BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Kajian Pustaka

Kemajuan dalam teknologi *deep learning* telah mempengaruhi berbagai bidang teknologi saat ini, terutama dalam klasifikasi dan deteksi objek pada gambar. Banyak penelitian telah dilakukan untuk memanfaatkan performa *deep learning* dalam menyelesaikan berbagai permasalahan yang ada. Dalam penelitian sebelumnya, telah ditemukan bahwa *deep learning*, yang didasarkan pada konsep meniru otak manusia, mampu meningkatkan akurasi dalam pengenalan gambar dalam konteks aplikasi deteksi dan klasifikasi objek. Dalam tulisan ini, peneliti akan menjelaskan temuan dari tinjauan terhadap penelitian-penelitian sebelumnya.

Penelitian pertama tentang Deteksi Hama Pada Daun Apel Menggunakan Algoritma *Convolutional Neural Network* yang dilakukan oleh [23] pada tahun 2022. Tujuan utama dari penelitian tersebut adalah untuk mengidentifikasi keberadaan hama pada buah apel. Dalam penelitian ini, digunakan *Convolutional Neural Network* (CNN), yaitu model *deep learning* yang umum digunakan untuk klasifikasi objek dalam gambar. Data yang digunakan diambil dari situs Kaggle. Hasil dari pelatihan dan pengujian klasifikasi gambar, baik yang terinfeksi hama maupun yang tidak, menggunakan metode CNN, menunjukkan akurasi yang tinggi, yaitu 99.66%, dengan parameter tertentu seperti jumlah epoch sebanyak 60, resolusi gambar 256x256, dan teknik augmentasi data seperti *horizontal flip, vertical flip,* dan rotasi acak.

Penelitian kedua tentang Arsitektur *Convolutional Neural Network* (CNN) *Alexnet* Untuk Klasifikasi Hama Pada Citra Daun Tanaman Kopi yang dilakukan oleh [24] pada tahun 2021. Penelitian tersebut memiliki tujuan melakukan klasifikasi hama pada citra daun tanaman kopi. *Dataset* yang digunakan sebanyak 1560 gambar daun kopi robusta dengan tungau dan bintik-bintik yang terlihat (menunjukkan keberadaan karat daun kopi) untuk kasus infeksi dan gambar tanpa struktur seperti itu untuk kasus sehat. Selain itu, kumpulan data mencakup anotasi mengenai keadaan (sehat dan tidak sehat) dan tingkat keparahan penyakit (area

daun dengan bintik-bintik). Metode yang digunakan pada penelitian tersebut menggunakan CNN yang merupakan model *deep learning* pada khususnya arsitektur *Alexnet*. Data diperoleh melalui website *mendeley* data. Hasil pelatihan dan pengujian untuk proses klasifikasi gambar hama dan tanpa hama dengan menerapkan *Convolutional Neural Network* khususnya arsitektur *Alexnet* didapatkan akurasi 81.6%.

Penelitian ketiga tentang Implementasi *Deep Learning* pada Sistem Klasifikasi Hama Tanaman Padi Menggunakan *Convolutional Neural Network* (CNN) [25] pada tahun 2022. Studi ini bertujuan mengklasifikasikan hama pada tanaman padi menggunakan metode CNN. Sumber data yang digunakan adalah data sekunder yang diperoleh dari Google dan Instagram, dengan total 1.065 gambar yang terbagi dalam lima kategori hama. Hasil penelitian menunjukkan bahwa sistem klasifikasi hama pada tanaman padi ini mencapai akurasi paling tinggi 77.33%, dengan distribusi data sebesar 90% untuk *training* dan 10% untuk *testing*.

Penelitian keempat tentang *Implementation of Convolutional Neural Network Algorithm to Pest Detection in Caisim* [26] pada tahun 2023. Penelitian tersebut bertujuan untuk mendeteksi hama pada tanaman sawi hijau dengan cara klasifikasi gambar menggunakan metode *convolutional neural network* (CNN). *Dataset* yang digunakan pada penelitian ini menggunakan data primer sebanyak 1000 data yang dibagi menjadi 500 tanpa hama dan 500 dengan hama. Hasil penelitian dan pengujian tersebut mendapatkan akurasi sebesar 92% pada percobaan ketiga tanpa menggunakan *transfer learning*.

Penelitian kelima tentang *Transfer Learning* untuk Klasifikasi Penyakit dan Hama Padi Menggunakan *MobileNetV2* [27] pada tahun 2023. Penelitian ini bertujuan untuk mengklasifikasikan penyakit pada tanaman padi ke dalam sembilan kategori, dengan jumlah citra yang beragam untuk setiap kategori, termasuk hama dan penyakit. Penggunaan model CNN sederhana dan berbagai arsitektur *transfer learning* seperti *MobileNetV2*, *InceptionV3*, *NasnetMobile*, VGG16, dan *EfficientNetB7* diimplementasikan untuk membedakan antara berbagai penyakit dan hama. Dari hasil pengujian, *MobileNetV2* terbukti cukup efektif dalam mengidentifikasi berbagai hama dan penyakit pada tanaman padi.

Penelitian terakhir tentang Rancang Bangun Aplikasi Identifikasi Penyakit Tanaman Pepaya *California* Berbasis Android Menggunakan Metode CNN Model Arsitektur *SqueezeNet* [28] pada tahun 2021. Penelitian ini berfokus pada pengembangan aplikasi berbasis Android untuk mengidentifikasi penyakit pada tanaman pepaya *California*. Metode yang digunakan adalah *convolutional neural network* dengan arsitektur *SqueezeNet*. Aplikasi yang dibangun berhasil mengenali penyakit Antraknosa, Ringspot Virus, serta kondisi tanaman pepaya yang sehat. Dalam pengujian, model *SqueezeNet* menunjukkan akurasi 97% dalam mendeteksi penyakit pada daun dan 70% pada buah.

