan fungsi ReLU.

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

Rumus 2.2 Fungsi ReLU
Sumber: [57]

x = Nilai input

2.2.6.3 Swish

Swish adalah fungsi *activation* yang berbasis pada fungsi sigmoid dan mengikuti karakteristik ReLU. Namun, berbeda dengan fungsi ReLU yang mengosongkan output neuron jika < 0, swish memungkinkan nilai negatif yang kecil untuk tetap lewat. Karakteristik swish ini dapat mencegah kejadian di mana neuron tetap 0 selama proses pelatihan model yang akan membuat neuron tersebut tidak mempelajari fitur data. Swish juga memiliki kurva output yang lebih landai dibandingkan ReLU, sehingga memungkinkan proses pelatihan yang lebih stabil [58], [59]. Rumus 2.3 menunjukkan fungsi swish.

$$f(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}}$$

Rumus 2.3 Fungsi Swish
Sumber: [58], [59]

x = Nilai input

$\sigma(x)$ = Fungsi sigmoid

2.2.7 Fungsi *Cross-entropy Loss*

Fungsi *cross-entropy loss* adalah fungsi yang menghitung *loss* berdasarkan perbedaan antara distribusi probabilitas kelas yang diprediksi oleh model dengan distribusi probabilitas kelas asli. Fungsi *cross-entropy loss* oleh karena itu menjadi ideal untuk tujuan pelatihan model klasifikasi karena pengurangan *cross-entropy loss* akan menghasilkan kinerja prediksi model yang lebih akurat [60]. Rumus 2.4 menunjukkan fungsi *cross-entropy loss*.

$$cce(y, \hat{y}) = -\frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{c} y_{i,j} \log(\hat{y}_{i,j})$$

Rumus 2.4 Fungsi *Cross-entropy Loss*
Sumber: [60]

M = Jumlah sampel dalam *batch*

c = Jumlah kelas

$y_{i,j}$ = Label *one-hot* asli untuk sampel *i* dalam vektor kelas *j*

$\hat{y}_{i,j}$ = Label *one-hot* yang diprediksi untuk sampel *i* dalam vektor kelas *j*

2.2.8 *Optimizer*

Optimizer adalah sebuah algoritma yang digunakan untuk membantu meningkatkan akurasi model *deep learning* saat pelatihan dan mempercepat proses pelatihan dengan mencari parameter yang paling optimal untuk *neural network* yang dibuat. *Optimizer* bekerja atas prinsip *gradient descent*, yaitu pergerakan parameter secara iteratif terhadap nilai *loss* terkecil sehingga prediksi yang dibuat oleh *neural network* semakin akurat dengan data asli [61].

2.2.8.1 RMSprop

RMSprop, atau *Root Mean Square Propagation* adalah algoritma *optimizer* yang menggunakan prinsip *stochastic gradient descent*

(SGD) dan dikombinasikan dengan konsep *adaptive learning rate*. SGD adalah versi yang lebih efisien dari *gradient descent* konvensional dengan menggunakan salah satu poin data pelatihan secara random pada setiap iterasi daripada menggunakan seluruh *dataset*, sehingga mempercepat iterasi dan mengurangi penggunaan sumber daya memori pada komputer yang melatih model. *Adaptive learning rate* memungkinkan setiap parameter model untuk memiliki *learning rate* tersendiri yang dapat berubah secara independen sehingga membantu model untuk lebih melekat pada fitur data. Perubahan *learning rate* dilakukan dari perhitungan kuadrat gradien (*second moment*) untuk menstabilkan pelatihan model [61], [62].

2.2.8.2 Adam

Adam, atau *Adaptive Moment Estimation* adalah algoritma *optimizer* yang mengembangkan mekanisme RMSprop lebih lanjut. Seperti RMSprop, Adam juga menggunakan prinsip SGD dengan *adaptive learning rate* yang berubah berdasarkan perhitungan kuadrat gradien (*second moment*). Namun, Adam juga menambahkan konsep momentum melalui perhitungan rata-rata gradien (*first moment*). Konsep momentum ini berfungsi untuk mempercepat proses konvergensi model dengan memanfaatkan arah pergerakan pada iterasi-iterasi sebelumnya. Hal ini mencegah perubahan nilai gradien yang terlalu drastis sehingga menjaga stabilitas pelatihan model. [62], [63].

2.2.9 *Learning rate*

Learning rate adalah *hyperparameter* yang dapat digunakan selama pelatihan model untuk mengendalikan ekstremitas model dalam mengubah *weight* terhadap nilai error yang lebih rendah. *Learning rate* secara langsung mempengaruhi kecepatan konvergensi model, di mana nilai yang terlalu tinggi dapat membuat konvergensi yang kurang optimal dan nilai yang terlalu rendah dapat membuat proses pelatihan yang terlalu lambat, sehingga nilai *learning rate* yang ideal perlu ditentukan secara cermat [64].

2.2.10 *Transfer Learning*

Transfer learning adalah pendekatan pelatihan model *deep learning* yang menggunakan model yang telah dilatih sebelumnya untuk melakukan tugas selain yang telah didesain untuk model tersebut. *Transfer learning* bekerja di atas prinsip bahwa pengetahuan yang telah didapatkan oleh model dapat dibawa kepada pengaplikasian lain [65]. Manfaat utama dari pendekatan *transfer learning* adalah meningkatkan efisiensi proses pelatihan model dengan mengurangi waktu yang dibutuhkan untuk mendapatkan hasil kinerja model yang cukup karena model sudah memiliki pengetahuan dasar yang dapat diaplikasikan terhadap tujuan baru.

Secara garis besar, *transfer learning* memiliki beberapa jenis pendekatan. Pendekatan yang paling sederhana adalah *full fine-tuning* di mana semua lapisan dalam model dilatih kembali untuk tujuan baru sehingga dapat mempelajari fitur-fitur dalam *dataset* baru secara maksimal. Namun, pendekatan ini dapat menyebabkan kejadian *catastrophic forgetting* di mana fitur-fitur yang telah dipelajari oleh model sebelumnya terhapus dan dapat menyebabkan *overfitting* pada *dataset* yang kecil dengan jumlah parameter model yang banyak. Sebaliknya, ada pendekatan pembekuan lapisan fitur, di mana beberapa atau semua dari lapisan fitur dalam jaringan model dibekukan sehingga model tetap menggunakan fitur yang telah dipelajari sebelumnya, dan hanya lapisan *fully connected* yang diadaptasi untuk tujuan baru. Pendekatan ini cenderung mengurangi terjadinya *overfitting*, namun sebagai *tradeoff*, model belum tentu sepenuhnya mempelajari fitur-fitur pada *dataset* baru [27], [66].

