BAB II
TINJAUAN PUSTAKA

2.1 Kajian Pustaka

Banyak penelitian yang membahas mengenai *deep learning* untuk mengklasifikasi sebuah gambar yang diterapkan di berbagai bidang teknologi termasuk aplikasi *mobile*. *Deep learning* merupakan teknik *machine learning* yang mengajarkan komputer seperti halnya otak manusia yaitu belajar berdasarkan contoh. Model *deep Learning* melakukan klasifikasi data baik dari data gambar, teks, atau suara menggunakan jenis-jenis arsitektur *neural network* yang bermacam-macam dan bervariasi lapisan-lapisannya [11].

Penelitian dengan judul " Pet Paradise: Aplikasi Adopsi Hewan Peliharaan Berbasis Android Menggunakan *Convolutional Neural Network*" yang dilakukan oleh Dery Sudrajat pada tahun 2021 bertujuan untuk mengembangkan aplikasi Android yang membantu masyarakat dalam proses adopsi hewan peliharaan. Aplikasi ini menggunakan teknologi *machine learning* dari pustaka Firebase ML-Kit untuk mengenali jenis hewan berbasis Android. Namun, model *machine learning* yang digunakan masih memiliki akurasi yang rendah untuk mengenali kucing dan anjing, yaitu sebesar 41% dan 21%, hal ini disebabkan oleh kualitas *dataset* yang kurang memadai [10].

Penelitian berjudul "Aplikasi Pendeteksian Objek Buah-Buahan yang Memiliki Kemiripan Menggunakan Algoritma *Faster R-CNN* Berbasis Android" yang dilakukan oleh Ferdinan Mulyanto pada tahun 2020 bertujuan untuk mengembangkan aplikasi Android yang dapat mengidentifikasi buah-buahan yang memiliki kemiripan seperti apel, pir, anggur, blueberry, leci, dan rambutan dengan menggunakan algoritma Faster R-CNN. Penelitian ini menggunakan total 1018 gambar data untuk pelatihan dan 129 gambar data untuk validasi, yang berasal dari foto penulis, pixabay, dan kaggle. Hasil penelitian mencapai akurasi sebesar 95,98%[12]. Aplikasi ini hanya mampu berjalan pada *platform* Android saja.

Penelitian lain yang dilakukan oleh Guanhao Yang, Wei Feng, Jintao Jin, Qujiang Lei, Xiuhao Li, Guangchao Gui, dan Weijun Wang pada tahun 2020

dengan judul *"Face Mask recognition System with YOLOV5 Based on Image Recognition*" memiliki tujuan untuk mengembangkan model pendeteksi masker yang dapat diintegrasikan dengan kamera. Model tersebut dilatih untuk mengenali apakah seseorang menggunakan masker atau tidak dengan menggunakan *dataset* berjumlah 7959 data wajah yang sudah diannotasi. Hasil penelitian menunjukkan bahwa model YOLOv5 mencapai akurasi sebesar 97,9% dan mengungguli beberapa model *machine learning* lainnya dalam perbandingan [7]. Namun pada penelitian ini belum mampu mendeteksi masker yang tertutup tangan.

Penelitian Penelitian lain yang berjudul " *Fruit Classification Comparison Based on CNN and YOLO*" yang ditulis oleh Riyanshu Raj, S S Nagaraj, Saurav Ritesh, Thushar T A, dan V M Aparanji dan diterbitkan pada tahun 2021 memiliki tujuan untuk membandingkan kinerja algoritma CNN dan YOLO dalam mengklasifikasi buah. Penelitian ini bertujuan untuk membantu petani dalam mendeteksi kondisi buah dan mengoptimalkan pola panen serta mengurangi biaya pekerja. Hasil penelitian menunjukkan bahwa YOLO unggul dibandingkan dengan CNN dengan mencapai akurasi sebesar 85%, sedangkan CNN hanya mencapai akurasi sebesar 63% [13]. Namun, penelitian ini tidak menjelaskan tentang implementasi algoritma yang digunakan dalam dunia nyata.

Penelitian lain yang berjudul "Rancang Bangun Aplikasi Android "Kuliah Apa?" Berbasis Flutter dan TensorFlow Lite" oleh Arif Taufiq M Pratama dan Ahmad R. Pratama yang diterbitkan pada tahun 2021, yang bertujuan untuk memudahkan calon mahasiswa baru untuk memilih program studi dengan memberikan rekomendasi program studi berdasarkan nilai mata pelajaran, minat, dan data diri lainnya. Data yang digunakan untuk pengembangan model *machine learning* terdiri dari variabel kategorik dan numerik. Aplikasi ini dikembangkan menggunakan framework Flutter, TensorFlow Lite, dan Google Firebase[14]. Meskipun aplikasi dikembangkan menggunakan Flutter, aplikasi ini hanya tersedia untuk *platform* Android dan terdapat catatan bahwa akurasi model *machine learning* masih rendah.

Dari penjelasan di atas, ringkasan penelitian yang relevan ditunjukkan pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No.	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
1.	Pet Paradise: Aplikasi Adopsi Hewan Peliharaan Berbasis Android Menggunakan *Convolutional Neural Network* [10]	Melakukan penelitian untuk membuat aplikasi *mobile* menggunakan metode MADLC dan bertujuan untuk memudahkan orang yang ingin mengadopsi hewan terlantar khususnya kucing dan anjing.	Pada penelitian ini aplikasi yang dibuat hanya dikhususkan untuk aplikasi Android saja.	Akurasi model *machine learning* masih rendah untuk kucing dan anjing yaitu 41% dan 21% karena *dataset* yang kurang baik	Penulis pada penelitian ini melakukan penelitian untuk membuat aplikasi untuk membantu orang yang ingin mengadopsi hewan terlantar dengan fitur seperti mengenali jenis anjing dan kucing, pencarian hewan, dan kemudahan dalam proses adopsi.	Metode MADLC pada penelitian ini sukses dalam membuat aplikasi yang terintegrasi dengan model *machine learning* dan bertujuan untuk membantu orang yang ingin mengadopsi hewan terlantar.
2.	Aplikasi Pendeteksian Objek Buah-Buahan yang Memiliki Kemiripan Menggunakan Algoritma Faster R-Cnn Berbasis Android [12]	Melakukan penelitian untuk mendeteksi buah yang memiliki kemiripan menggunakan algoritma Faster R-CNN	Membahas mengenai bagaimana membuat aplikasi Android yang mampu mendeteksi buah-buahan menggunakan algoritma Faster R-CNN	Aplikasi yang dibuat hanya mampu berjalan pada platform Android	Peneliti menggunakan 1018 gambar sebagai data training dan 129 gambar sebagai data validasi yang diambil dari hasil foto penulis, pixabay, dan Kaggle.	Penelitian ini menghasilkan aplikasi Android yang mampu mendeteksi buah – buahan yang memiliki kesamaan seperti apel, pir, anggur, blueberry, leci, dan rambutan dengan menggunakan algoritma Faster R-CNN dan mencapai akurasi 95,98%.
3.	*Face Mask recognition System with YOLOV5 Based on Image Recognition* [7]	Melakukan penelitian untuk menghasilkan model pendeteksi masker wajah dengan menggunakan YOLOv5	Membahas mengenai bagaimana mendeteksi masker wajah menggunakan model *You Only Look Once* (YOLOv5)	Hanya membahas cara membuat model menjadi optimal tanpa menjelaskan *development* aplikasinya	Peneliti mengambil *dataset* dari (https://github.com/AIZOOTech/FaceMaskDetection) AIZOOTe team's FaceMaskDetection, model dilatih menggunakan metode YOLOv5 kemudian	Penelitian ini menunjukkan bahwa model YOLOv5 dapat melakukan klasifikasi gambar dengan sangat baik dan mengungguli beberapa model

No.	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
					model dioptimasi	lain yang digunakan sebagai perbandingan
4.	*Fruit Classification Comparison Based on CNN and YOLO* [13]	Melakukan penelitian klasifikasi buah menggunakan algoritma CNN dan YOLO	Melakukan perbandingan algoritma CNN dengan untuk mengklasifikasi buah	Hanya membandingkan algoritma tanpa mengimplementasikan aplikasinya.	Peneliti melakukan penelitian ini untuk menguji algoritma yang nantinya mampu membantu petani untuk mendeteksi kondisi buah untuk mengoptimalkan pola panen dan mengurangi biaya pekerja.	Penelitian ini menunjukkan bahwa algoritma YOLO mampu mengungguli CNN dalam mendeteksi buah dengan akurasi 85% dibandingkan dengan CNN yang hanya mendapatkan 63% akurasi.
5.	Rancang Bangun Aplikasi Android "Kuliah Apa?" Berbasis Flutter Dan Tensorflow Lite [14]	Melakukan penelitian untuk membuat aplikasi mobile menggunakan *framework* Flutter, TensorFlow Lite, dan Google Firebase untuk membantu calon mahasiswa memilih program studi	Pada penelitian ini aplikasi hanya dikembangkan untuk platform Android saja	Pengembangan aplikasi hanya untuk *platform* Android meskipun menggunakan *framework* Flutter yang mendukung *multi-platform* dan akurasi model masih rendah	Peneliti mengambil data dari Badan Sistem Informasi (BSI) di UII berupa data kategorik dan numerik untuk melatih model *machine learning*.	Penelitian ini menggunakan *framework* Flutter, TensorFlow Lite, dan Google Firebase untuk membangun aplikasi yang bertujuan untuk membantu calon mahasiswa baru untuk memilih proram studi. Aplikasi telah dinyatakan lolos uji fungsionalitas dan usabilitas