Tabel 2. 1 Penelitian Terdahulu

No.	Peneliti, Tahun	Judul	Masalah	Perbedaan	Metode	Hasil
1.	Husen, Dede Kusrini, Kusrini Kusnawi, Kusnawi. Tahun 2022	Deteksi Hama Pada Daun Apel Menggunakan Algoritma *Convolutional Neural Network*	Hama pada tanaman apel seperti *Apple Scab*, *Black Rot*, dan Cedar/*Apple Rust* mengurangi produksi dan kualitas buah, dan mempengaruhi ekonomi petani. Pencarian solusi lebih efisien untuk mengendalikan hama pada tanaman apel termasuk penggunaan teknologi, karena metode penggunaan pestisida saat ini berdampak negatif pada lingkungan dan kesehatan, dan terdapat keterbatasan pakar hama.	Dataset diambil dari Kaggle dengan label ada hama ulat dan tanpa hama ulat pada daun sawi hijau, menggunakan arsitektur CNN *MobileNetV2* dengan metode *transfer learning*.	*Convolutional Neural Network*	Hasil dari penelitian tersebut dengan menggunakan metode *Convolutional Neural Network* (CNN) memiliki akurasi 99.66% dengan parameter *epoch* 60, ukuran citra 256x256 serta penerapan teknik augmentasi data *horizontal flip, vertical flip* dan *random rotation*.
2.	Irfansyah, Dicki Mustikasari, Metty Suroso, Amat. Tahun 2021	Arsitektur *Convolutional Neural Network* (CNN) *Alexnet* Untuk Klasifikasi Hama Pada Citra Daun Tanaman Kopi	Produksi kopi Indonesia rendah karena dua penyebab utama: penyakit pada pohon kopi tua dan kurangnya peremajaan tanaman kopi. Produksi ini hanya mampu memenuhi 4% kebutuhan kopi Uni Eropa. Masalahnya adalah produktivitas kopi Indonesia dan upaya untuk memecahkan masalah ini dengan teknologi.	Menggunakan arsitektur CNN *MobileNetV2* untuk klasifikasi hama ulat pada citra daun sawi hijau.	*Convolutional Neural Network* berbasis arsitektur *Alexnet*	Hasil dari penelitian tersebut dengan menggunakan metode *Convolutional Neural* Network (CNN) berbasis arsitektur *Alexnet* yang diuji dengan *confusion matrix* memiliki akurasi 81.6%.
3.	Yuliany, Susi Aradea Rachman, Andi Nur. Tahun 2022	Implementasi *Deep Learning* pada Sistem Klasifikasi Hama Tanaman	Salah satu masalah utama dalam penelitian ini adalah pengenalan objek dalam industri pertanian dan kurangnya akurasi dan efisiensi dalam pengendalian hama tanaman. Petani sering kali menyemprot pestisida tanpa	Membuat sistem klasifikasi hama ulat pada daun sawi hijau	*Convolutional Neural Network*	Hasil dari penelitian tersebut dengan menggunakan metode *Convolutional Neural*

No.	Peneliti, Tahun	Judul	Masalah	Perbedaan	Metode	Hasil
		Padi Menggunakan Metode *Convolutional Neural Network* (CNN)	mempertimbangkan dosis, waktu, metode, dan sasaran yang tepat. Ini dapat membahayakan organisme lain selain hama target dan mengganggu musuh alami hama.	menggunakan metode *transfer learning.*		*Network* (CNN) memiliki akurasi tertinggi 77.33% dengan pembagian data 90:10.
4.	C. L. Nazalia, P. Palupiningsih, B. Prayitno, Y. S. Purwanto. Tahun 2023	*Implementation of Convolutional Neural Network Algorithm to Pest Detection in Caisim*	Masalah ini disebabkan oleh pertumbuhan sektor pertanian yang positif meskipun pandemi Covid-19 menyebabkan ekonomi nasional menurun. Pada kuartal keempat tahun 2020, sektor pertanian tumbuh 2,59 persen tahun ke tahun, menurut data dari Badan Pusat Statistik (BPS). Sayuran daun yang populer di Indonesia, *caisim*, atau *green mustard*, adalah salah satu produk ekspor utama. Namun, budidaya *caisim* menghadapi tantangan karena ada hama yang dapat merusak tanaman.	Mengimplementasi arsitektur CNN MobileNetV2 dengan metode *transfer learning*, melakukan *deployment* ke dalam aplikasi android.	*Convolutional Neural Network*	Hasil dari penelitian tersebut dengan menggunakan metode *Convolutional Neural Network* (CNN) dengan tiga kali percobaan mendapatkan akurasi sebesar 92%
5.	Virgantara Putra, Oddy Zaim Mustaqim, Muhammad Muriatmoko, Dihin	*Transfer Learning* untuk Klasifikasi Penyakit dan Hama Padi Menggunakan MobileNetV2	Peranan tanaman padi sangat krusial dalam memastikan kecukupan pangan di Indonesia. Namun, ada berbagai faktor yang berpengaruh terhadap hasil panen padi, di antaranya adalah serangan penyakit dan hama.	Membuat sistem klasifikasi hama ulat pada daun sawi hijau yang diterapkan pada aplikasi android.	*NasNetMobile*, *InceptionV3*, VGG16, *EfficientNetB7*, *simple* CNN, *MobileNetV2*.	Dalam penelitian ini, model CNN dengan *transfer learning* menggunakan *MobileNetV2* cukup efektif untuk menemukan penyakit pada tanaman padi dengan akurasi sebesar 96%.

No.	Peneliti, Tahun	Judul	Masalah	Perbedaan	Metode	Hasil
6.	Angga Irawan, Ferry Sudarma, Made Care Khrisne, Duman. Tahun 2021	Rancang Bangun Aplikasi Identifikasi Penyakit Tanaman Pepaya *California* Berbasis Android Menggunakan Metode CNN Model Arsitektur *SqueezeNet*	Penyakit tanaman sering terjadi di Kebun Percobaan Pertanian Universitas Udayana, terutama di hortikultura, yang mengurangi hasil produksi. Oleh karena itu, penting untuk mengidentifikasi penyakit tanaman sejak dini.	Merancang dan membangun aplikasi berbasis android menggunakan metode *transfer learning* pada arsitektur CNN *MobileNetV2* untuk mendeteksi keberadaan hama ulat pada daun sawi hijau.	*Convolutional Neural Network* Berbasis Arsitektur *SqueezeNet*	Hasil dari penelitian tersebut dengan menggunakan metode *convolutional neural network* berbasis arsitektur *squeezeNet* memperoleh akurasi sebesar 97% pada daun dan 70% pada buah dalam mendeteksi penyakit pepaya *California*.

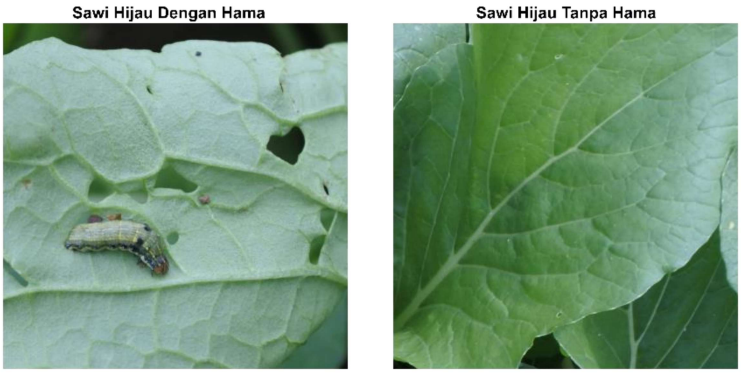
Berdasarkan Tabel 2.1 penelitian terdahulu, dapat ditarik kesimpulan bahwa penelitian mengenai deteksi hama ulat pada tanaman sayur sawi hijau berbasis android menggunakan *Convolutional Neural Network* (CNN) arsitektur *MobileNetV2* dengan metode *transfer learning* belum pernah dilakukan sebelumnya. Berdasarkan Tabel tersebut penelitian terdahulu rata-rata menggunakan *Convolutional Neural Network* (CNN) sebagai metode dalam penelitian yang mampu mengklasifikasi gambar dengan akurasi yang tinggi, serta mudah diintegrasikan ke dalam perangkat bergerak dengan bantuan *TensorFlow*. Dalam penelitian ini, penelitian nomor 4 dan 5 digunakan sebagai acuan karena memiliki tujuan yang sama. Perbedaan pada penelitian tersebut terletak pada arsitektur model, studi kasus, dan perangkat *deployment* yang digunakan.