2.2.10.1 ImageNet

ImageNet adalah sebuah *dataset labelled* yang terdiri dari lebih dari 14 juta gambar yang dipisah menjadi lebih dari 20 ribu kelas. Varian ImageNet yang lebih kecil, disebut sebagai ImageNet-1K yang terdiri dari 1.000 kelas dan sekitar 1.2 juta gambar, sering dipakai untuk *pretraining* model CNN karena memungkinkan model untuk mempelajari fitur-fitur gambar secara efektif. Hal ini memudahkan

transfer learning karena model yang sudah dilatih pada ImageNet-1K kemudian dapat dilatih lagi untuk tujuan yang lebih terfokus, seperti klasifikasi penyakit tanaman, dengan konvergensi yang lebih cepat [67].

2.2.11 *Perceptual Hashing*

Perceptual hashing (PHash) adalah teknik untuk mengkonversi sebuah gambar menjadi sebuah *string hash*. Dua gambar yang sangat mirip akan memiliki *string hash* yang hampir sama, sehingga gambar yang memiliki kemiripan dengan satu sama lain dapat dideteksi melalui metode ini. Untuk mengkonversi menjadi *string hash*, maka PHash pertama melakukan *downscaling* pada gambar dan membuatnya hitam-putih agar gambar lebih sederhana. Kemudian, data frekuensi rendah dalam gambar diekstraksi menggunakan *discrete cosine transform* (DCT) untuk mendapatkan struktur keseluruhan gambar dan mengurangi gangguan frekuensi tinggi seperti *noise*. Frekuensi ini dikonversi menjadi *string* yang merupakan *hash* dari gambar tersebut. Setelah *string hash* dari sebuah gambar didapatkan, maka *string hash* ini dapat dibandingkan dengan *string hash* dari gambar lain menggunakan jarak Hamming yang ditunjukkan pada Rumus 2.5. Semakin kecil jarak Hamming antara dua *string hash*, maka semakin mirip kedua gambar yang dibandingkan. PHash dapat mendeteksi gambar yang mirip, namun dengan limitasi. Rotasi, *cropping* atau distorsi berat dapat mengurangi kemampuan PHash [68].

$$d(H_1, H_2) = \sum_{i=1}^{m} (H_{1i} \oplus H_{2i})$$

Rumus 2.5 Jarak Hamming
Sumber: [68]

H = *String hash*

m = Panjang *hash*

2.2.12 CNN *Embeddings*

CNN *embeddings* adalah teknik untuk mengkonversi sebuah gambar menjadi vektor 1 dimensi yang berisi fitur-fitur dalam gambar tersebut. Teknik ini mengeksploitasi output dari blok ekstraksi fitur terakhir dalam sebuah jaringan CNN karena output tersebut merupakan representasi fitur tingkat tinggi dalam sebuah gambar seperti tekstur dan struktur sebuah objek. Implikasinya adalah karena CNN memahami gambar secara semantik, maka vektor 1 dimensi yang dihasilkan antara 2 gambar yang mirip akan lebih tahan terhadap gangguan ekstrim dibandingkan metode *perceptual hashing*. Jarak antara vektor 2 gambar dapat dihitung menggunakan kesamaan kosinus dan menghasilkan nilai antara -1 dan 1. Nilai yang mendekati 1 berarti gambar semakin mirip dengan gambar lain yang dibandingkan. Namun, kekurangan dari teknik CNN *embeddings* adalah membutuhkan waktu yang relatif lebih lama untuk menghasilkan vektor *embeddings* dari setiap gambar yang dibandingkan [68].

$$sim(z_1, z_2) = \frac{z_1 \times z_2}{|z_1||z_2|}$$

Rumus 2.6 Kesamaan Kosinus
Sumber: [68]

z = Vektor *embeddings*

2.2.13 Metrik Kinerja Prediksi Model Klasifikasi

Subbab ini menjelaskan metrik yang mengukur kinerja model klasifikasi secara kuantitatif. Hal-hal yang dibahas meliputi *confusion matrix*, skor akurasi, *precision*, *recall*, skor F1 dan *mean corruption accuracy*.

2.2.13.1 *True Positive, True Negative, False Positive, False Negative*

True positive (TP) adalah jumlah instansi di mana model memprediksi suatu data sebagai kelas positif dan sebenarnya memang kelas positif, sehingga prediksi model benar [69].

True negative (TN) adalah jumlah instansi di mana model memprediksi suatu data sebagai kelas negatif dan sebenarnya memang kelas negatif, sehingga prediksi model benar [69].

False positive (FP) adalah jumlah instansi di mana model memprediksi suatu data sebagai kelas positif, namun data tersebut sebenarnya termasuk kelas negatif, sehingga prediksi model salah [69].

False negative (FN) adalah jumlah instansi di mana model memprediksi suatu data sebagai kelas negatif, namun data tersebut sebenarnya termasuk kelas positif, sehingga prediksi model salah [69].

2.2.13.2 *Confusion Matrix*

Confusion matrix adalah salah satu cara untuk mengevaluasi kinerja model klasifikasi dengan meringkas jumlah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) yang diprediksi oleh model berdasarkan setiap kelas yang ada dalam data dalam bentuk matriks [70]. Gambar 2.3 menunjukkan bentuk *confusion matrix*.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Gambar 2.3 Bentuk *Confusion Matrix*
Sumber: [71]

2.2.13.3 Skor Akurasi

Skor akurasi adalah metrik evaluasi yang digunakan untuk mengukur seberapa sering sebuah model klasifikasi memprediksi dengan benar dibandingkan dengan jumlah prediksi yang dibuat [72]. Rumus 2.7 menunjukkan rumus dari skor akurasi.

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

Rumus 2.7 Skor Akurasi
Sumber: [72]

2.2.13.4 *Precision*

Precision adalah metrik evaluasi yang digunakan untuk mengukur seberapa akurat semua prediksi positif yang dihasilkan oleh sebuah model. Dalam kata lain, *precision* mencari berapa banyak instansi yang sebenarnya positif dari semua prediksi positif yang dihasilkan oleh sebuah model [69]. Rumus 2.8 menunjukkan rumus dari *precision*.