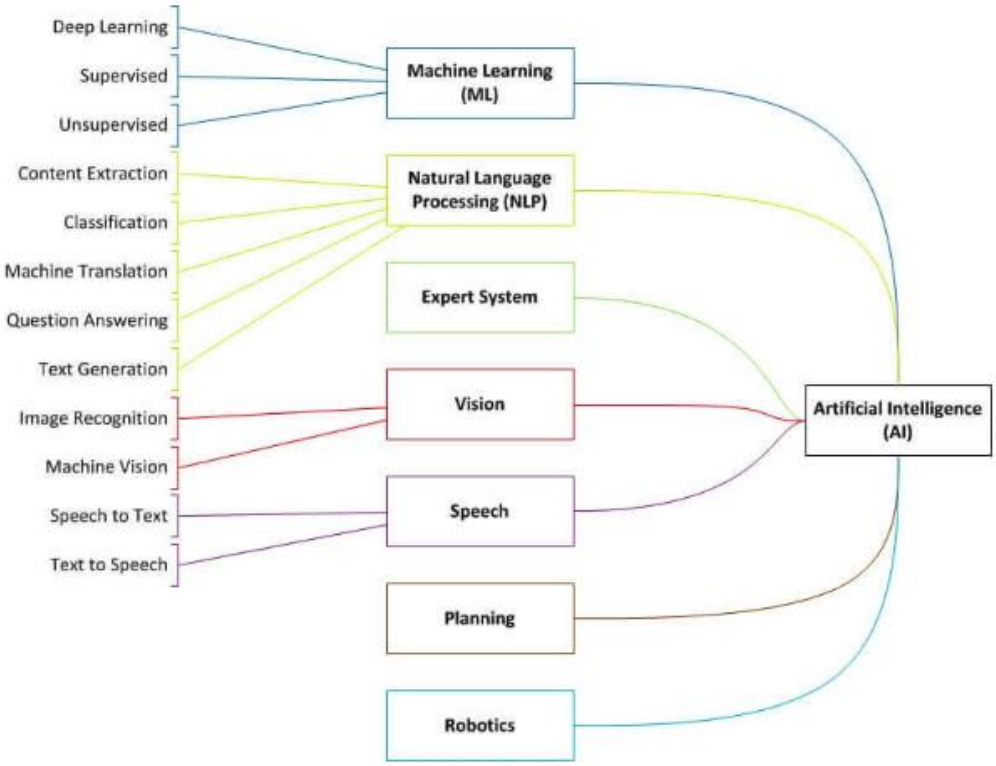
2.2 Dasar Teori

Berikut adalah kajian mengenai beberapa teori yang digunakan pada penelitian ini:

2.2.1 Artificial Intelligence

Artificial intelligence (AI) atau kecerdasan buatan adalah bidang keilmuan yang memungkinkan komputer untuk memiliki kemampuan kecerdasan seperti

manusia. AI memiliki definisi yang berbeda-beda dari beberapa ahli. Menurut Francis X. Govers, [15] secara umum, *artificial intelligence* adalah mesin yang menunjukkan beberapa karakteristik kecerdasan meliputi berpikir (*thinking*), penalaran (*reasoning*), perencanaan (*planning*), belajar (*learning*), dan beradaptasi (*adapting*). Sedangkan Stuart Russel dan Peter Norvig membagi kecerdasan buatan menjadi beberapa bagian yaitu *thinking rationally*, *thinking humanly*, dan *action rationally*. Menurut mereka, *action rationally* adalah aspek kecerdasan buatan yang paling sesuai, karena pendekatan ini didasarkan pada keyakinan bahwa komputer dapat melakukan penalaran logis dan bertindak secara rasional berdasarkan hasil penalaran tersebut. Ruang lingkup *artificial intelligence* ditunjukkan pada Gambar 2.1.



Gambar 2.1 Ruang lingkup *artificial intelligence* [16]

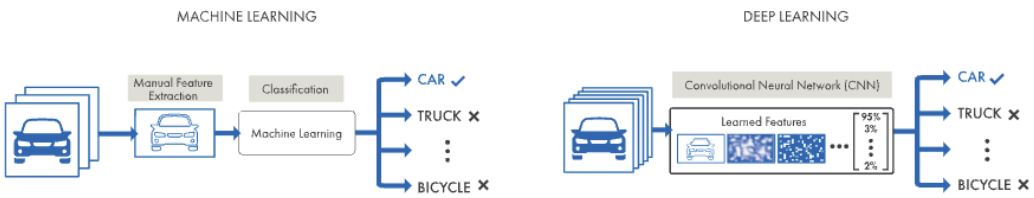
2.2.2 Machine Learning

Berikutnya adalah *machine learning* (ML), *machine learning* merupakan pengembangan dari *artificial intelligence*. Rifkie [16] menggambarkan bahwa *machine learning* merupakan bidang studi ilmu komputer yang memungkinkan

komputer untuk belajar tanpa perlu program yang spesifik. Algoritma *machine learning* mencari pola-pola tertentu dalam kumpulan data yang akan menentukan karakteristik khusus dari data tersebut, dan dari pola-pola tersebut, algoritma dapat menyusun aturan atau *rule*. Kemudian, aturan ini digunakan untuk mengidentifikasi dan memprediksi data baru yang relevan dengan model yang telah dibuat berdasarkan aturan tersebut.

2.2.3 Deep Learning

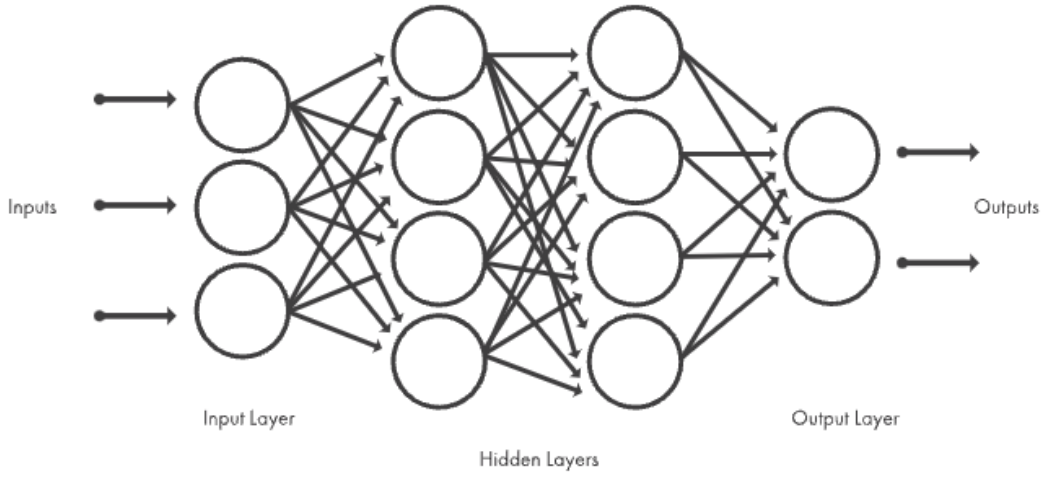
Setelah memahami konsep dari *artificial intelligence* dan *machine learning*, selanjutnya adalah *deep learning*. *Deep learning* adalah salah satu cabang dari *machine* learning yang terinspirasi dari struktur otak manusia. Algoritma dalam *deep learning* menggunakan jaringan syaraf tiruan yang mampu belajar dari sejumlah besar data. Berdasarkan Gambar 2.2, dapat diketahui perbedaan antara *machine learning* dan *deep learning*. *Machine learning* mengharuskan untuk mengekstraksi fitur dan pengklasifikasi secara manual, maka dengan *deep learning*, ekstraksi fitur dan langkah-langkah pemodelan dilakukan secara otomatis.



Gambar 2.2 Perbedaan *machine learning* dan *deep learning* [11]

Kebanyakan metode *deep learning* menggunakan arsitektur *neural network*. Kata "*deep*" mengacu kepada jumlah *hidden layers* di dalam *neural network* yang berjumlah sangat banyak. *Deep learning* dilatih menggunakan *dataset* berlabel yang berjumlah banyak dengan arsitektur *neural network* yang mempelajari fitur-fitur secara langsung dari data [11]. Cara kerja dari *neural network* adalah dengan *input layer* yang menyediakan vektor *input* ke dalam jaringan. Kemudian, *input* tersebut diolah melalui *layer* kedua, yang juga dikenal sebagai *hidden layer*. *Output* dari *hidden layer* ini kemudian digunakan sebagai *input* untuk *layer*

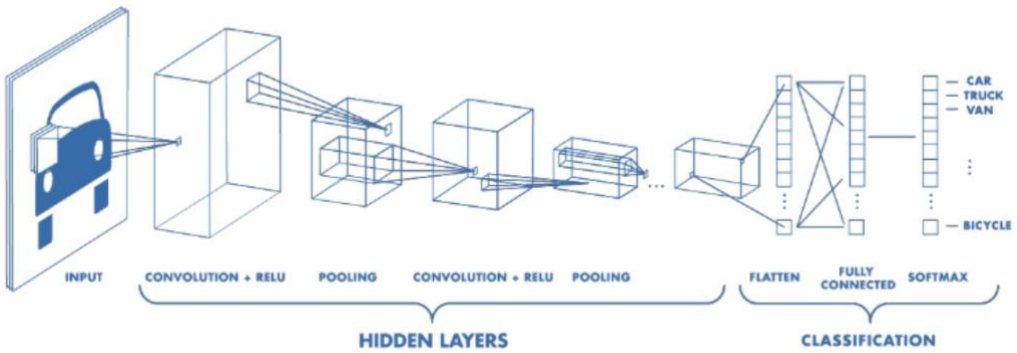
berikutnya, dan proses ini berlanjut hingga mencapai *output layer*. Struktur *neural network* ditunjukkan pada Gambar 2.3.