2.2 Landasan Teori

Pada landasan teori ini mengkaji tentang beberapa teori yang digunakan pada penelitian ini yaitu sebagai berikut :

2.2.1 Organisme Pengganggu Tanaman (OPT) pada Sayur Sawi Hijau

Pengembangan tanaman sawi hijau sendiri mengalami beberapa kendala yang disebabkan oleh faktor-faktor tertentu. Dalam konteks ini, salah satu faktor penting adalah gangguan yang diakibatkan oleh Organisme Pengganggu Tanaman (OPT), yang memiliki potensi untuk menghambat produksi sawi hijau baik dalam hal kualitas maupun kuantitas. Tingkat kerusakan yang disebabkan oleh OPT pada tanaman sawi hijau bervariasi, mencapai 32,2% hingga 50,24% [7]. Terdapat enam jenis hama utama yang sering menyerang tanaman sawi hijau, termasuk kumbang daun, lalat penggerek daun (*Liriomyza sp.*), ulat tritip (*Plutella xylostella L.*), ulat titik tumbuh (*Crocidolomia binotalis*), ulat grayak (Spodoptera litura), dan ulat jengkal (*Plusia spp.*) [7]. Serangan hama yang parah dapat berakibat pada gagal panen dan mengakibatkan kerugian yang signifikan, berdampak negatif terhadap perekonomian para petani yang mengusahakan tanaman sawi hijau.



Gambar 2. 1 *Sample Dataset*

2.2.2 Ekstraksi Ciri

Dalam pemrosesan citra, ekstraksi ciri sangat penting, terutama dalam CBIR (*Content Based Image Retrieval*). Dalam situasi ini, ekstraksi ciri merepresentasi n-dimensi yang mengkodekan isi gambar untuk diindeks dalam basis data gambar. Komponen ekstraksi ciri ini kemudian digunakan untuk menganalisis dan memproses gambar untuk membandingkannya dengan gambar lain. Salah satu karakteristik yang sering digunakan dalam proses ini adalah tekstur, yang memberikan karakteristik khusus pada gambar karena struktur permukaannya mengandung informasi penting. Tekstur ini sangat penting untuk CBIR karena kemampuannya untuk mendeskripsikan informasi visual yang unik dari gambar. [29]. Karakteristik ekstraksi ciri lainnya adalah bentuk yang akan digunakan juga pada penelitian ini yang mana akan diberikan label ada hama apabila terdapat bentuk hama ulat pada gambar tersebut.

2.2.3 *Deep Learning*

Konsep *Deep Learning* (DL) berawal dari sebuah makalah yang diterbitkan dalam *Science* oleh Hinton et al. pada tahun 2006 [30]. Ide dasar dari *deep learning* adalah menggunakan jaringan saraf untuk analisis data dan pembelajaran fitur, fitur data adalah diekstraksi oleh beberapa lapisan tersembunyi, setiap lapisan tersembunyi dapat dianggap sebagai *perceptron*, *perceptron* digunakan untuk mengekstrak fitur tingkat rendah, lalu menggabungkan fitur tingkat rendah untuk mendapatkan fitur tingkat tinggi abstrak, yang bisa secara signifikan mengatasi masalah minimum lokal. *Deep Learning* mengatasi kelemahan yang mengandalkan

algoritma tradisional pada fitur yang dirancang secara manual dan telah menarik lebih banyak perhatian peneliti. Sekarang telah berhasil diterapkan dalam *computer vision*, *pattern recognition*, *speech recognition*, *natural language processing* dan *recommendation system* [31].

Metode klasifikasi dan pengenalan gambar tradisional dari fitur desain manual hanya dapat mengekstraksi fitur yang mendasarinya, dan sulit untuk mengekstraksi yang dalam dan informasi fitur gambar yang kompleks [30]. Dan metode *deep learning* dapat mengatasi permasalahan ini. Bisa langsung melakukan pembelajaran tanpa pengawasan dari gambar aslinya untuk mendapatkan informasi fitur gambar *multi-level* seperti fitur tingkat rendah, fitur menengah dan tingkat tinggi fitur semantik. Algoritma deteksi penyakit dan hama tanaman tradisional terutama mengadopsi pengenalan gambar metode fitur yang dirancang manual, yang sulit dan bergantung pada pengalaman dan keberuntungan, dan tidak dapat secara otomatis mempelajari dan mengekstrak fitur dari gambar aslinya. Sebaliknya, *deep learning* bisa secara otomatis pelajari fitur dari data besar tanpa manipulasi manual. Model ini terdiri dari beberapa lapisan, yang memiliki kemampuan belajar otonom yang baik dan kemampuan ekspresi fitur, dan dapat secara otomatis mengekstraksi fitur gambar untuk klasifikasi dan pengenalan gambar. Oleh karena itu, dalam belajar dapat memainkan peran besar dalam bidang penyakit tanaman dan pengenalan gambar hama. Saat ini, metode *deep learning* telah mengembangkan banyak model *deep neural network* yang terkenal, termasuk *deep belief network* (DBN), *deep Boltzmann machine* (DBM), *stack de-noising autoencoder* (SDAE) dan *deep convolutional neural network* (CNN) [32]. Di bidang pengenalan gambar, penggunaan model *deep neural network* ini untuk mewujudkan ekstraksi fitur otomatis dari fitur dimensi tinggi ruang menawarkan keuntungan yang signifikan dibandingkan metode ekstraksi fitur desain manual tradisional. Selain itu, sebagai jumlah sampel pelatihan tumbuh dan komputasi daya meningkat, kekuatan karakterisasi *deep neural network* semakin ditingkatkan. Saat ini, ledakan *deep learning* melanda industri dan akademisi, dan kinerja model *deep neural network* semua secara signifikan di depan model tradisional. Dalam

beberapa tahun terakhir, kerangka kerja *deep learning* yang paling populer adalah *deep convolutional neural network*.

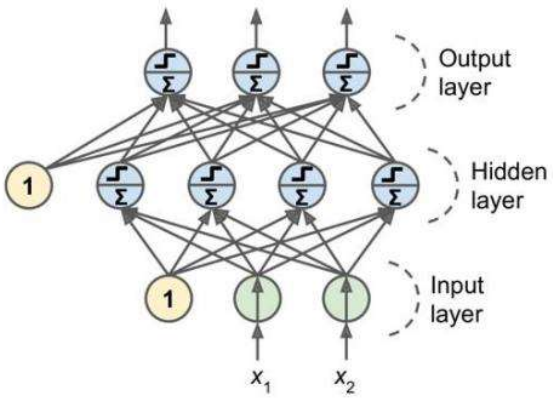
2.2.4 *Artificial Neural Network*

Sejalan dengan itu, terdapat juga teknologi machine learning yang memanfaatkan konsep jaringan saraf tiruan. Jaringan saraf tiruan, juga dikenal sebagai *Artificial Neural Network* (ANN), merupakan sebuah model dalam bidang machine learning yang terinspirasi oleh cara kerja saraf dalam otak manusia. ANN merupakan salah satu model *machine learning* yang memiliki fleksibilitas dan kekuatan dalam menangani berbagai masalah yang kompleks. Model ini mampu mengklasifikasikan jumlah gambar yang sangat besar, mengenali berbagai bahasa di dunia, memberikan rekomendasi video kepada jutaan pengguna, bahkan belajar untuk mengalahkan juara dunia dalam permainan papan GO [15].