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

Rumus 2.8 *Precision*
Sumber: [69]

2.2.13.5 *Recall*

Recall adalah metrik evaluasi yang digunakan untuk mengukur seberapa akurat prediksi model dalam mencari instansi yang sebenarnya memang positif. *Recall* dikenal juga sebagai *sensitivity* [72]. Rumus 2.9 menunjukkan rumus dari *recall*.

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

Rumus 2.9 *Recall*
Sumber: [72]

2.2.13.6 Skor F_1

Skor F_1 adalah rata-rata harmonis antara *precision* dan *recall* yang dapat memberikan gambaran kinerja model klasifikasi yang lebih representatif ketika *dataset* yang digunakan memiliki jumlah instansi per kelas yang tidak seimbang [69]. Rumus 2.10 menunjukkan rumus dari skor F_1.

$$F_1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Rumus 2.10 Skor F_1
Sumber: [69]

2.2.13.7 *Mean Corruption Accuracy*

Mean corruption accuracy atau mCA adalah metrik yang digunakan untuk menghitung akurasi rata-rata dari semua kombinasi

jenis korupsi dan tingkat korupsi gambar [73] yang pertama distandarisasi oleh Hendrycks et al. pada publikasinya di tahun 2021. Jenis korupsi gambar yang diusulkan antara lain adalah perubahan kecerahan, kontras, penambahan *blur* dan *noise*. Korupsi ini dirancang untuk menguji *robustness* suatu model terhadap berbagai macam korupsi yang sering terjadi pada gambar [74].

2.2.14 Metrik Efisiensi Model

Subbab ini menjelaskan metrik yang mengukur efisiensi model. Ada tiga jenis metrik efisiensi yang digunakan dalam penelitian ini yaitu ukuran file model, penggunaan RAM model dan waktu inferensi model. Setiap metrik mengevaluasi faktor efisiensi yang berbeda terhadap kelayakan *deployment* telepon genggam.

2.2.14.1 Ukuran File Model

Ukuran model adalah total ruang yang diambil oleh sebuah model pada media penyimpanan. Untuk telepon genggam modern, media penyimpanan yang dipakai adalah sejenis *flash memory* [75]. Ukuran model penting untuk dipertimbangkan karena telepon genggam pada umumnya memiliki kapasitas penyimpanan terbatas, terutama pada telepon genggam yang ditujukan untuk pasar *low-mid-range*, sehingga model *deep learning* yang akurat tetapi memiliki ukuran besar tidak pantas untuk diimplementasikan dalam telepon genggam [76].

2.2.14.2 Penggunaan RAM Model

Penggunaan RAM adalah total data sementara yang disimpan oleh model pada RAM ketika model menjalankan inferensinya. Berbeda dari media penyimpanan permanen, RAM pada umumnya memiliki kapasitas yang beberapa kali lipat lebih rendah karena dirancang untuk memprioritaskan kecepatan operasi data daripada kapasitas murni. Kapasitas RAM pada telepon genggam juga pada umumnya terbatas, terutama pada telepon genggam yang ditujukan untuk pasar *low-mid-end*. Penggunaan RAM menjadi pertimbangan kritis ketika model

ditujukan untuk telepon genggam karena limitasi ini [77]. Pada perangkat dengan kapasitas RAM terbatas, model dengan penggunaan RAM tinggi dapat menyebabkan penurunan kinerja aplikasi latar depan maupun latar belakang [78], atau bahkan menyebabkan kegagalan aplikasi (*crash*) [44].

2.2.14.3 Waktu Inferensi Model

Waktu inferensi model adalah waktu yang dibutuhkan oleh model untuk menghasilkan output akhir sejak data input pertama memasuki jaringan model. Waktu inferensi dipengaruhi oleh beberapa faktor, namun salah satu faktor tersebut adalah kecepatan *processor* yang menjalankan model seperti CPU atau GPU [77]. Bagi pengguna akhir model, waktu inferensi mempengaruhi rasa responsivitas model [79]. Contohnya, jika model digunakan untuk memprediksi gambar langsung dari kamera telepon genggam, maka waktu inferensi yang lambat akan memperlambat waktu yang dibutuhkan bagi pengguna untuk melihat prediksi terbaru dari model.

2.2.15 Interpretabilitas Model (*Explainable AI*)

Interpretabilitas model adalah kemampuan untuk memahami alasan di balik prediksi yang dibuat oleh model *machine learning* dan *deep learning*. Interpretabilitas model mengatasi masalah *black box*, yaitu masalah prediksi model yang tidak dilakukan secara transparan sehingga tidak diketahui bagaimana model mencapai prediksi tersebut. Model yang dapat diinterpretasi dapat meningkatkan kepercayaan terhadap model tersebut karena pengguna model dapat memvalidasi prediksi model. Interpretabilitas model juga dapat membantu dalam pengembangan model *machine learning* dan *deep learning* karena dapat menunjukkan jika model memang memperhatikan fitur-fitur data yang seharusnya dipelajari [80], [81].

2.2.15.1 Grad-CAM

Gradient class activation mappings, disingkat sebagai Grad-CAM, adalah teknik untuk mengatasi masalah *black box* dalam model CNN

dengan menunjukkan area dalam gambar yang paling diperhatikan oleh model dalam membuat prediksinya. Visualisasi ini ditunjukkan menggunakan sebuah *heatmap* yang ditimpa di atas gambar asli, di mana warna biru menunjukkan area yang paling tidak berkontribusi terhadap prediksi model, dan warna merah menunjukkan area yang paling berkontribusi terhadap prediksi model [38].

Secara teknis, Grad-CAM bekerja pada lapisan konvolusional terakhir (atau mendekati terakhir) pada sebuah model CNN. Lokasi lapisan ini biasanya menjadi pilihan utama karena masih memiliki informasi spasial mengenai fitur dalam gambar (sebelum menjadi lapisan *fully connected*) dan memiliki representasi fitur tingkat tinggi. Grad-CAM menghitung gradien dari kelas yang diprediksi oleh model terhadap setiap *feature map* yang ada dalam lapisan konvolusional yang digunakan oleh Grad-CAM, kemudian rata-rata dari setiap gradien dihitung untuk mendapatkan *weight* dari setiap *feature map*. *Weight* ini menunjukkan seberapa besar kontribusi setiap *feature map* dalam mempengaruhi model untuk memilih kelas yang diprediksi. Semua *feature map* kemudian dikombinasikan berdasarkan *weight*-nya untuk mendapatkan *heatmap* akhir [33], [82].