Gambar 2.3 Struktur *neural network* yang terdiri dari satu *set node* yang saling berhubungan [11]

2.2.4 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan salah satu bagian dari *Deep Neural Network* yang digunakan untuk melakukan pengenalan gambar. CNN umumnya terdiri dari tiga jenis lapisan (*layer*), yaitu *Convolutional Layer*, *Pooling Layer*, dan *Fully Connected Layer*.



Gambar 2.4 Arsitektur CNN

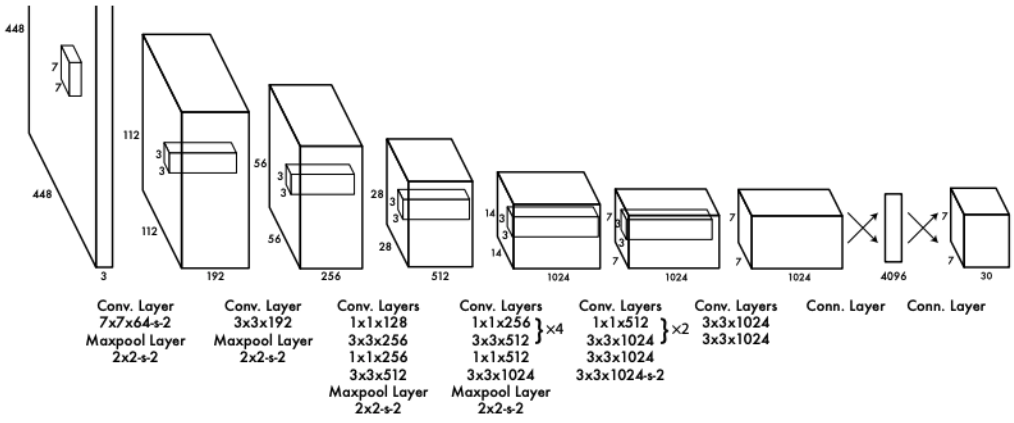
Pada Gambar 2.4, menunjukkan proses layer CNN yang dibagi menjadi 2 bagian yaitu *hidden layers* dan *classification*. *Hidden layers* terdiri dari *convolutional layer*, *activation* dalam hal ini *relu*, dan *pooling layer*. Sedangkan

classification terdiri dari *flatten*, *fully connected*, dan fungsi aktivasi dalam hal ini *softmax*. *Hidden layer* berfungsi untuk mengubah *input* menjadi fitur-fitur berdasarkan ciri atau keunikan pada *input* dan *output* dari *hidden layer* berupa peta fitur tiga dimensi (biasanya berbentuk *tensor*) yang mewakili fitur-fitur yang telah diekstraksi dari gambar. Kemudian dilanjutkan ke dalam bagian *classification*, *Layer flatten* akan mengubah peta fitur tiga dimensi menjadi vektor satu dimensi dengan cara menggabungkan semua elemen peta fitur menjadi satu baris vektor. Setelah data di *flatten* menjadi vektor, selanjutnya dapat diumpankan ke lapisan *fully connected* (*dense layers*) untuk melakukan klasifikasi akhir dengan menggunakan fungsi *softmax* pada *output layer*.

2.2.5 YOLO

You Only Look Once (*YOLO*) adalah rangkaian model pembelajaran *end-to-end* yang dirancang untuk mendeteksi objek dengan akurasi tinggi dan waktu inferensi yang cepat. Sebelum membahas YOLOv5 yang akan digunakan dalam penelitian ini, penting untuk mengetahui tentang variasi lain dari seri YOLO.

a. YOLOv1



Gambar 2.5 Arsitektur YOLOv1 [17]

YOLOv1 pertama kali diperkenalkan oleh Joseph Redmon pada bulan Juni 2015. YOLOv1 dibangun di atas metode R-CNN. *Input* gambar hanya diproses sekali sesuai dengan namanya (*You Only Look Once*), fitur yang berbeda akan diekstraksi melalui beberapa lapisan *konvolusi*, dan *kernel konvolusi parameter*. YOLOv1 dapat mendeteksi gambar sangat cepat mencapai 45 *frame* per detik.

Akan tetapi YOLOv1 memiliki kekurangan yaitu akurasi deteksi posisi rendah dan tidak bisa mendeteksi objek yang kecil [18]. Arsitektur YOLOv1 ditunjukkan pada gambar 2.5.

b. YOLOv2

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Gambar 2.6 Arsitektur Darknet 19 [19]

Selanjutnya adalah YOLOv2, YOLOv2 merupakan pengembangan dari *YOLOv1*. Seperti yang ditunjukkan pada Gambar 2.6, *YOLOv2* menggunakan arsitektur *Darknet 19* sebagai *backbone* nya dengan 19 *convolutional layers* dan 5 *max pooling layers* dan sebuah *layer softmax* untuk mengklasifikasikan objek. *YOLOv2* memiliki kelemahan yaitu akurasi deteksi objek kecil tidak terlalu tinggi.

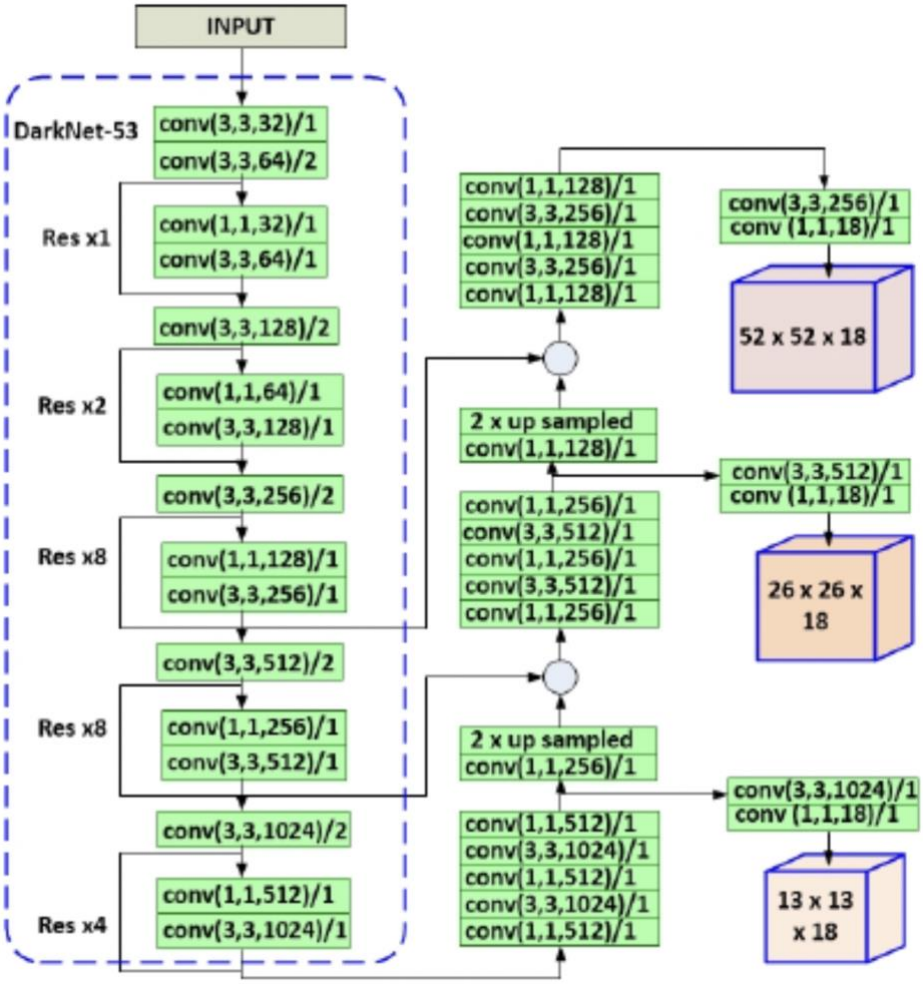
c. YOLOv3

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Gambar 2.7 Arsitektur Darknet-53[20]

YOLOv3 membuat beberapa peningkatan dari seri sebelumnya. Lapisan konvolusi pada YOLOv3 sekitar 2,8 kali lipat lebih banyak dari YOLOv2, dan meningkatkan kedalaman dan ketebalan jaringan sehingga meningkatkan akurasi model. Selain itu *layer softmax classifier* diganti dengan *multiple logistic classifier* [20]. Arsitektur YOLOv3 ditunjukkan pada Gambar 2.7.