Dengan kemampuan yang dimiliki oleh ANN, teknologi ini menjadi sangat relevan dan efektif dalam menghadapi tantangan-tantangan di bidang *machine learning* yang semakin kompleks dan skala yang semakin besar.

2.2.5 *Multi-Layer Perceptron*

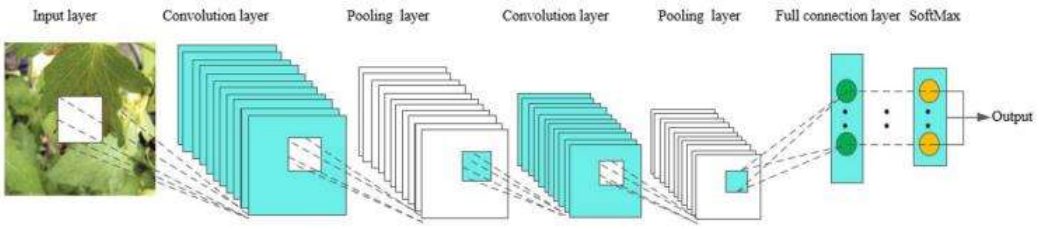
Multi Layer Perceptron (MLP) adalah jenis jaringan saraf yang terdiri dari beberapa lapisan. Jaringan ini terdiri dari lapisan *input*, satu atau lebih lapisan tersembunyi, dan satu lapisan *output*. MLP dikembangkan dari *perceptron* [15]. MLP memiliki kemampuan untuk menyelesaikan masalah yang membutuhkan pembelajaran dan penelitian di bidang ilmu saraf komputasi dan pemrosesan paralel. Dapat dilihat contoh dari model MLP pada Gambar 2.2.



Gambar 2. 2 Model *Multi Layer Perception* [15]

2.2.6 *Convolutional Neural Network*

Convolutional Neural Networks, disingkat CNN, memiliki struktur jaringan yang kompleks dan dapat melakukan operasi konvolusi. Seperti yang ditunjukkan pada Gambar 2.3, model *convolutional neural network* terdiri dari *input* layer, *convolution layer*, *pooling layer*, *full connection layer* dan *output layer*. Dalam satu model, *convolution layer* dan *pooling layer* bergantian beberapa kali, dan ketika *neuron* dari *convolutional layer* terhubung ke *neuron* dari *pooling layer*, tidak diperlukan *full connection*. CNN adalah model populer di bidang *deep learning*. Alasannya terletak pada kapasitas model yang sangat besar dan informasi kompleks yang dibawa oleh karakteristik struktural dasar CNN, yang memungkinkan CNN memainkan keuntungan dalam pengenalan gambar. Pada saat yang sama, keberhasilan CNN dalam bidang *computer vision* telah mendorong pertumbuhan popularitas *deep learning*.

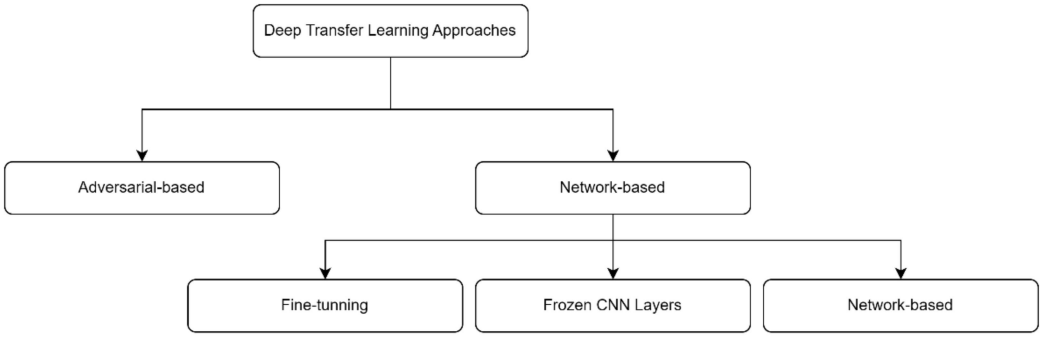


Gambar 2. 3 Struktur Dasar CNN [30]

Di *convolution layer*, inti konvolusi didefinisikan pertama. Inti konvolusi dapat dianggap sebagai sebuah *local receptive field*, dan *local receptive field* adalah keuntungan terbesar dari CNN. Saat memproses informasi data, inti konvolusi meluncur pada peta fitur untuk mengekstrak bagian dari informasi fitur. Setelah ekstraksi fitur dari *convolution layer*, *neuron* dimasukkan ke dalam *pooling layer* untuk mengekstraksi fitur lagi. Saat ini, metode *pooling* yang umum digunakan termasuk menghitung nilai rata-rata, maksimum dan acak dari semua nilai dalam *local receptive field* [33], [34]. Setelah data memasuki beberapa *convolution layer* dan *pooling layer*, mereka memasuki *full connection layer*, dan *neuron* di *full connection layer* sepenuhnya terhubung dengan *neuron* di *upper layer*. Akhirnya, data dalam *full connection layer* dapat diklasifikasikan berdasarkan metode *softmax*, dan kemudian nilai ditransmisikan ke *output layer* untuk hasil keluaran.

2.2.7 *Transfer Learning*

Transfer learning adalah metode dalam bidang *machine learning* yang memanfaatkan pengetahuan yang diperoleh oleh model dari tugas tertentu untuk diaplikasikan pada tugas lain yang sejenis [35]. Metode ini bertujuan untuk meningkatkan kecepatan pelatihan model baru dengan menggunakan pengetahuan dari model yang telah dilatih sebelumnya dengan dataset yang lebih besar atau lebih lengkap. Keunggulan dari *transfer learning* terletak pada kemampuannya mengurangi kebutuhan akan dataset besar dan waktu pelatihan untuk menghasilkan model *machine learning* yang akurat. Hal ini sangat bermanfaat dalam situasi di mana model *machine learning* harus diaplikasikan pada domain baru atau ketika dataset yang tersedia untuk pelatihan terbatas.



Gambar 2. 4 Pendekatan *Deep Transfer Learning* yang Paling Umum

Dalam *Deep Transfer Learning*, metode pendekatan utama pertama adalah *fine-tuning*, di mana model yang telah dilatih pada dataset serupa disesuaikan untuk data target. Ini terkenal karena kemudahan penerapan dan efektivitasnya dalam memangkas biaya pelatihan serta mengatasi keterbatasan dataset target. Metode kedua melibatkan *freezing* lapisan CNN pada model terlatih dan penyesuaian pada *fully connected layer* untuk klasifikasi spesifik data target, memanfaatkan lapisan CNN untuk ekstraksi fitur [22].