2.2.16 *System Usability Scale*

System Usability Scale, disingkat sebagai SUS, adalah sebuah standar kuesioner yang pertama dibuat oleh John Brooke pada tahun 1996. Kuesioner ini terdiri dari 10 pertanyaan berupa pernyataan yang dirancang untuk mengukur kegunaan sebuah sistem seperti aplikasi telepon genggam. Setiap pertanyaan cukup dijawab menggunakan skala Likert yaitu dari angka 1-5, di mana 1 berarti partisipan sangat tidak setuju dengan pernyataan yang sedang dievaluasi, dan 5 berarti partisipan sangat setuju [83]. Kelebihan SUS adalah jumlah pertanyaannya yang sedikit, sehingga pengisian survei SUS dapat dilakukan dengan mudah dan cepat. SUS juga telah diverifikasi sebagai instrumen survei yang dapat diandalkan dan valid secara psikometrik [84]. Tabel 2.2 menunjukkan 10 pertanyaan dalam SUS.

Tabel 2.2 Pertanyaan dalam *System Usability Scale*
Sumber: [83]

No	Pertanyaan
1	Saya merasa bahwa saya ingin sering menggunakan sistem ini.
2	Saya merasa sistem ini terlalu rumit untuk digunakan.
3	Saya merasa sistem ini mudah untuk digunakan.
4	Saya merasa bahwa saya akan membutuhkan dukungan dari seseorang yang mengerti teknologi untuk menggunakan sistem ini.
5	Saya merasa bahwa semua fitur dalam sistem ini terintegrasi dengan baik.
6	Saya merasa bahwa ada terlalu banyak ketidakkonsistenan dalam aplikasi ini.
7	Saya merasa bahwa sebagian besar orang lain dapat mempelajari dengan cepat bagaimana caranya menggunakan sistem ini.
8	Saya merasa bahwa sistem ini sangat merepotkan untuk dipakai.
9	Saya merasa sangat yakin ketika menggunakan sistem ini.
10	Saya perlu mempelajari banyak hal baru sebelum saya dapat menggunakan sistem ini.

Dari 10 pertanyaan yang ada dalam SUS, pertanyaan ganjil (1, 3, 5, 7, 9) memiliki konotasi positif, sedangkan pertanyaan genap (2, 4, 6, 8, 10) memiliki konotasi negatif. Pertanyaannya dirancang seperti ini untuk mengurangi bias respon. Untuk pertanyaan ganjil, skor yang dihitung adalah angka dari jawaban responden lalu diselisih dengan 1. Sedangkan, untuk pertanyaan genap, skor yang dihitung adalah 5 diselisih dengan angka dari jawaban responden. Setelah semua skor dari setiap pertanyaan telah dihitung, maka dapat dijumlah menjadi satu skor akhir yang berskala 0-40, kemudian dikali dengan 2.5 agar skala ini berubah menjadi 0-100. Skor SUS yang berada di atas 70 secara umum berarti sistem yang dievaluasi cukup baik dalam kegunaan [85].

2.2.17 *Multi-Criteria Decision Making*

Multi-criteria decision making yang disingkat sebagai MCDM adalah cabang ilmu keputusan yang dirancang untuk mengevaluasi dan memilih solusi terbaik dalam skenario di mana pemilihan suatu solusi melibatkan kriteria yang berlawanan dengan satu sama lain (peningkatan dalam nilai satu kriteria dapat menyebabkan nilai kriteria lain untuk memburuk). Dalam kasus seperti ini, optimalisasi solusi terhadap satu kriteria menjadi tidak efektif. Sebaliknya, MCDM memberikan kerangka kerja matematis untuk memilih solusi dengan *tradeoff* yang paling seimbang antara semua kriteria berlawanan. Untuk melakukan hal tersebut, pada umumnya MCDM pertama mengidentifikasi

kriteria evaluasi yang ada, melakukan normalisasi data untuk menyeragamkan nilai, menetapkan bobot ke setiap kriteria, kemudian menghasilkan daftar peringkat dari solusi yang terbaik sampai dengan solusi yang terburuk berdasarkan bobot tersebut [86]. Melalui pendekatan ini, MCDM mengurangi minimalisasi bias subjektif dan menghasilkan basis argumentasi lebih yang transparan dan konsisten [87].

2.2.17.1 CRITIC

Criteria Importance Through Intercriteria Correlation atau disingkat sebagai CRITIC adalah metode untuk menetapkan bobot pada kriteria dalam MCDM secara objektif dengan menghitung nilai kepentingan setiap kriteria secara langsung dari data dalam *decision matrix*. Prinsip dasar CRITIC adalah menggunakan kontras intensitas beserta dengan konflik antar-kriteria untuk menghitung bobot objektif untuk setiap kriteria. Hal ini mengurangi bias atau subjektivitas manusia karena bobot didapatkan secara matematis dan berdasarkan karakteristik data dalam *decision matrix* itu sendiri [35], [36].

Untuk menghitung bobot dari setiap kriteria dalam *decision matrix* MCDM, CRITIC pertama melakukan normalisasi pada setiap nilai untuk menyamakan skala dari semua kriteria. Normalisasi ini dapat dilakukan menggunakan metode seperti pada Rumus 2.15. Deviasi standar dari setiap kriteria kemudian dihitung. Deviasi standar ini akan dipakai nantinya untuk menghitung konten informasi yaitu kekuatan diskriminatif dari setiap kriteria. Selanjutnya, dihitung koefisien korelasi dari seluruh kriteria menggunakan Rumus 2.11. Setelah mendapatkan koefisien korelasi, maka dapat dihitung konflik antar kriteria menggunakan Rumus 2.12. Selanjutnya, dihitung konten informasi menggunakan Rumus 2.13. Akhirnya, dapat dihitung bobot objektif dari setiap kriteria menggunakan Rumus 2.14 [35], [36].