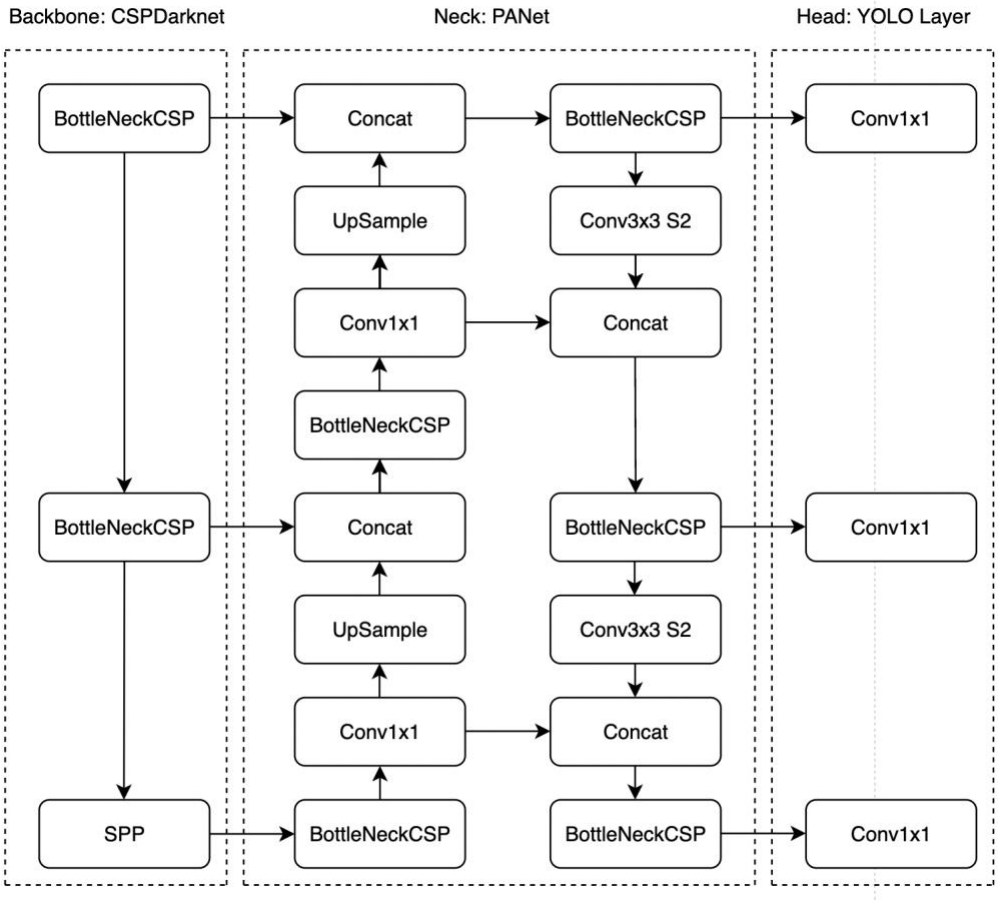
d. YOLOv4



Gambar 2.8 Arsitektur YOLOv4

YOLOv4 muncul pada tahun 2019. Memiliki tujuan utama untuk merancang sistem deteksi target yang cepat yang dapat diterapkan di lingkungan nyata dan dapat dioptimalkan secara paralel. YOLOv4 menggunakan beberapa teknologi terbaru dalam jaringan *deep learning* seperti *CutMix data enhancement*, *Swish*, dan *Mish activation functions* [21]. Arsitektur YOLOv4 ditunjukkan pada Gambar 2.8.

e. YOLOv5



Gambar 2.9 Arsitektur YOLOv5 [22]

YOLOv5 merupakan pengembangan dari YOLOv4. YOLOv5 memiliki kecepatan yang lebih cepat mencapai 140 *frame* per detik, ukuran yang lebih kecil, dan *weight* yang hampir 90% lebih kecil dari YOLOv4. Selain itu, YOLOv5 memiliki akurasi yang lebih tinggi dan kemampuan yang lebih baik dalam mengenali benda-benda kecil [18]. Arsitektur YOLOv5 ditunjukkan pada Gambar 2.9.

2.2.6 Transfer Learning

Transfer learning adalah melatih kembali model *machine learning* pada data baru tanpa harus melatih ulang seluruh jaringan menggunakan bobot atau *weight* yang sudah dilatih sebelumnya. YOLOv5 yang akan digunakan pada penelitian ini, telah dilatih pada kumpulan data *Common Objects in Context*

(COCO), kumpulan data untuk pengenalan, segmentasi, dan pelabelan objek. *Dataset* ini berisi lebih dari 200.000 gambar berlabel dengan 80 kelas yang berbeda[10].

Cara melakukan transfer learning pada YOLOv5 pada penelitian ini adalah dengan membekukan *backbone,* merubah jumlah *default class* menjadi 20 seperti jumlah class pada penelitian ini. Kemudian melakukan *tuning hyperparameter* seperti *batch size,* dan *epochs*. Dengan *transfer learning* ini membutuhkan *resources* yang lebih sedikit daripada *training* normal dan membuat waktu *training* menjadi lebih cepat.

2.2.7 Dataset YOLOv5

Dataset merupakan kumpulan data yang dikustomisasi atau dibuat khusus untuk digunakan dalam pelatihan dan pengujian model deteksi objek YOLOv5. *Dataset* YOLOv5 perlu disiapkan secara spesifik untuk kasus penggunaan yang diinginkan. Pemilihan *dataset* yang sesuai dengan kebutuhan sangat penting untuk dapat meningkatkan performa dan akurasi deteksi objek YOLOv5. Langkah untuk membuat *dataset* YOLOv5 adalah:

- Pengumpulan Data: Data dapat berupa gambar atau video yang berisi objek-objek yang ingin dideteksi. Sebaiknya *dataset* mencakup variasi objek, pose, ukuran, dan kondisi pencahayaan yang berbeda.
- Anotasi Data: Setiap gambar dalam *dataset* perlu di-anotasi, yaitu menandai lokasi dan kelas objek yang ada dalam gambar. Anotasi dilakukan dengan membuat bounding box (kotak pembatas) di sekitar objek dan memberi label kelas objek tersebut.
- Pembagian *Dataset*: *Dataset* perlu dibagi menjadi *dataset* pelatihan (*training set*), *dataset* validasi (*validation set*), dan *dataset* pengujian (*testing set*) untuk keperluan pelatihan dan evaluasi model.
- Konversi ke Format YOLOv5: *Dataset* dan *anotasi* perlu dikonversi menjadi format yang dapat dikenali oleh YOLOv5. Dalam YOLOv5, anotasi disimpan dalam *file* dengan ekstensi .txt yang berisi informasi tentang lokasi dan kelas objek dalam setiap gambar.

2.2.8 Matriks Evaluasi

Pada penelitian ini, matriks evaluasi untuk model *deep learning* YOLOv5 dengan menggunakan *mean average precision* dan *confusion matrix* dan untuk evaluasi aplikasi *mobile* menggunakan akurasi aplikasi dalam mendeteksi objek pada gambar.

Pada model *machine learning* YOLO, model di evaluasi menggunakan *mean average precision* (mAP) yang merupakan matriks yang sering digunakan untuk mengevaluasi model objek detection seperti R-CNN dan YOLO. *Mean average precision* adalah metode perbandingan yang digunakan untuk membandingkan kotak pembatas kebenaran (*ground-truth bounding box*) dengan kotak yang terdeteksi oleh model, dan menghasilkan skor yang menunjukkan tingkat keakuratan model dalam memprediksi objek. Semakin tinggi skor, menandakan semakin akurat model dalam memprediksi [23]. Sementara *confusion matrix* merupakan salah satu metode yang digunakan untuk mengukur performa dalam masalah klasifikasi *machine learning* baik itu masalah dua kelas (*binary classification*) atau lebih (*multiclass classification*). Pada aplikasi *mobile*, akurasi aplikasi dalam mendeteksi dan mengidentifikasi objek dalam gambar dari berbagai kondisi akan diukur menggunakan *confusion matrix*.

1. *Mean Average Precision*

- ***Average Precision***

$$AP = \sum_{k=o}^{k=n-1} [recalls(k) - recalls(k+1) * precisions(k) \quad (2.1)$$

Average precision merupakan metode untuk menggambarkan secara ringkas kurva *precision* dan *recall* dengan menggunakan nilai tunggal yang mewakili rata-rata semua *precision*. *Average precision* dihitung menggunakan perulangan yang melewati semua *precision* atau *recall*, perbedaan antara nilai *recall* saat ini dan berikutnya dihitung dan kemudian dikalikan dengan nilai *precision* saat ini [23]. Rumus *average precision* seperti yang ditunjukkan pada persamaan 2.1.

- ***Mean Average Precision***

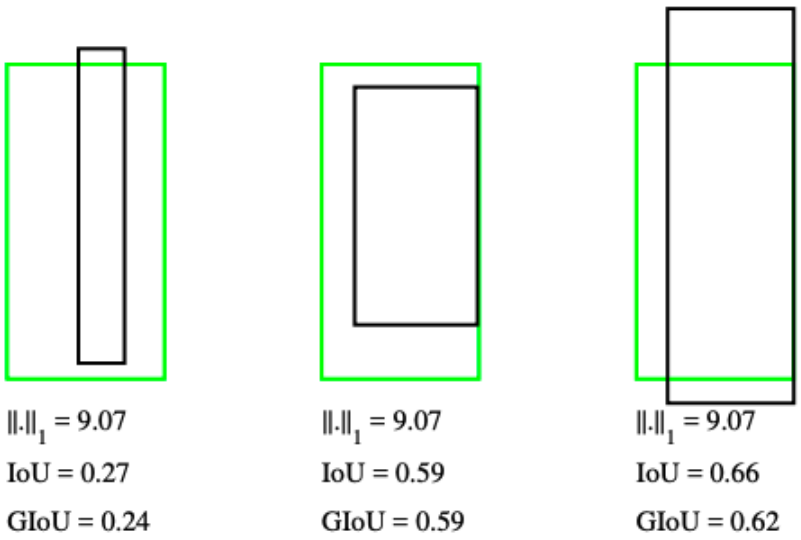
$$mAP = \frac{1}{N}\sum_{N}^{i=1} APi \qquad (2.2)$$

Berdasarkan persamaan 2.2, dapat diketahui bahwa *mean average precision (mAP)* adalah nilai rata-rata dari *Average precision (AP).* Semakin tinggi nilai mAP maka semakin akurat model dalam memprediksi.

- ***Intersection Over Union***

$$IoU = \frac{Luas\ perpotongan\ dua\ kotak}{\text{Luas gabungan dua kotak}} \qquad (2.3)$$

Dalam melatih model *objek detection* biasanya terdapat dua *input*, yaitu: gambar dan kotak pembatas kebenaran (*ground-truth boxes*) untuk setiap objek di dalam gambar. Untuk menilai kesesuaian antara kotak prediksi dengan *ground-truth boxes* adalah dengan menggunakan ukuran kuantitatif. Ukuran ini disebut dengan *intersection over union* (*IoU*). *IoU* adalah matriks evaluasi untuk *object detection* yang dapat membantu untuk mengetahui apakah suatu wilayah memiliki objek atau tidak. Rumus *intersection over union* seperti yang ditunjukkan pada persamaan 2.3.