2.2.8 *MobileNetV2*

MobileNetV2 adalah evolusi dari *MobileNetV1*, diperkenalkan pertama kali pada tahun 2018. Inovasi utama dalam arsitektur ini adalah pengenalan *Linear Bottlenecks* dan *Inverted Residual*. *Linear Bottlenecks* bertujuan untuk mengurangi

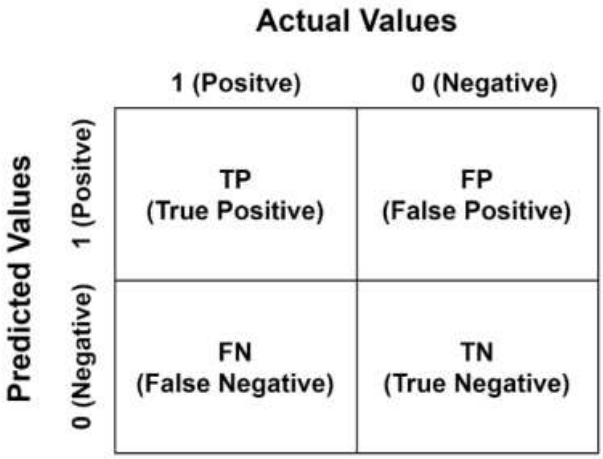
dimensi ruang fitur, sehingga mengurangi beban komputasi sehingga meningkatkan efisiensi penggunaan memori. *Inverted Residuals*, di sisi lain, menggabungkan *shortcut connections* dengan *Depthwise Separable Convolutions*. Ini memungkinkan jaringan untuk mempertahankan informasi penting melalui lapisan dan secara signifikan mengurangi kehilangan informasi [36].

Tabel 2. 2 Arsitektur *MobileNetV2*

Input	Operator	*t*	*c*	*n*	*s*
224^2 x 3	conv2d	-	32	1	2
112^2 x 32	*bottleneck*	1	16	1	1
112^2 x 16	*bottleneck*	6	24	2	2
56^2 x 24	*bottleneck*	6	32	3	2
28^2 x 32	*bottleneck*	6	64	4	2
14^2 x 64	*bottleneck*	6	96	3	1
14^2 x 96	*bottleneck*	6	160	3	2
7^2 x 160	*bottleneck*	6	320	1	1
7^2 x 320	conv2d 1x1	-	1280	1	1
7^2 x 1280	avgpool 7x7	-	-	1	-
1 x 1 x 1280	conv2d 1x1	-	k	-	

2.2.9 *Confusion Matrix*

Confusion Matrix, sebagai salah satu Teknik yang digunakan untuk mengukur kinerja model dalam kasus klasifikasi [37]. Untuk mengukur performa model, terdapat sejumlah kriteria yang tercantum dalam *confusion matrix*. Matriks ini menghasilkan empat nilai, yakni *True Positif* (TP), *True Negatif* (TN), *False Positif* (FP), dan *False Negatif* (FN), ilustrasi ini dapat dilihat pada Gambar 2. 6.



Gambar 2. 5 Tabel *Confusion Matrix* 2 Kelas

Pada Gambar 2.6 terlihat ilustrasi Tabel *confusion matrix* yang terdiri dari dua kelas. Tabel ini menggambarkan nilai yang sebenarnya dan nilai yang di klasifikasi. Di dalam Tabel ini, terdapat beberapa elemen yang dapat diukur nilainya, yang akan dijelaskan berikut ini:

1. *True Positif* (TP) menunjukkan jumlah anggota class 1 yang berhasil diprediksi dengan benar.
2. *True Negatif* (TN) menunjukkan jumlah anggota class 1 yang gagal diprediksi dengan benar.
3. *F.alse Positif* (FP) menunjukkan jumlah anggota class 0 yang berhasil diprediksi dengan benar.
4. *False Negatif* (FN) menunjukkan jumlah anggota class 0 yang gagal diprediksi dengan benar.

Elemen-elemen tersebut berguna dalam mengukur nilai *accuracy*, *recall*, *precision*, dan F1-*Score* dalam sebuah model. Informasi ini membantu dalam mengevaluasi apakah model yang dirancang tersebut baik atau tidak.

a. *Accuracy*

Akurasi merupakan metrik yang mengevaluasi sejauh mana model mampu mengklasifikasikan data secara tepat. Metrik ini dihitung dengan menjumlahkan prediksi yang benar (*True Positive* dan *True Negative*) dibandingkan dengan total jumlah prediksi yang dibuat oleh model. Akurasi memberikan pandangan umum tentang kinerja model dalam mengklasifikasikan data. Namun, dalam beberapa

situasi, terutama pada dataset yang tidak seimbang atau dengan kelas yang berbeda-beda, akurasi mungkin tidak selalu menjadi ukuran yang tepat:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

b. *Recall*

Recall adalah ukuran efektivitas model dalam mengidentifikasi kasus positif. Metrik ini dihitung dengan membandingkan jumlah *True Positive* dengan total kasus positif sebenarnya. *Recall* memberikan wawasan tentang kemampuan model dalam menangkap kasus positif dan biasanya penting dalam situasi di mana mengidentifikasi semua kasus positif adalah krusial :

$$recall = \frac{TP}{TP + FN} \tag{2.2}$$

c. *Precision*

Precision merupakan ukuran efektivitas model dalam mengidentifikasi data sebagai kelas positif secara akurat. Metrik ini dihitung dengan membandingkan jumlah *True Positive* dengan total prediksi positif oleh model. *Precision* menggambarkan kemampuan model dalam menghindari kesalahan klasifikasi data sebagai positif ketika sebenarnya bukan, dan sangat penting dalam situasi di mana penting untuk meminimalisir kesalahan positif.:

$$precision = \frac{TP}{TP + FP} \tag{2.3}$$

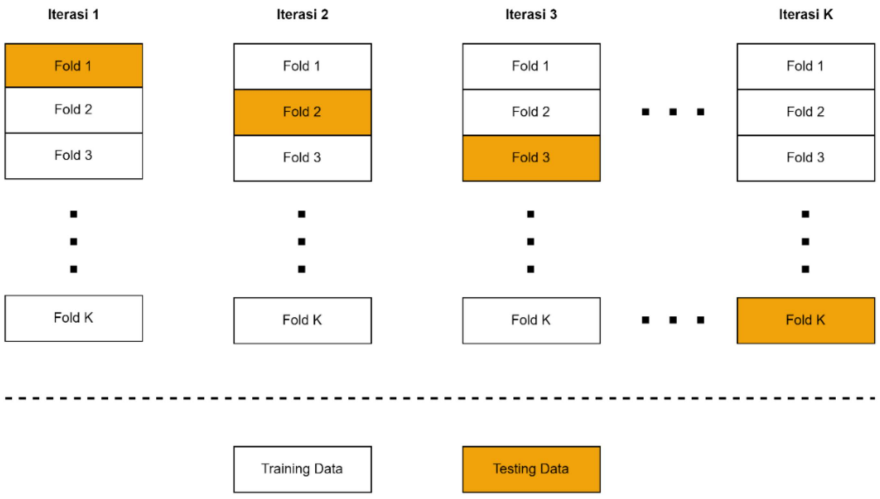
d. *F1 Score*

F1 Score merupakan metrik yang mengukur keseimbangan antara *precision* dan *recall* dalam kinerja model. Skor ini dihasilkan dengan menggabungkan nilai *precision* dan *recall* menggunakan rata-rata harmonik. Dengan mempertimbangkan kedua aspek ini, *F1 Score* menjadi alat ukur yang efektif dalam situasi di mana *precision* dan *recall* sama pentingnya, memberikan gambaran menyeluruh tentang performa model dalam klasifikasi data:

$$F1\ score = \frac{2(recall * precision)}{recall + precision} \quad (2.4)$$

2.2.10 *Cross-Validation*

Selain *Confusion Matrix* ada pula *Cross-validation*. *Cross-validation* merupakan teknik yang sering digunakan untuk melakukan evaluasi dalam model *Machine Learning* [15]. Teknik ini melakukan *resampling* data untuk mengevaluasi kemampuan generalisasi dari model prediktif dan mencegah terjadinya *overfitting* [38]. Dalam teknik *Cross-validation*, data dibagi menjadi beberapa bagian, yang dikenal sebagai K lipatan. Pada setiap iterasi, satu lipatan digunakan sebagai data uji dan sisanya sebagai data latih. Metode ini bertujuan untuk mendapatkan evaluasi yang lebih akurat terhadap model, karena semua data digunakan baik untuk pelatihan maupun pengujian. Ini membantu memastikan bahwa hasil evaluasi mencerminkan kinerja model secara keseluruhan.