$$r_{ij} = \frac{\sum_{p=1}^{n}(x_{pi} - \bar{x_i})(x_{pi} - \bar{x_j})}{\sum_{p=1}^{n}(x_{pi} - \bar{x_i})^2 \sum(x_{pj} - \bar{x_j})^2}$$

Rumus 2.11 Koefisien Korelasi
Sumber: [35], [36]

$$R_j = \sum_{j=1}^{n}(1 - r_{ij})$$

Rumus 2.12 *Conflict of Indicators*
Sumber: [35], [36]

$$C_j = \sigma_j \sum_{j=1}^{n}(1 - r_{ij})$$

Rumus 2.13 *Information Content*
Sumber: [35], [36]

$$w_j = \frac{C_j}{\sum_{j=1}^{n} C_j}$$

Rumus 2.14 *Objective Weight*
Sumber: [35], [36]

i = Kriteria yang sedang dievaluasi

j = Kriteria lain untuk perbandingan

p = Data yang sedang dievaluasi

2.2.17.2 TOPSIS

Technique for Order of Preference by Similarity to Ideal Solution atau disingkat sebagai TOPSIS adalah metode untuk menentukan solusi terbaik yang merupakan *trade-off* terbaik pada semua kriteria dari sekian banyak solusi lain. TOPSIS bekerja secara geometris, di mana solusi terbaik yang dipilih otomatis memiliki jarak terdekat dengan solusi ideal positif dan jarak terjauh dari solusi ideal negatif. TOPSIS pertama menormalisasi semua nilai dalam *decision matrix* menggunakan Rumus 2.15. Kemudian, *weight* dari setiap kriteria (seperti yang didapatkan dari hasil CRITIC) dikalikan terhadap setiap nilai dalam matriks TOPSIS. Solusi ideal positif yaitu solusi teoritis yang paling baik kemudian dihitung bersama dengan solusi ideal negatif yaitu solusi teoritis yang paling buruk, menggunakan Rumus

2.16 diikuti dengan Rumus 2.17. Setelah didapatkan kedua solusi teoritis, maka dapat dihitung jarak setiap alternatif terhadap kedua solusi teoritis tersebut menggunakan Rumus 2.18 dan Rumus 2.19. Terakhir, koefisien kedekatan dihitung untuk setiap alternatif menggunakan Rumus 2.20, di mana nilai yang paling tinggi adalah alternatif yang paling dekat terhadap solusi ideal positif dan paling jauh dari solusi ideal negatif [35], [88].

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} {x_{ij}}^2}}$$

Rumus 2.15 Normalisasi Vektor
Sumber: [35], [88]

$${\lambda_j}^+ = max_i\{\lambda_{ij}\}$$

Rumus 2.16 Nilai Maksimum per Kriteria
Sumber: [35], [88]

$$A^+ = \{{\lambda_1}^+, {\lambda_2}^+ \dots {\lambda_m}^+\}$$

Rumus 2.17 Solusi Ideal Positif
Sumber: [35], [88]

$${D_j}^+ = \sqrt{\sum_{j=1}^{m} ({\lambda_{ij}}^+ - \lambda_{ij})^2}$$

Rumus 2.18 Jarak terhadap Solusi Ideal Positif
Sumber: [35], [88]

$${D_j}^- = \sqrt{\sum_{j=1}^{m} (A_{ij} - {A_{ij}}^-)^2}$$

Rumus 2.19 Jarak terhadap Solusi Ideal Negatif
Sumber: [35], [88]

$$T_j = \frac{{D_j}^-}{{D_j}^+ + {D_j}^-}$$

Rumus 2.20 Koefisien Kedekatan
Sumber: [35], [88]

j = Kriteria yang sedang dievaluasi

i = Alternatif yang sedang dievaluasi

m = Jumlah total kriteria

n = Jumlah total alternatif

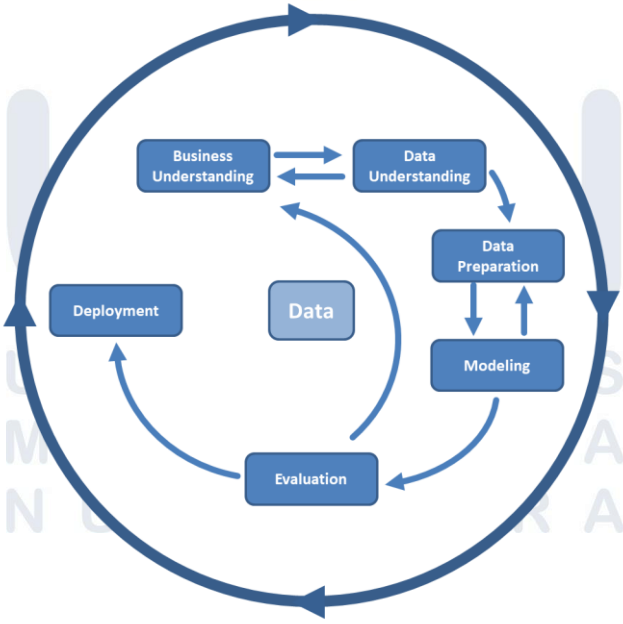
x = Nilai

2.3 *Framework* dan Algoritma

Subbab ini menjelaskan *framework* yang diikuti untuk melaksanakan penelitian ini, serta algoritma *neural network* yang diteliti. *Framework* yang digunakan dalam penelitian ini adalah CRISP-DM. Sedangkan, algoritma *neural network* yang diteliti adalah CNN dengan 3 arsitektur yaitu EfficientNetV2B0, MobileNetV3S dan SqueezeNet1_1.

2.3.1 *Framework* CRISP-DM

Kerangka kerja yang digunakan dalam penelitian ini adalah *Cross-Industry Standard Process for Data Mining*, disingkat sebagai CRISP-DM. CRISP-DM merupakan kerangka kerja yang dirancang pada tahun 1990-an oleh beberapa organisasi seperti Daimler-Chrysler, Teradata, SPSS dan OHRA. Kerangka kerja ini dibuat untuk menjadi pedoman bagi proyek *data science* yang memungkinkan proyek untuk dilaksanakan dengan lebih cepat, teratur, dan dapat diulang kembali [89]. Gambar 2.4 menunjukkan diagram kerangka kerja CRISP-DM yang terdiri dari 6 tahap.