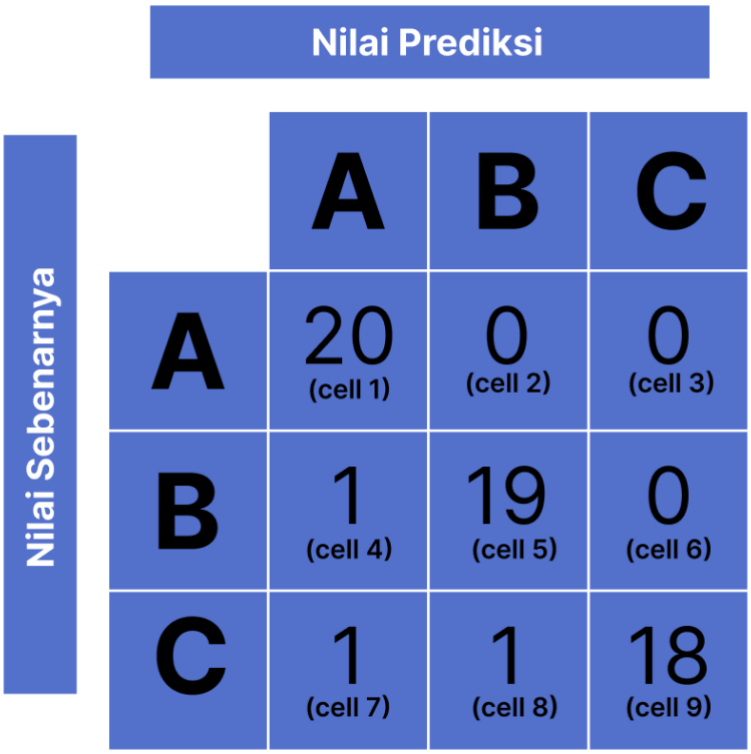


Gambar 2.10 Contoh *Intersection over Union* [24]

Seperti yang ditunjukkan pada Gambar 2.10, IoU menghitung berdasarkan persamaan berikutnya dengan membagi luas perpotongan antara dua kotak dengan luas persatuannya. Semakin tinggi IoU semakin baik prediksinya. Skor IoU mengukur seberapa dekat kotak prediksi dengan kotak pembatas kebenaran (*ground-truth boxes*) dan bernilai antara 0,0 sampai 1,0 dimana 1,0 adalah hasil yang optimal. Ketika IoU lebih besar dari *threshold*, maka kotak atau *box* mengelilingi objek dan akan digolongkan sebagai positif dan sebaliknya akan dianggap negatif [23]. Pengukuran IoU dilakukan pada saat pelatihan model.

2. *Confusion Matrix* untuk Evaluasi *Model Machine Learning*

Confusion matrix adalah tabel yang menunjukkan jumlah prediksi yang dilakukan untuk mengklasifikasikan kelas yang benar atau salah pada setiap data dari masing-masing kelas.. Pada penelitian ini akan menggunakan *confusion matrix multiclass classification* seperti yang terlihat pada Gambar 2.11.



Gambar 2.11 Tabel *confusion matrix multiclass classification*

Pada penelitian ini, *confusion matrix* digunakan untuk mengukur kinerja *model machine learning* dengan memanfaatkan beberapa ketentuan seperti nilai *True Positive* (TP), *False Positive* (FP), *False Negative* (FN) yang dapat digunakan untuk mengukur *accuracy*, *precision*, dan *recall*.

- *True Positive* (TP) = Model memprediksi objek dengan benar dari sebuah objek yang ada di gambar.
- *False Positive* (FP) = model salah memprediksi objek yang sebenarnya tidak ada di dalam gambar.
- *False Negative* (FN) = Objek sebenarnya ada di dalam gambar tetapi tidak terdeteksi oleh model [25].

Komponen diatas dapat digunakan untuk menjadi acuan dalam menentukan *recall, precision,* dan *F1-Score*. Dan hasilnya dapat menjadi acuan apakah model dapat dikatakan baik atau tidak.

a. Precision

$$recall = \frac{TP}{\mathrm{TP} + \mathrm{FP}} \qquad (2.4)$$

Berdasarkan persamaan 2.4, diketahui bahwa *precision* adalah perbandingan antara jumlah data yang berhasil diprediksi dengan benar dibandingkan dengan banyaknya data yang diprediksi positif. *Precision* mengukur akurasi model dalam mengklasifikasikan sampel sebagai positif [23].

b. Recall

$$precision = \frac{TP}{\mathrm{TP} + \mathrm{FN}} \qquad (2.5)$$

Berdasarkan persamaan 2.5, diketahui bahwa *recall* adalah perbandingan antara jumlah data yang berhasil diprediksi dengan benar dibandingkan dengan banyaknya data yang sebenarnya positif. *Recall* mengukur seberapa bagus

kemampuan model untuk mendeteksi sampel positif. Semakin tinggi nilai *recall*, semakin banyak sampel yang terdeteksi positif [23].

c. F1-Score

$$F1 - Score = \frac{2x(Precision\ x\ Recall)}{Precision + Recall} \qquad (2.6)$$

F1-Score adalah metode untuk menggambarkan perbandingan rata-rata *precision* dan *recall* yang dihitung dangan bobot yang sama. *F1-Score* digunakan untuk melihat hubungan dan kombinasi performa *precision* dan *recall* dan untuk meringkas performa model ke dalam satu matriks. Nilai *F1-Score* selalu berada diantara nilai *precision* dan *recall* [26]. Perhitungan *F1-Score* ditunjukkan pada persamaan 2.6.

3. Akurasi Aplikasi dalam Mendeteksi Objek

Akurasi dalam hal ini merupakan kemampuan aplikasi dalam mendeteksi dan memprediksi kelas dengan benar. Berdasarkan persamaan 2.7 diketahui bahwa akurasi dapat dihitung dengan cara membagi jumlah *class* yang diprediksi dengan benar dengan seluruh jumlah data.

$$\text{Akurasi} = \frac{\text{Jumlah data diprediksi dengan benar}}{\text{Jumlah data}} \qquad (2.7)$$

2.2.9 Android

Android adalah sistem operasi berbasis *linux* yang dikembangkan oleh Google dan telah menjadi salah satu *platform* yang paling populer. Sistem operasi *Android* memungkinkan pengguna untuk mengakses berbagai aplikasi, konten, dan layanan melalui Google Play Store atau toko aplikasi pihak ketiga. *Android* juga bersifat *open source*, artinya kode sumbernya dapat diakses oleh pengembang yang ingin memodifikasinya atau mengembangkan versi khusus dari

sistem operasi tersebut. Hal ini telah menyebabkan adopsi yang luas dan kemajuan pesat dalam pengembangan aplikasi dan ekosistem Android secara keseluruhan. Android pertama kali secara resmi dirilis pada bulan September 2008 dengan versi 1.0. Android dapat berjalan di berbagai perangkat seperti *smartphone*, tablet, televisi, jam tangan, dan bahkan mobil.

Android memiliki *Integrated Development Environment (*IDE*)* yangdikenal dengan nama Android Studio. Android Studio merupakan editor kode yang *powerful* dan dilengkapi dengan berbagai fitur yang dapat meningkatkan produktivitas seperti *visual layout editor, APK Analyzer, flexible build system*, dan *Emulator*.

2.2.10 IOS

IOS merupakan salah satu sistem operasi paling popular di dunia. IOS sendiri merupakan akronim dari *iPhone Operating System*, dan merupakan sistem operasi berbasi *Unix* yang mendukung semua perangkat seluler Apple. Versi pertama dari iOS dirilis pada tahun 2007, ketika iPhone diluncurkan di pasar untuk pertama kalinya. IOS menawarkan antarmuka pengguna yang intuitif dengan desain yang halus dan responsif. Ini mendukung berbagai aplikasi yang dapat diunduh melalui App Store, toko aplikasi resmi Apple. Seiring dengan fitur-fitur inti yang kuat, IOS juga menonjol dengan keamanan yang ketat dan kebijakan privasi yang kuat. Apple berusaha untuk melindungi data pengguna dan memastikan aplikasi yang tersedia di App Store telah melalui proses pengawasan dan verifikasi yang ketat.

Menurut Nishkarsh Verma dan Saurabh Sambhav [27], IOS merupakan sistem operasi yang paling aman, dikarenakan IOS adalah *proprietary software*, sehingga hanya tim pengembang Apple yang memiliki dan mampu memodifikasi *source code* IOS yang membuat performa IOS menjadi sangat baik dan sangat aman dibandingkan dengan sistem operasi lain seperti Android, Tizen, Ubuntu touch dan sebagainya.

2.2.11 Dart

Dart adalah bahasa pemrograman yang dirancang dan dikembangkan oleh Google, dan pengumumannya secara resmi dilakukan pada 10 Oktober 2011. Dart merupakan bahasa pemrograman yang berorientasi objek, yang artinya pemrograman dilakukan melalui penggunaan objek dan konsep-konsep terkait. Bahasa ini memiliki kemampuan untuk dikompilasi menjadi kode ARM (dikompilasi menjadi kode mesin yang dapat dijalankan pada arsitektur ARM) serta kode JavaScript. Keunggulan ini membuat kode Dart dapat dijalankan langsung pada platform yang mendukung ARM dan juga diubah menjadi kode JavaScript, yang dikenal sebagai bahasa pemrograman yang umum digunakan di web.