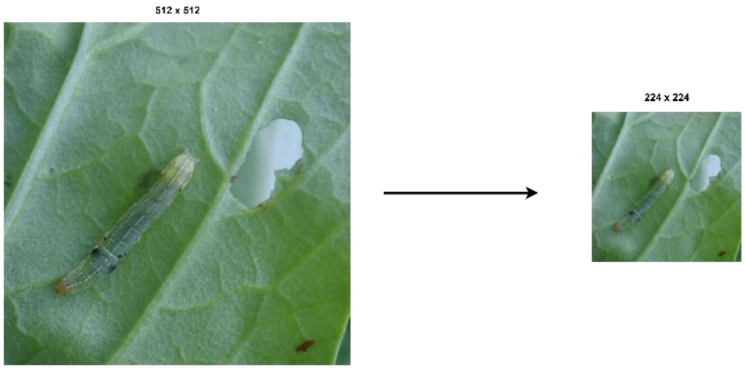


Gambar 2. 6 *K-Cross Validation* [15]

2.2.11 *Preprocessing Data*

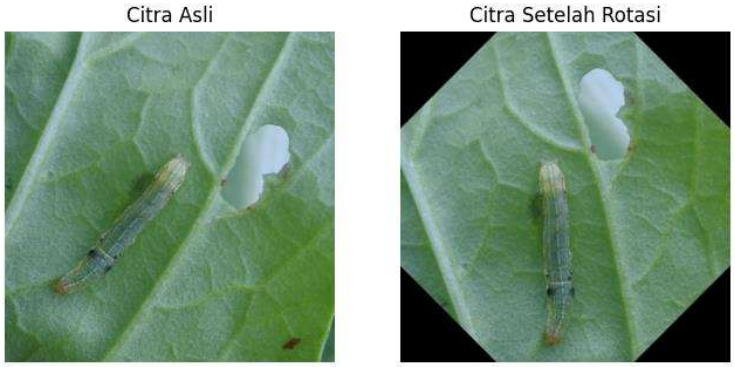
Data *preprocessing* adalah tahap dimana citra yang telah dikumpulkan kemudian akan diolah lebih lanjut sehingga dapat diterima dengan baik oleh komputer dan siap digunakan untuk melakukan pengembangan model *machine learning*. Ada beberapa cara dalam melakukan *preprocessing* data, yaitu Mengubah format, memisahkan antara atribut dan label, *resize*, *normalization*, dan *split* pada

dataset [15][39]. Proses-proses tersebut dapat mengurangi *loss* dan meningkatkan akurasi dari model pada saat pelatihan model dilakukan [40]. Salah satu contoh *preprocessing resize* pada citra dapat dilihat pada Gambar 2.8.



Gambar 2. 7 *Resize* Ukuran Citra

Augmentation merupakan salah satu proses *preprocessing* yang dapat menambah data dari data asli, sebab pada proses *preprocessing* sebelumnya data yang ada masih sedikit, sehingga dengan melakukan proses augmentasi ini dapat memperbanyak data yang lebih bervariasi [39]. Augmentasi yang umum dilakukan seperti *padding*, *flipping horizontal*, dan *cropping*, *rescale*, *flipping horizontal*, *zoom range*, *rotation range* [41][42].



Gambar 2. 8 *Rotation Range* Pada Citra

2.2.12 Python

Python merupakan bahasa pemrograman interpretatif yang memiliki kemampuan untuk digunakan di berbagai *platform* dan memiliki banyak fungsi. Bahasa ini dibuat oleh Guido van Rossum (GvR) dan dirilis pada tahun 1991. Salah satu keunggulan Python adalah kemampuannya dalam penanganan kesalahan

(*exception handling*) serta fokus pada sintaks yang mudah dibaca dan dimengerti (*readability*). Desain Python juga ditujukan untuk memudahkan dalam *prototyping*, sehingga Python menjadi bahasa yang sangat mudah dipahami dan fleksibel [43]. Python adalah bahasa pemrograman yang sangat populer dan digunakan secara luas di berbagai bidang seperti pemrograman aplikasi, analisis data, pembelajaran mesin, dan pengembangan web. Kelebihan utama Python adalah koleksi *library* yang luas dan dukungan komunitas yang besar. Python juga dikenal karena sintaksnya yang intuitif dan mudah dipelajari, menjadikannya pilihan yang tepat untuk pemula maupun profesional. Python menyediakan berbagai *library* yang sering digunakan oleh pengembang pembelajaran mesin dan ilmuwan data.

Berikut ini adalah beberapa library yang sering digunakan oleh pengembang machine learning dan data scientist dalam Python :

a. TensorFlow

TensorFlow, yang dikembangkan oleh Google *Brain*, merupakan *platform open-source* yang komprehensif untuk *machine learning*. *Platform* ini menawarkan ekosistem yang fleksibel dan lengkap, termasuk alat, perpustakaan, dan sumber daya komunitas, yang memudahkan para pengembang dalam membangun dan menerapkan aplikasi pembelajaran mesin. *TensorFlow* sangat populer di kalangan pengembang karena kemudahan penggunaannya dalam pengembangan model *machine learning*.

b. Keras

Keras adalah API untuk deep learning yang dibuat dalam Python. Ia menyediakan antarmuka penggunaan yang mudah dan dapat dijalankan di atas platform seperti *TensorFlow*, CNTK, dan *Theano*. Popularitas *Keras* di kalangan pengembang berkat kemudahan dan fleksibilitasnya dalam pengembangan model *deep learning*.

c. Numpy

NumPy (*Numerical* Python) adalah *library* Python untuk manipulasi *array* dan operasi aljabar linier, transformasi Fourier, serta manipulasi matriks. Dibuat oleh Travis Oliphant pada 2005, NumPy adalah proyek

open source yang mempercepat pengolahan *array*, meningkatkan efisiensi pengolahan data.

d. Sciket-Learn

Scikit-learn, *library* Python, digunakan untuk mengolah data kompleks. Bersifat *open source*, ia mendukung *machine learning* dengan beragam algoritma untuk klasifikasi, regresi linier, dan lainnya.

e. Pandas

Scikit-learn, sebagai sebuah *library* dalam bahasa pemrograman Python, digunakan untuk memproses data yang memiliki tingkat kompleksitas yang tinggi. *Library* ini tersedia secara *open source*, sehingga dapat digunakan secara bebas oleh pengguna.

f. Matplotlib

Matplotlib adalah sebuah *library* dalam bahasa pemrograman Python yang digunakan untuk melakukan visualisasi data. *Library* ini memiliki berbagai jenis *plot* yang dapat digunakan untuk membuat grafik dengan definisi tinggi, termasuk diagram lingkaran, histogram, *scatterplot*, grafik, dan lain-lain. Selain itu, *Matplotlib* juga berguna dalam merencanakan data numerik. Fungsi-fungsi yang dimiliki oleh *Matplotlib* membuatnya menjadi pilihan populer dalam melakukan analisis data. Sebagai library terbuka, *Matplotlib* dapat digunakan secara bebas oleh siapa pun.