Gambar 2.4 Kerangka Kerja CRISP-DM
Sumber: [90]

1. *Business Understanding*

Tahap pertama dari CRISP-DM bertujuan untuk memahami dan menetapkan terlebih dahulu masalah bisnis yang akan diselesaikan dengan proyek. Setelah masalah ditetapkan, dilihat bagaimana *data mining* atau *machine learning* dapat membantu untuk menyelesaikan masalah tersebut. Tahap ini juga menetapkan sumber daya yang akan dibutuhkan proyek untuk mencapai tujuannya [89].

2. *Data Understanding*

   Tahap kedua dari CRISP-DM bertujuan untuk memahami data yang dibutuhkan untuk proyek, kemudian melakukan proses pengambilan data tersebut. Data yang sudah diambil dianalisis secara sederhana untuk mendapatkan pemahaman dasar mengenai data tersebut seperti pola-pola yang ada dalam data dan kualitas data tersebut [89].

3. *Data Preparation*

   Tahap ketiga dari CRISP-DM adalah membersihkan dan mempersiapkan data untuk tahap selanjutnya berdasarkan pemahaman yang didapatkan pada tahap sebelumnya. Beberapa hal yang dapat dilakukan pada tahap ini contohnya menangani data duplikat dan kosong, serta menangani data *outlier* [89].

4. *Modeling*

   Tahap keempat dari CRISP-DM adalah membangun dan melatih model *machine learning* atau *deep learning* yang dapat membantu untuk mencapai tujuan proyek [89].

5. *Evaluation*

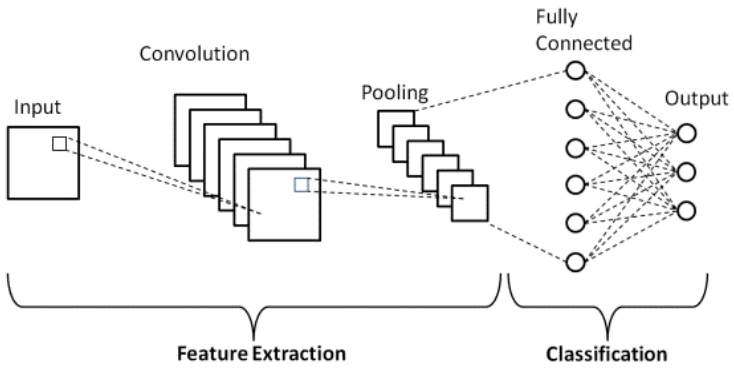
   Tahap kelima dari CRISP-DM adalah untuk mengevaluasi model yang telah dibangun untuk melihat kinerjanya pada data yang belum pernah dilihat oleh model sebelumnya. Metrik yang digunakan untuk mengevaluasi model harus sesuai dengan jenis model. Tahap ini juga melihat apakah model yang dibangun sudah memenuhi tujuan bisnis dari proyek [89].

6. *Deployment*

Tahap terakhir dari CRISP-DM bertujuan untuk mengimplementasikan model yang telah dibangun dan dievaluasi ke dalam lingkungan operasional model [89].

2.3.2 Algoritma *Convolutional Neural Network* (CNN)

Convolutional neural network, disingkat sebagai CNN, adalah salah satu jenis algoritma deep learning. CNN sudah menjadi salah satu algoritma *de facto* dalam bidang *computer vision* karena karakteristiknya seperti kemampuan untuk mengekstrasi fitur pada gambar secara otomatis serta kemampuan untuk mendeteksi fitur dalam gambar walaupun fitur tersebut memiliki posisi yang berbeda pada setiap gambar [91]. Gambar 2.5 menunjukkan arsitektur CNN pada umumnya.



Gambar 2.5 Arsitektur CNN
Sumber: [92]

1. Lapisan Konvolusi

   Lapisan ini mencari fitur yang penting dalam data dengan menghasilkan sebuah *feature map*. Untuk data gambar, sebuah *feature map* akan berisi fitur penting dalam gambar seperti tepi dan sudut objek, serta objek itu sendiri [91].

2. Lapisan *Pooling*

   Lapisan ini mengurangi dimensi dari *feature map* dengan tujuan untuk mempercepat operasi komputasi yang harus dilakukan. Lapisan ini menyimpan fitur-fitur yang paling penting dalam *feature map* dan membuang detail lainnya yang kurang penting [93].

3. Lapisan *Dropout*

Lapisan ini menonaktifkan beberapa *neuron* secara random untuk mengurangi *overfitting* pada model. Kesempatan suatu *neuron* dinonaktifkan oleh lapisan *dropout* dapat ditentukan [94]. Contohnya, lapisan *dropout* dengan kesempatan 50% berarti 50% dari *neuron* yang telah dikalkulasi akan dinonaktifkan.

4. Lapisan *Flatten*

   Lapisan ini mengubah *feature map* yang multi-dimensi menjadi vektor 1 dimensi agar dapat dibaca oleh lapisan *fully connected*. Lapisan ini merupakan langkah yang esensial karena lapisan *fully connected* tidak mendukung data multi-dimensi [93].

5. Lapisan *Fully Connected*

   Lapisan ini menghubungkan semua *neuron* yang telah dikalkulasi dengan satu sama lain (sehingga disebut *fully connected*) untuk kemudian membuat prediksi akhir dari data input [91].

2.3.2.2 EfficientNetV2B0

EfficientNet adalah sekelompok arsitektur CNN yang dirancang untuk mencapai akurasi prediksi beserta dengan efisiensi yang tinggi. EfficientNet didasari oleh prinsip *compound scaling* yang menyeimbangkan kedalaman lapisan, lebar kanal, dan resolusi gambar secara bersamaan. Melalui pendekatan ini, EfficientNet berhasil menjadi salah satu arsitektur pertama yang mampu mencapai akurasi tinggi dengan jumlah parameter yang jauh lebih sedikit dibandingkan arsitektur sebelumnya [95]. Namun, dalam implementasi praktisnya, EfficientNetV1 memiliki beberapa kelemahan sehingga EfficientNetV2 dirancang untuk mengatasi kelemahan EfficientNetV1. Tabel 2.3 menunjukkan kelemahan EfficientNet versi pertama dan bagaimana EfficientNetV2 menangani setiap kelemahan tersebut [96].