Bahasa dart dioptimalkan untuk memungkinkan pengembangan aplikasi dengan cepat, terutama dalam pengembangan lintas platform. Dart berupaya untuk memberikan pengalaman yang konsisten dan efisien bagi pengembang ketika mengembangkan aplikasi untuk berbagai platform seperti mobile (Android dan IOS), web, serta desktop (Windows, Linux, dan MacOS). Fokus Dart pada pengembangan dan kualitas produksi tinggi memungkinkan pengembang untuk menciptakan aplikasi berkualitas dengan cepat tanpa harus mengorbankan kinerja dan fitur pada berbagai platform yang berbeda.

2.2.12 Flutter

Flutter pertama kali diluncurkan pada tahun 2015 pada Dart Developer Summit dengan nama *Sky*. Flutter adalah *framework open source* yang dikembangkan oleh Google dan menggunakan bahasa pemrograman dart untuk membangun aplikasi modern, *native*, dan reaktif untuk IOS dan Android. Google juga bekerja pada Flutter desktop dan Flutter web dan juga *embedded device* (Raspberry Pi, home, otomotif, dan banyak lagi) [28].

Flutter menggunakan mesin *rendering* Skia 2D yang disponsori oleh Google dan bekerja dengan berbagai jenis platform perangkat keras dan perangkat lunak dan juga digunakan oleh Google Chrome, Chrome OS, Android, Mozilla Firefox,

dan lainnya. Skia menggunakan render jalur berbasis *CPU* dan juga mendukung *backend* yang dipercepat *OpenGL ES2* [28].

2.2.13 Python

Python merupakan bahasa pemrograman serba guna dengan sintaks yang indah. Python dikembangkan oleh Guido Van Rossum dan diperkenalkan pada tahun 1991. Python diciptakan dengan kemampuan *exception handling* atau penanganan kesalahan dan mengutamakan sintaks yang mudah dibaca (*readability)* dan dimengerti. Python tersedia di berbagai *platform* termasuk *Windows, Linux,* dan *MacOS* [29]. Salah satu kelebihan python adalah memiliki banyak fungsi *built-in* yang dapat mempercepat pembuatan perangkat lunak.

Python banyak digunakan untuk *data science, machine learning*, pembuatan *website* dan *software*, *task automation* dan banyak lagi. Kelebihan python adalah mendukung *Object Oriented Programming*, lebih fleksibel, memiliki banyak pustaka, dan menyediakan *interface cross-platform*. Selain itu Python juga lebih efisien karena python dibuat menggunakan kode yang dikompilasi untuk meningkatkan efisiensi [30].

2.2.14 PyTorch

PyTorch merupakan pustaka python yang membantu untuk membangun projek *deep learning*. PyTorch menekankan fleksibilitas dan memungkinkan model *deep learning* diekspresikan dalam python *idiomatic* [31]. PyTorch dioptimalkan untuk *deep learning* menggunakan *GPU* dan *CPU* [32].

PyTorch memiliki berbagai komponen yang dapat dilihat pada Tabel 2.2.

Tabel 2.2 Komponen – komponen PyTorch

Komponen	Deskripsi
torch	Pustaka tensor seperti NumPy dengan dukungan *GPU*
torch.autograd	Pustaka diferensiasi otomatis berbasis *tape* yang mendukung semua operasi

Komponen	Deskripsi
	tensor di torch
torch.jit	*TorchScript* yang dioptimalkan untuk membuat model serial dan dioptimalkan dari kode PyTorch
torch.nn	Pustaka *neural network* yang terintegrasi dengan *autograd* yang dirancang untuk meningkatkan fleksibilitas
torch.multiprocessing	Python *multiprocessing* tetapi dengan *memory sharing* dari tensor torch yang berguna untuk memuat data
torch.utils	*DataLoader* dan fungsi utilitas untuk meningkatkan kenyamanan

2.2.15 Application Programming Interface (API)

Application Programming Interface, atau yang lebih dikenal sebagai API, berfungsi sebagai perantara yang memungkinkan berbagai pihak atau komponen dalam sistem komputer untuk berinteraksi dan saling terhubung. API berisi prosedur, fungsi, cara komunikasi atau peralatan untuk komunikasi. Penggunaan API sering melibatkan pengiriman dan penerimaan data melalui jaringan, dan dalam konteks ini, banyak API yang mengadopsi protokol *HyperText Transfer Protocol* (HTTP). Metode HTTP, juga dikenal sebagai HTTP method, mengacu pada jenis permintaan yang dikirim oleh klien (client) untuk menginstruksikan server tentang tindakan yang diinginkan.

Semakin berkembang, muncul konsep *Representational State Transfer* (REST), yang merupakan pendekatan arsitektural untuk merancang API. REST API mengikuti prinsip-prinsip REST, yang meliputi identifikasi sumber daya melalui URI, penggunaan metode HTTP yang sesuai, representasi data dalam format seperti JSON atau XML, dan prinsip *statelessness*. Dengan demikian, API,

terutama dalam bentuk REST API, memainkan peran sentral dalam menyederhanakan interaksi dan integrasi antara berbagai komponen perangkat lunak dalam lingkungan yang semakin kompleks. Beberapa HTTP method yang biasa digunakan pada REST API dapat dilihat di Tabel 2.3.

Tabel 2.3 HTTP Method yang umum digunakan

HTTP Method	Keterangan
get	Mengambil atau mencari *record* di *server*
post	Membuat *record* baru di *server*
put	Mengubah seluruh atribut *record* yang sudah ada di *server*
patch	Merubah sebagian atribut *record* yang sudah ada di *server*
delete	Menghapus *record* di *server*

2.2.16 FastAPI

FastAPI adalah *web framework* yang modern, cepat dan berkinerja tinggi untuk membangun API yang efisien dan kuat menggunakan bahasa pemrograman Python. FastAPI menggunakan uvicorn sebagai *server* yang akan menjalankan API [33]. FastAPI juga menggunakan dokumen API otomatis interaktif yang disediakan oleh SwaggerUI sehingga membuat FastAPI lebih intuitif.

Penggunaan FastAPI dapat mengurangi kompleksitas pengembangan API dengan menyediakan alat yang efisien dan modern. Dengan dukungan terintegrasi untuk validasi tipe data, penanganan permintaan asinkron, serta dokumentasi otomatis, FastAPI menghilangkan banyak tugas yang biasanya memakan waktu saat membangun dan memelihara API. Dengan menggabungkan Python dan konsep-konsep modern dari REST, FastAPI memungkinkan pengembang untuk dengan mudah mewujudkan proyek-proyek API yang baik dan efisien.

2.2.17 Google Cloud Platform

Google Cloud Platform adalah kumpulan layanan komputasi awan Google untuk membangun, mengelola, dan mengimplementasikan aplikasi dengan fleksibilitas dan efisiensi. Didalamnya termasuk komputasi, *cloud storage, big*

data, *artificial intelligence* (AI), dan jaringan [34]. Dalam konteks penelitian ini, peneliti menggunakan beberapa layanan dari Google Cloud Computing untuk mengoptimalkan dan mengelola komponen aplikasi yang digunakan.

Salah satu layanan yang digunakan adalah Google Container Registry, yang berperan sebagai tempat penyimpanan untuk kontainer Docker yang berisi aplikasi API dan model *machine learning*. Kontainer Docker adalah suatu cara yang efisien dan terisolasi untuk menjalankan aplikasi dan semua dependensinya, sehingga mengizinkan portabilitas dan konsistensi di berbagai lingkungan. Google Container Registry memungkinkan peneliti untuk dengan mudah menyimpan dan mengelola kontainer ini, serta memungkinkan akses yang aman dan terkendali ke kontainer tersebut.

Selain itu, layanan Google Cloud Run juga digunakan dalam penelitian ini. Google Cloud Run adalah layanan yang memungkinkan peneliti untuk menjalankan kontainer Docker secara otomatis dan skalabel di lingkungan *cloud*. Dengan menggunakan Google Cloud Run, kontainer Docker yang berisi API dan model *machine learning* dapat dijalankan dan dikelola dengan mudah tanpa perlu khawatir tentang infrastruktur dan skala, karena Google Cloud Run akan menangani proses manajemen otomatis. Ini memungkinkan peneliti untuk fokus pada pengembangan dan fungsionalitas aplikasi tanpa harus terlalu terikat dengan aspek infrastruktur yang kompleks.

2.2.18 Firebase

Firebase adalah suatu layanan cloud serta *backend* dari Google yang memungkinkan untuk mendapatkan data terorganisir dan *real-time* untuk aplikasi mobile. Firebase dapat diterapkan pada platform IOS, Android, dan bahkan *platform* web. Mempunyai berbagai alat dan layanan yang melibatkan berbagai aspek pembangunan aplikasi, Firebase meliputi sejumlah fitur, dari basis data hingga otentikasi pengguna, analitik, pemberitahuan, dan lebih banyak lagi. Diantara berbagai fitur ini, Firebase memiliki fitur inti berupa database *real-time* yang menggunakan arsitektur NoSQL dan memfasilitasi sinkronisasi instan antara aplikasi dan perangkat. Tidak hanya itu, Firebase juga menyediakan layanan

otentikasi yang aman serta mudah diintegrasikan, yang memungkinkan para pengembang untuk mengatur identitas pengguna dengan mudah, tanpa harus memulai dari awal.

Secara keseluruhan, Firebase merupakan pilihan yang banyak digunakan oleh para pengembang dalam mengembangkan aplikasi modern. Melalui alat-alat yang dapat diadaptasi dan integrasi yang baik dengan berbagai platform, Firebase mempercepat proses pembangunan, meningkatkan kualitas aplikasi, dan menyediakan pendekatan efisien untuk mengelola berbagai aspek dari aplikasi tersebut.