2.2.13 Android

Android merupakan sistem operasi yang sangat populer yang dikembangkan oleh Google. Sistem operasi Android dapat digunakan pada berbagai perangkat, seperti *smartphone*, *tablet*, jam tangan, mobil, dan televisi. Banyak produsen perangkat telah memilih Android sebagai sistem operasi untuk produk-produk mereka. Selain itu, Android juga memiliki toko aplikasi yang memiliki lebih dari 2,5 miliar pengguna aktif setiap bulannya [44].

Android versi terbaru adalah Android versi 13 dengan nama Android 13 (Tiramisu) yang rilis pada 10 Februari 2022. Android sudah memiliki banyak versi dari sejarahnya yang pertama kali rilis pada 23 September 2008 dengan versi 1.0

dengan nama Android 1.0 (*Alpha*) tetapi versi ini belum di pakai secara resmi waktu itu.

2.2.14 Kotlin

Android mendukung beberapa bahasa pemrograman resmi, termasuk Java, C++, dan Kotlin yang merupakan bahasa terbaru. Pada tahun 2017, Kotlin secara resmi diakui oleh Google sebagai bahasa pemrograman yang disetujui untuk membangun aplikasi Android bersama dengan Java dan C++ dalam acara tahunan Google I/O 2017 [45].

Kotlin, sebagai bahasa pemrograman yang relatif baru dalam pengembangan Android, memiliki banyak keunggulan yang mempermudah proses pengembangan perangkat lunak. Kotlin adalah bahasa modern yang dengan cepat mendapatkan popularitas, bahasa ini mudah dipahami, sederhana, dan lebih manusiawi. Selain itu, Kotlin memiliki beberapa fitur unggulan, antara lain:

a. Ringkas (*Concise*)

Kotlin mengurangi jumlah kode *boilerplate* secara drastis dibandingkan dengan Java. Berikut ini adalah contoh perbedaan kode antara Java dan Kotlin dalam membuat aksi untuk sebuah tombol.

b. Aman (*Safe*)

Kotlin menghindari kesalahan kelas seperti *NullPointerException* (NPE), yang sering disebut sebagai "*The Billion Dollar Mistake*". Kotlin menangani pengecualian *Null* secara otomatis saat kompilasi berjalan.

c. Dapat dioperasikan secara bersilangan (*Interoperable*)

Kode Kotlin dapat berjalan secara bersamaan dengan kode Java. Artinya, Kotlin dapat memanggil kode yang ditulis dalam bahasa Java, dan sebaliknya.

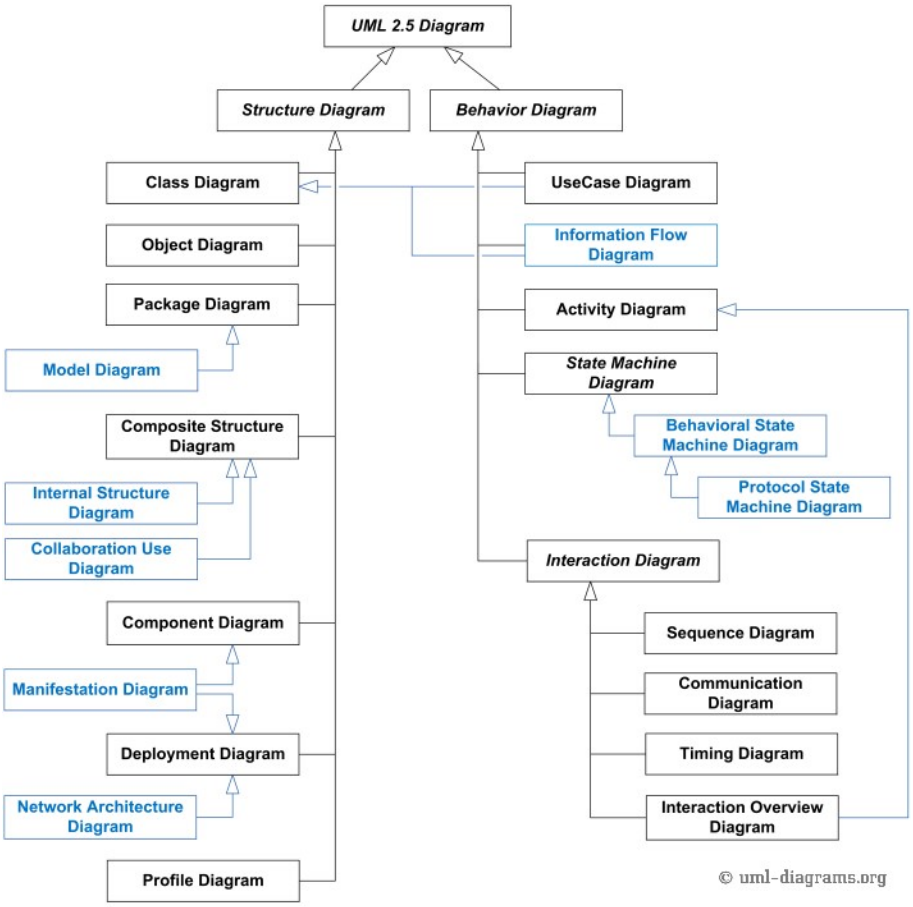
d. Dapat digunakan di berbagai alat (*Tool-friendly*)

Kotlin dapat ditulis di berbagai *editor* kode seperti IDE Java, dan juga dapat digunakan melalui baris perintah. Namun, sangat disarankan untuk menggunakan Intellij yang sudah terintegrasi dengan Kotlin, begitu juga dengan Android Studio.

2.2.15 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah sebuah bahasa yang digunakan untuk menggambarkan, menganalisis, dan mendeskripsikan model sistem

informasi atau aplikasi [46]. *Unified Modeling Language* (UML) menyediakan berbagai simbol dan notasi untuk merepresentasikan struktur dan perilaku sistem dalam pengembangan perangkat lunak. Dikembangkan oleh *Object Management Group* (OMG), UML telah menjadi standar industri yang penting. UML membantu dalam perancangan sistem, memahami spesifikasi sistem yang ada, dan berkomunikasi dengan tim pengembangan. Versi terbaru, UML 2.5 yang dirilis pada 2015, mencakup dua kategori diagram: struktur dan perilaku, seperti yang ditunjukkan pada Gambar 2.11.



Gambar 2.9 Klasifikasi diagram UML 2.5

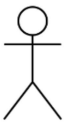
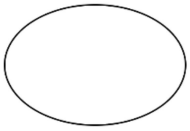
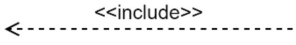
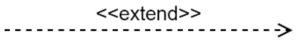
Dalam penelitian ini, digunakan diagram UML dari kategori perilaku, yaitu *use case diagram* dan *activity diagram*.