Tabel 2.3 Perbedaan EfficientNet dengan EfficientNetV2
Sumber: [95], [96]

Kelemahan EfficientNet Versi Pertama	Solusi dari EfficientNetV2
Menggunakan teknik *neural architecture search* (NAS) untuk mencari keseimbangan struktur dan	Menggunakan teknik NAS yang *training-aware* untuk memastikan

akurasi yang paling optimal, namun tidak memperhitungkan kecepatan pelatihan.	keseimbangan antara struktur, akurasi dan kecepatan pelatihan model.
Waktu pelatihan yang masih lama ketika input gambar yang digunakan memiliki resolusi yang besar, karena EfficientNet menggunakan *compound scaling*.	Menggunakan konsep *progressive learning* di mana pelatihan dimulai dengan input gambar beresolusi kecil dan pelan-pelan diperbesar, sehingga *epoch-epoch* awal bisa mempelajari fitur dasar secara efisien.
Lapisan konvolusi awal berupa blok MBConv yang berisi konvolusi *depthwise* dan *pointwise*, sehingga boros sumber daya komputasi pada lapisan-lapisan awal.	Lapisan konvolusi awal diganti dengan blok Fused-MBConv yang menggabungkan konvolusi *depthwise* dan *pointwise* sehingga sumber daya komputasi dapat dipakai dengan lebih efisien. Blok MBConv masih digunakan pada lapisan konvolusi yang lebih dalam karena lebih efisien untuk gambar yang lebih kecil.
Meningkatkan *scale* input pada tingkat yang sama untuk setiap tahap konvolusi, sehingga boros parameter pada lapisan-lapisan awal.	Menggunakan *stage-wise scaling* di mana konvolusi tahap awal memiliki lapisan dan kanal yang lebih sedikit daripada konvolusi tahap akhir, sehingga jumlah parameter hanya meningkat pada tahap akhir.

Perbaikan yang dibuat dengan arsitektur EfficientNetV2 untuk mengatasi kelemahan EfficientNet versi pertama membuat EfficientNetV2 mencapai skor akurasi yang sama atau lebih tinggi daripada EfficientNet versi pertama, dan sekaligus semakin mengurangi waktu yang dibutuhkan untuk pelatihan model [96]. EfficientNetV2B0 merupakan varian terkecil dari kelompok arsitektur EfficientNetV2 yang terdiri dari sekitar 7,1 juta parameter.

2.3.2.3 MobileNetV3S

MobileNetV3 adalah perkembangan dari MobileNetV2 dan MobileNetV1 yang dirancang untuk mengurangi beban komputasi CNN untuk implementasi dalam perangkat elektronik yang memiliki daya komputasi terbatas. MobileNetV3 menggunakan teknik-teknik yang sudah dipelopori oleh MobileNet versi sebelumnya, yaitu konvolusi *depthwise separable* (blok konvolusi yang terdiri dari konvolusi *depthwise* dan konvolusi *pointwise*) dari MobileNetV1 dan struktur *inverted residual* dari MobileNetV2 yang meningkatkan efisiensi komputasi [97].

MobileNetV3 menambahkan modul *squeeze-and-excitation* untuk menghemat jumlah parameter, menggunakan fungsi *activation hard-swish* (kalkulasi *swish* yang lebih mudah dihitung daripada *swish* konvensional), dan menggunakan teknik *neural architecture search* seperti EfficientNetV2 untuk mencari konfigurasi arsitektur yang paling optimal untuk memastikan waktu inferensi yang cepat [97]. MobileNetV3S merupakan varian terkecil dari kelompok arsitektur MobileNetV3 yang terdiri dari sekitar 2,5 juta parameter.

2.3.2.4 SqueezeNet1_1

SqueezeNet adalah arsitektur CNN yang dirancang untuk menghasilkan model CNN dengan ukuran terkecil tanpa mengurangi akurasi prediksi secara signifikan, sehingga tujuan utama dari SqueezeNet adalah untuk meningkatkan efisiensi model, bukan meningkatkan akurasi prediksi. Model SqueezeNet oleh karena itu pantas diimplementasikan dalam kasus di mana perangkat elektronik yang menjalankan model memiliki daya komputasi yang sangat terbatas [98].

SqueezeNet bekerja atas 3 strategi utama, yaitu:

1. Menggantikan filter berukuran 3x3 dengan filter berukuran 1x1, karena filter 1x1 memiliki jumlah parameter yang 9x lebih rendah daripada filter 3x3.
2. Mengurangi jumlah kanal input ke filter yang berukuran 3x3 menggunakan lapisan *squeeze* sehingga mengurangi jumlah parameter.
3. Melakukan *downsampling* hanya pada tahap akhir, sehingga model dapat mempelajari fitur yang lebih detail tanpa menambah jumlah parameter.

Untuk membantu memenuhi strategi SqueezeNet, maka dibuat *fire module* yang terdiri dari 2 bagian utama, yaitu:

1. Lapisan *squeeze*

Menggunakan konvolusi 1×1 untuk mengurangi jumlah channel input.

2. Lapisan *expand*

   Menggunakan kombinasi konvolusi 1×1 dan 3×3 untuk mengekstraksi fitur.

SqueezeNet juga unik karena tidak menggunakan lapisan *fully connected* seperti arsitektur CNN pada umumnya. Lapisan ini diganti dengan lapisan *global average pooling* untuk mendapatkan prediksi akhir. Perubahan lapisan ini juga dilakukan untuk mengurangi jumlah parameter model [98].

SqueezeNet1_1 merupakan penyempurnaan dari SqueezeNet versi awal (SqueezeNet1_0) yang memiliki jumlah parameter yang lebih sedikit dengan beban komputasi yang lebih ringan, namun mempertahankan akurasi yang sama [99].

2.4 Alat Penelitian

Subbab ini menjelaskan alat-alat yang dipakai selama penelitian ini. Ada 3 alat yang dipakai yaitu Python, PyTorch, dan ExecuTorch.

2.4.1 Python

Python adalah bahasa pemrograman yang dapat digunakan untuk mengembangkan berbagai macam perangkat lunak perangkat lunak. Python pertama dirintis oleh Guido van Rossum pada akhir 1980-an dan pertama dirilis pada tahun 1991 [100]. Sebagai bahasa pemrograman, Python memiliki sintaks yang sederhana dan mudah untuk dipahami, serta dapat dieksekusi pada berbagai macam sistem operasi seperti Windows dan macOS [101].