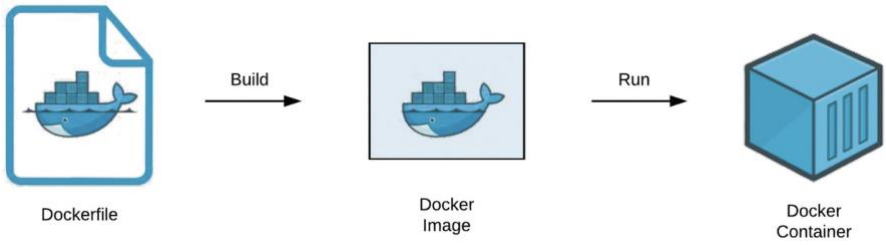
2.2.19 ChatGPT

ChatGPT (*Chat Generative Pre-Trained Transformer*) merupakan program komputer berbasis teks yang menggunakan model pengolahan bahasa alami canggih yang dikembangkan oleh OpenAI. ChatGPT memiliki 175 miliar parameter dan dilatih dengan menggunakan data yang sangat banyak. Chat GPT dilatih agar dapat menghasilkan respons percakapan yang alami terhadap input dari pengguna. Kemampuan ChatGPT untuk memahami dan merespons input bahasa alami ini adalah fitur utama. ChatGPT menggunakan NLP (*Natural Language Processing*) untuk menganalisis input pengguna dan memberikan respons yang relevan, terasa alami dan intuitif [35].

Tujuan ChatGPT adalah sebagai asisten percakapan yang dapat menjawab pertanyaan, memberikan informasi, memberikan saran, dan berinteraksi dengan pengguna dengan cara yang mirip dengan percakapan manusia. Model ini dirancang untuk menafsirkan dan memahami teks yang diberikan, serta menghasilkan tanggapan yang relevan dan sesuai. ChatGPT diimplementasikan sebagai chatbot yang dapat diakses melalui berbagai platform seperti aplikasi mobile, situs web, atau layanan pesan. Pengguna dapat berinteraksi dan menerima respons secara real time dengan ChatGPT melalui teks atau suara.

2.2.20 Docker

Docker adalah sebuah aplikasi virtualisasi yang mengabstraksikan aplikasi ke dalam container. Container ini merupakan sebuah platform terpadu yang mencakup *software tools* dan *dependency* untuk mengembangkan aplikasi atau perangkat lunak [34]. Proses menjalankan *docker container* seperti dapat dilihat pada Gambar 2.12.



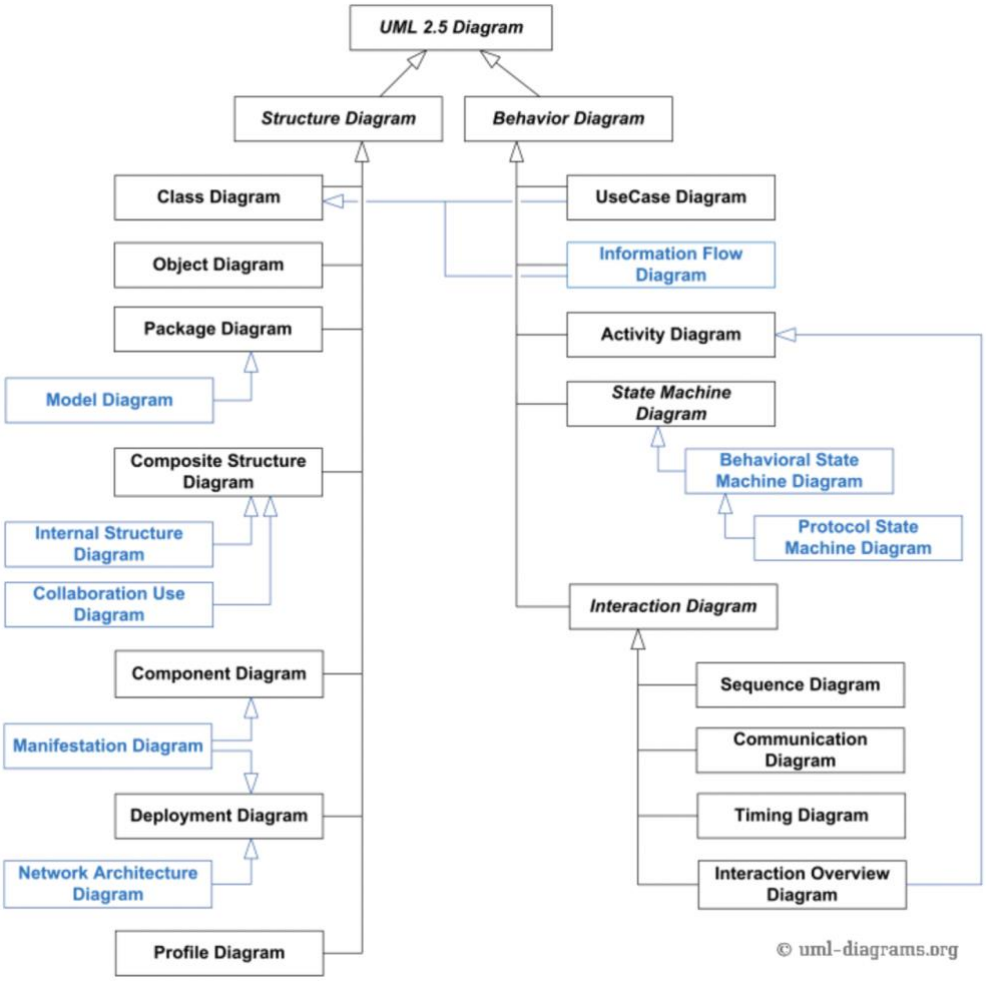
Gambar 2.12 Proses membuat image dan menjalankan *container* [34]

- *Dockerfile*: sebuah *file* teks yang berisi langkah-langkah yang dijalankan secara berurutan untuk mengonfigurasi lingkungan dalam *image* dan memuat aplikasi ke dalamnya.
- *Docker image*: paket lengkap yang berisi sebuah aplikasi atau perangkat lunak beserta semua dependensinya yang diperlukan untuk dijalankan. Sebuah image berfungsi sebagai *template* atau *blueprint* untuk membuat dan menjalankan *container*..
- *Docker container*: *environment* untuk mengemas aplikasi yang mencakup *system tools*, *library*, *code*, *runtime*, dan konfigurasi.

2.2.21 Unified Modelling Language (UML)

Unified Modelling Language (UML) adalah bahasa standar yang digunakan untuk merancang, memvisualisasikan, dan mendokumentasikan sistem melalui pemodelan visual. UML digunakan sebagai sarana perancangan dalam pengembangan sistem berorientasi objek. Dengan menggunakan UML, diharapkan pengembangan perangkat lunak dapat menjadi lebih mudah dan mampu memenuhi kebutuhan pengguna dengan tepat, lengkap, dan efektif [36]. Berdasarkan diagram UML 2.5 [37], diagram UML diklasifikasikan menjadi dua

kategori utama yaitu diagram struktur (*structure diagrams*) dan diagram perilaku (*behavior diagrams*) seperti yang ditunjukkan pada Gambar 2.13.



Gambar 2.13 Klasifikasi diagram UML 2.5 [37]

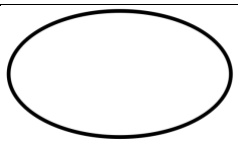
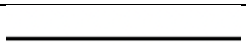
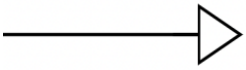
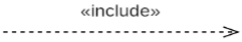
Pada penelitian ini, menggunakan diagram UML dengan kategori perilaku (*behavior diagrams*) yaitu *use case diagram* dan *activity diagram*.

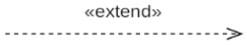
1. Use Case Diagram

Use case diagram digunakan untuk mengilustrasikan serangkaian langkah yang wajib atau dapat dilakukan oleh subjek sistem saat berinteraksi dengan satu atau lebih pengguna eksternal (aktor). Tujuan dari interaksi ini adalah menghasilkan hasil-hasil yang bernilai dan dapat diperhatikan bagi aktor atau

pihak-pihak yang terkait dengan sistem [37]. Beberapa komponen dari diagram kasus penggunaan dapat ditemukan dalam Tabel 2.4.

Tabel 2.4 Tabel *use case*

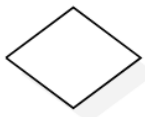
Nama	Simbol	Deskripsi
Aktor		Segala hal yang akan memanfaatkan sistem dan melakukan tindakan. Aktor bisa berupa individu, sistem, atau perangkat yang memiliki peran dalam menjalankan operasi sistem dengan sukses.
Use case		Deskripsi fungsional dalam suatu sistem, bertujuan agar baik pembuat maupun pengguna dapat saling memahami mengenai urutan langkah sistem yang akan diciptakan.
Use case subject		Menunjukkan cakupan tugas dan batasan-batasan dari sistem yang telah dibuat.
Hubungan asosiasi		Menggambarkan hubungan yang terjalin antara aktor khusus dengan kasus penggunaan tertentu.
Hubungan generalisasi		Menunjukkan kaitan antara dua aktor atau dua kasus penggunaan di mana salah satunya mewarisi dan memperluas karakteristik dari yang lain.
Hubungan *include*		Menunjukkan bahwa suatu kasus penggunaan memerlukan dukungan fungsionalitas dari kasus penggunaan lainnya dalam gambaran diagram.

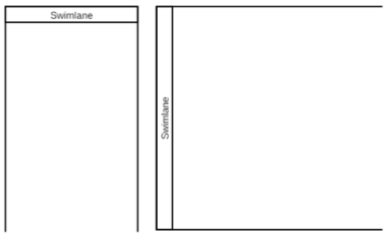
Nama	Simbol	Deskripsi
Hubungan *extend*		Menunjukkan bahwa sebuah kasus penggunaan memiliki independensi dan merupakan perluasan fungsionalitas dari kasus penggunaan lain.