1. ***Use Case Diagram***

Use Case Diagram sendiri merupakan salah satu jenis diagram dalam UML yang digunakan untuk menggambarkan sebuah interaksi antara sistem

dan pengguna. *Use Case Diagram* terdiri dari beberapa elemen yang dapat dilihat pada Tabel 2.3.

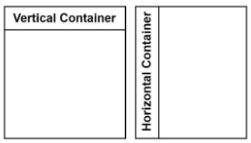
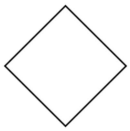
Tabel 2. 3 Elemen *Use Case Diagram*

Simbol	Nama	Keterangan
	*System*	Sistem yang sedang dikembangkan
	*Actor*	Entitas yang berinteraksi dengan sistem
	*Use case*	Aktivitas yang dapat dilakukan *actor* pada sistem
	Association	Hubungan antara *actor* dengan *use case*
	Generalization	Menggambarkan suatu *use case* merupakan turunan dari *use case* lain.
	*Include*	Suatu *use case* termasuk bagian dari *use case* lain.
	*Extend*	Satu *use case* dapat diperluas dengan *use case* lain.

2. Activity Diagram

Activity Diagram merupakan gambaran dari aktivitas atau aliran kerja dari sebuah sistem. Setiap dari *use case* yang telah dibuat akan dapat memiliki satu *activity diagram*. *Activity Diagram* terdiri dari beberapa elemen yang dapat dilihat pada Tabel 2.4.

Tabel 2. 4 Elemen *Activity Diagram*

Simbol	Nama	Keterangan
	*Swimline*	Wadah dari setiap aktivitas yang terjadi
	*Initial Node*	Menandakan awal dimulainya suatu *workflow*.
	*Activity*	Aktivitas yang dilakukan pada sistem
	*Decision*	Digunakan untuk menggambarkan suatu tindakan yang harus diambil pada kondisi tertentu.
	*Join*	Penggabungan dua atau lebih aktivitas menjadi satu
	Control Flow	Menghubungkan satu sumbol dengan simbol lainnya
	*Final Node*	Menandakan berakhirnya suatu *workflow*.

2.2.16 *Extreme Programming*

Kerangka kerja untuk pengembangan perangkat lunak tangkas yang memprioritaskan kesenangan klien disebut *Extreme Programming* (XP). Untuk menghasilkan perangkat lunak berkualitas tinggi secara efisien, seperangkat keyakinan, prinsip, dan praktik harus diimplementasikan dengan cara yang

terorganisir. Iterasi yang sering, desain yang lugas, umpan balik yang cepat dari pengujian yang berkelanjutan, dan interaksi dengan klien, semuanya memainkan peran penting dalam proses pengembangan. XP paling cocok untuk tim kecil hingga menengah dan sangat baik dalam menangani persyaratan yang berubah-ubah. Ini mencakup teknik yang meningkatkan kualitas perangkat lunak dan kecepatan pengembangan, seperti pemrograman berpasangan, kepemilikan kode kolaboratif, dan integrasi berkelanjutan [47].

Tahapan yang dilalui pada proses pengembangan perangkat lunak pada XP dijabarkan menjadi 4 tahapan, yaitu [48]:

1. *Planning*

   Ini adalah langkah awal dalam pembangunan sistem dan melibatkan beberapa tindakan perencanaan, seperti menemukan masalah, menganalisis kebutuhan, mengumpulkan data dan menetapkan jadwal pelaksanaan.

2. *Design*

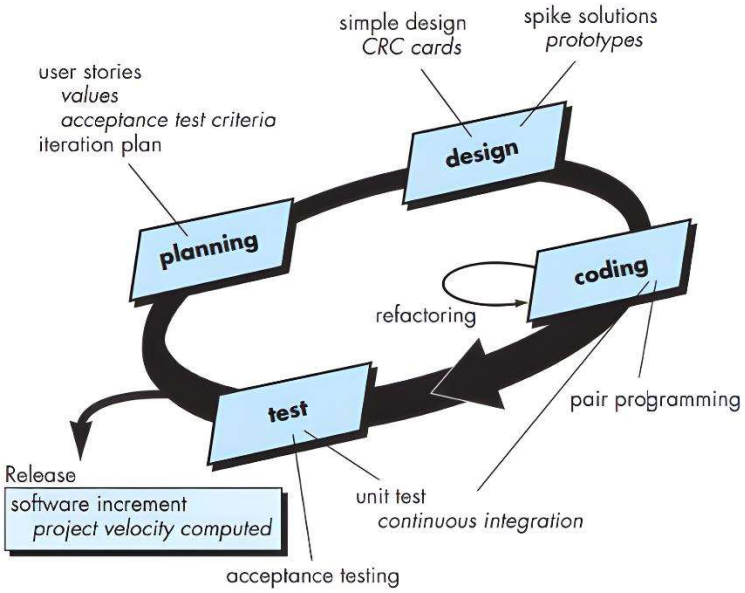
   Tahap selanjutnya adalah perancangan. Pada tahap ini, kegiatan pemodelan dilakukan, termasuk *design User Interface* (UI) aplikasi, pemodelan sistem. *Design* UI aplikasi menggunakan Figma, pemodelan sistem dan arsitektur menggunakan diagram Unified Modelling Language (UML).

3. *Coding*

   Penerapan pemodelan yang telah dibuat ke dalam UI dengan menggunakan bahasa pemrograman Kotlin dan pembuatan model CNN dengan menggunakan bahasa pemrograman Python.

4. *Testing*

   Setelah tahapan pengkodean selesai, tahapan pengujian sistem dilakukan. Tahapan ini menguji beberapa masukan untuk memastikan bahwa aplikasi berjalan sesuai dengan fungsinya dan untuk mengidentifikasi kesalahan apa saja yang muncul saat aplikasi berjalan. Pada tahap ini, pengujian *blackbox* digunakan untuk menguji aplikasi dengan beberapa input untuk memastikan bahwa aplikasi berjalan sesuai dengan fungsinya masing-masing.



Gambar 2. 10 Metode *Extreme Programming* [48]

XP memfokuskan pada pengembangan produk yang berkualitas tinggi, dengan mengutamakan komunikasi tim, *feedback* yang cepat, dan perbaikan terus-menerus.

2.2.17 *Black Box*

Proses pengujian sistem adalah kegiatan untuk memverifikasi dan memvalidasi sebuah sistem guna memastikan kepatuhan terhadap spesifikasi yang ditentukan serta kecocokan dengan kebutuhan pengguna, dan penggunaannya secara aman serta efisien. Dalam pengujian ini, terdapat beragam metode yang dapat diaplikasikan, termasuk *blackbox testing*. *Blackbox testing* adalah metode di mana penguji hanya fokus pada *input* yang diberikan dan *output* yang dihasilkan oleh sistem, tanpa mengetahui bagaimana proses *internal* sistem tersebut bekerja [49]. Dalam *blackbox testing*, penguji tidak mengakses implementasi *internal* sistem yang diuji, hanya berfokus pada *input* dan *output* yang diharapkan. Metode ini bertujuan menguji kemampuan sistem untuk memproses *input* sesuai dan menghasilkan *output* yang tepat berdasarkan spesifikasi yang telah ditentukan.