Komunitas Python yang telah berkembang seiring berjalannya waktu membuat Python menjadi pilihan yang sangat logis untuk proyek-proyek seperti analisis data, automasi pekerjaan, dan pengembangan *machine learning*. Hal ini dikarenakan Python memiliki banyak *libraries* pihak ketiga yang dipelihara oleh komunitas Python, di mana setiap *library* memiliki fungsi

unik yang membantu menyederhanakan pemrograman [100]. Gambar 2.6 menunjukkan logo Python.

Gambar 2.6 Logo Python
Sumber: [102]

2.4.1.1 PyTorch

PyTorch adalah salah satu *library machine learning* yang dikembangkan untuk Python, pertama dirilis pada tahun 2016. Untuk proyek *machine learning* dan *deep learning*, PyTorch sangat membantu untuk mempercepat pelatihan model karena dukungannya terhadap penggunaan CUDA pada GPU NVIDIA. Fitur ini memungkinkan PyTorch untuk menggunakan kapasitas komputasi masif yang dimiliki oleh GPU NVIDIA untuk pelatihan dan eksekusi model *machine learning*, daripada menggunakan CPU yang relatif jauh lebih lambat untuk tujuan *machine learning* [103].

Selain PyTorch sebagai *library* sendiri, PyTorch juga memiliki *library* dukungan seperti TIMM (Torch Image Models) yang memberikan akses instan terhadap berbagai macam model *machine learning* yang berkualitas yang sudah dikembangkan oleh komunitas PyTorch, sehingga menyederhanakan proses pelatihan model dengan *transfer learning* [103]. Gambar 2.7 menunjukkan logo PyTorch.

Gambar 2.7 Logo PyTorch
Sumber: [104]

2.4.1.2 ExecuTorch

ExecuTorch adalah sekelompok *library machine learning* berbasis PyTorch yang bertujuan untuk memungkinkan eksekusi model

PyTorch pada perangkat keras portabel seperti telepon genggam dan perangkat tepi [105]. *Library* ini secara spesifik dirancang untuk mempercepat proses *deployment* model dari lingkungan pengembangan (*development*) ke lingkungan produksi pada perangkat keras portabel atau perangkat tepi. ExecuTorch merupakan perkembangan lanjut dari PyTorch Mobile yaitu proyek PyTorch pada perangkat tepi sebelumnya. Dibandingkan PyTorch Mobile, ExecuTorch memiliki penggunaan memori yang lebih efisien sehingga mengurangi beban komputasi pada perangkat keras tujuan [106]. Gambar 2.8 menunjukkan logo ExecuTorch.

Gambar 2.8 Logo ExecuTorch
Sumber: [106]

2.4.1.3 Optuna

Optuna (logo pada Gambar 2.9) adalah *library* Python yang bertujuan untuk memudahkan proses *hyperparameter tuning* melalui antarmuka yang bertingkat tinggi dan dapat dipasangkan dengan berbagai macam *library machine learning* seperti PyTorch. Optuna juga mempercepat waktu yang dibutuhkan untuk menemukan *hyperparameter* terbaik melalui pendekatan *Bayesian optimization* (BO). BO sendiri bekerja atas dua konsep yaitu eksplorasi dan eksploitasi. Eksplorasi bertujuan untuk mencari hasil dari kombinasi *hyperparameter* tertentu yang belum dicoba, sedangkan eksploitasi bertujuan untuk mencari hasil dari kombinasi sekitar *hyperparameter* yang sudah dicoba dan diketahui memiliki hasil yang baik [107], [108].

Algoritma *Bayesian optimization* spesifik yang dipakai oleh Optuna dalah *Tree-structured Parzen Estimator* (TPE). Algoritma ini pertama menguji beberapa kombinasi *hyperparameter* kemudian memisahkan kombinasi *hyperparameter* dengan hasil baik dan

kombinasi *hyperparameter* dengan hasil buruk. Berdasarkan persebaran kedua kelompok ini, TPE akan mencoba kombinasi *hyperparameter* baru di sekitar kelompok kombinasi yang diketahui memiliki hasil baik karena kesempatan untuk mendapatkan hasil yang lebih baik lebih besar di kawasan tersebut [107], [108].

Gambar 2.9 Logo Optuna
Sumber: [109]

2.4.2 Kaggle

Kaggle adalah platform kompetisi proyek *machine learning* yang pertama dimulai pada tahun 2010 oleh Anthony Goldbloom. Kaggle mencapai 1 juta pengguna pada tahun 2017 dan juga dibeli oleh Google pada tahun yang sama, sehingga Kaggle sekarang menjadi salah satu portofolio produk Google [110].

Selain menjadi salah satu platform kompetisi *machine learning* terbesar di dunia, Kaggle juga memberikan sumber daya komputasi terbatas yang gratis untuk digunakan penggunanya. Sumber daya komputasi ini berbentuk sebuah *virtual machine* yang dialokasikan ke satu pengguna melalui internet dengan abstraksi antarmuka *notebook*. Untuk tujuan *machine learning*, sumber daya komputasi yang sering kali menjadi relevan adalah GPU, sehingga Kaggle juga mengalokasikan GPU yang memadai kepada pengguna [110]. Gambar 2.10 menunjukkan logo Kaggle.

Gambar 2.10 Logo Kaggle
Sumber: [111]

2.4.3 Android Studio

Android Studio adalah perangkat lunak *integrated development environment* (IDE) yang dirancang untuk mengembangkan aplikasi untuk sistem operasi Android. Android Studio merupakan varian dari IDE IntelliJ IDEA, sebuah IDE untuk bahasa pemrograman Java dan Kotlin yang

dikembangkan oleh JetBrains. Sebagai IDE, Android Studio sudah mengintegrasikan *software development kit* (SDK) Android bersama dengan Gradle untuk manajemen *build* aplikasi. Android Studio juga memiliki *emulator* untuk menguji aplikasi yang sedang dikembangkan secara langsung pada mesin pengembangan [112]. Fitur dan integrasi yang dimiliki oleh Android Studio memungkinkan pengembangan aplikasi Android yang lebih cepat dan praktis. Gambar 2.11 menunjukkan *wordmark* dan logo Android Studio.

Gambar 2.11 *Wordmark* dan Logo Android Studio
Sumber: [113]