2. Activity Diagram

Activity diagram berperan dalam menampilkan urutan serta kondisi untuk mengatur perilaku pada tingkat yang lebih terperinci daripada klasifikasi yang memiliki perilaku tersebut. Hal ini umumnya disebut sebagai model aliran kontrol (*control flow*) dan model aliran objek (*object flow*). Beberapa unsur dari diagram aktivitas dapat ditemukan dalam Tabel 2.5.

Tabel 2.5 Tabel *activity diagram*

Nama	Simbol	Deskripsi
Initial node		Menandakan keadaan permulaan, langkah pertama, atau titik awal dari sebuah diagram aktivitas atau sistem.
Activity		Menandakan suatu tindakan dalam sistem, biasanya *activity* ini dimulai dengan kata kerja yang menggambarkan kegiatan yang sedang dilakukan.
Decision		Menggambarkan situasi di mana suatu sistem memiliki berbagai pilihan atau jika terdapat beberapa opsi aktivitas yang dapat dipilih berdasarkan kondisi tertentu.

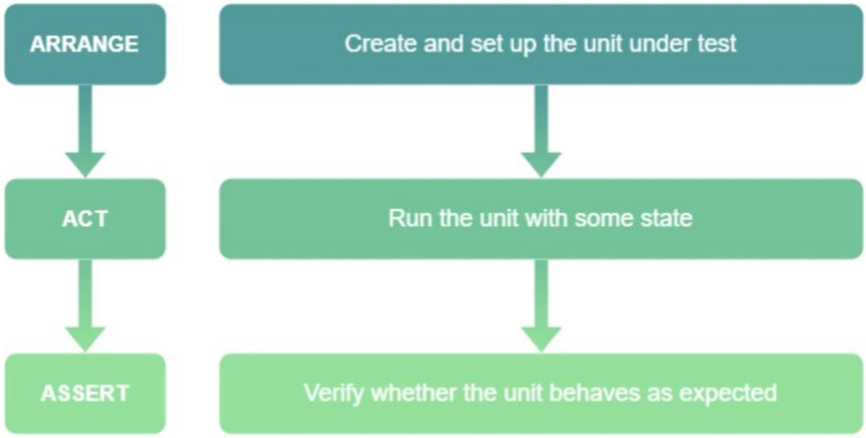
Nama	Simbol	Deskripsi
Join		Digunakan untuk menggabungkan jalur atau aliran yang sebelumnya telah dipisah atau dibagi menjadi beberapa bagian oleh aliran dari lebih dari satu aktivitas.
Control flow		Digunakan untuk mengindikasikan aliran kontrol dari suatu *action* atau *activity* ke *action* atau *activity* lainnya. Setelah *action* atau *activity* selesai dilaksanakan, *flow* akan melanjutkan ke *action* atau *activity* berikutnya.
Final node		Elemen yang mengindikasikan bahwa suatu proses telah selesai atau mencerminkan status akhir dari tindakan dalam sebuah sistem.
Swimlane		Merupakan struktur organisasi dan pemecah bagi diagram aktivitas yang terbagi menjadi baris dan kolom, memberikan tanggung jawab kepada setiap objek untuk melakukan aktivitas tertentu.

2.2.22 Pengujian Aplikasi

Dalam pengembangan aplikasi atau perangkat lunak, pengujian merupakan salah satu tahap yang sangat penting, terutama untuk mengetahui kualitas aplikasi dan untuk memastikan aplikasi sudah berjalan sesuai dengan kebutuhan [38]. Hal ini bertujuan untuk memastikan aplikasi atau perangkat lunak berjalan dengan baik dan tidak terdapat kegagalan atau cacat yang dapat mempengaruhi kenyamanan maupun keamanan pengguna. Pada penelitian ini, tahap pengujian yang digunakan adalah pengujian kode yang mencakup *unit testing, widget testing, dan integration testing* dan pengujian yang melibatkan responden yaitu pengujian *end-to-end testing*, dan pengujian aplikasi mendeteksi gambar.

1. Pengujian Unit (*Unit Testing*)

Pengujian unit atau *unit testing* merupakan sebuah pengujian bertipe *white box* dan merupakan proses pengujian di mana unit kecil kode diuji dengan beberapa kasus penggunaannya untuk menguji keandalannya. Unit dapat berupa fungsi, kelas, variabel dan sebagainya.



Gambar 2.14 Fase pengujian *unit testing*

Seperti yang dapat dilihat pada Gambar 2.14, *Unit Testing* memiliki beberapa fase yaitu fase *Arrange, Act,* dan *Assert.* Pada fase *Arrange*, hanya perlu membuat objek dari unit yang diuji dan menyiapkan prasyarat untuk pengujian, yaitu mengatur variabel status, mengatur *mock* dan sebagainya. Pada fase *Act*, menjalankan *unit* dengan beberapa *state* (melewati argumen) dan menyimpan

hasilnya jika ada. Pada fase *Assert*, memverifikasi apakah unit berperilaku seperti yang diharapkan. Akan tetapi, fase *Arrange* and *Act* tidak bersifat wajib.

Untuk melakukan pengujian *unit* pada penelitian ini, peneliti menggunakan pustaka *flutter_test*. Flutter_test adalah sebuah pustaka atau pustaka yang disediakan oleh Flutter untuk menguji aplikasi Flutter secara otomatis. Pustaka ini memungkinkan untuk menulis dan menjalankan berbagai jenis tes otomatis untuk menguji kode dan antarmuka pengguna (UI) aplikasi Flutter. Dengan menggunakan *flutter_test*, dapat menulis tes untuk menguji *widget*, logika bisnis, interaksi pengguna, dan banyak aspek lain dari aplikasi. Tes-tes ini membantu untuk memastikan bahwa aplikasi berfungsi sebagaimana mestinya dan memberikan hasil yang diharapkan.

Flutter_test menyediakan berbagai macam fungsi dan metode yang memungkinkan untuk mengatur keadaan (*state*) aplikasi, mensimulasikan tindakan pengguna, dan memverifikasi output yang dihasilkan oleh *widget* dan komponen lainnya. Tes-tes ini dapat dijalankan di perangkat fisik, *emulator*, atau *simulator* untuk berbagai platform seperti Android, IOS, dan *website*.

2. Pengujian *Widget* (*Widget Testing*)

Widget testing adalah pengujian widget pada aplikasi Flutter untuk memastikan widget berperilaku sesuai dengan yang diharapkan. Pada widget testing ini akan menguji widget dan tampilan antarmuka (UI) secara otomatis. *Widget testing* pada Flutter menggunakan kerangka pengujian "*flutter_test*" dan "*WidgetTester*" untuk mengatur lingkungan tes, membangun widget, mengirim input ke *widget*, dan memeriksa hasil keluaran dari *widget*. Dengan *widget testing*, *widget* dapat secara otomatis diuji pada berbagai kasus dan situasi yang mungkin terjadi dalam aplikasi, sehingga memastikan bahwa antarmuka pengguna berfungsi dengan baik sebelum dirilis kepada pengguna akhir.

3. Pengujian Integrasi (*Integration Testing*)

Integration testing merupakan jenis pengujian perangkat lunak yang bertujuan untuk menguji apakah berbagai komponen aplikasi bekerja dengan benar saat diintegrasikan dalam satu kesatuan. Fokus dari pengujian ini adalah

pada pengujian keseluruhan aplikasi dan bagaimana seluruh bagian aplikasi berinteraksi satu sama lain.

Pentingnya integration testing dalam aplikasi Flutter adalah untuk memastikan bahwa semua bagian aplikasi, seperti *widget*, layar, *routing*, dan fitur lainnya, berfungsi dan terintegrasi dengan baik. Dalam aplikasi Flutter, *integration testing* membantu memastikan bahwa aplikasi dapat berjalan dengan benar pada level *widget* dan bahwa fitur-fitur yang kompleks yang melibatkan beberapa *widget* dapat berinteraksi dengan benar.

Untuk melakukan *integration testing* pada flutter telah disediakan pustaka bernama *flutter_test* dan *integration_test*. Dengan menggunakan kedua pustaka ini, dapat dilakukan skenario pengujian yang lebih kompleks, seperti menguji navigasi antar layar, mengirimkan input pengguna, dan memverifikasi output yang diharapkan.

4. Pengujian *End-to-end*

End-to-end testing adalah metode pengujian yang dilakukan untuk memastikan bahwa semua fungsi dan fitur yang dilakukan oleh pengguna aplikasi berjalan seperti yang diharapkan. *End-to-end testing* dilakukan oleh pengguna berdasarkan sudut pandang pengguna tersebut dan mensimulasikan penggunaan aplikasi pada keadaan nyata (*real-world skenario*). Pada pengujian *end-to-end* ini, peneliti menggunakan pengujian dengan teknik *usability testing think aloud*.

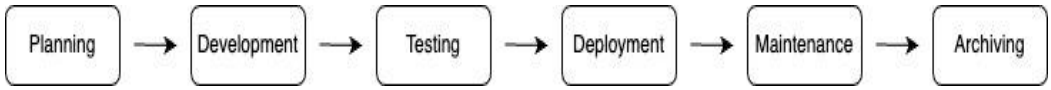
Think aloud merupakan teknik pengumpulan data dari pengguna secara eksternal dengan menilai apa yang dipikirkan dan dirasakan oleh pengguna dalam berinteraksi dengan sistem. Dengan teknik *think aloud* ini, pengembang mendapatkan evaluasi dan rekomendasi dari pengguna tentang pemahaman pikiran pengguna dalam menggunakan aplikasi dari tugas-tugas yang diberikan saat pengujian dan dari segi desain interaksinya.

5. Pengujian Akurasi Aplikasi

Pengujian akurasi aplikasi dalam mendeteksi gambar dilakukan dengan menguji aplikasi untuk mendeteksi dan mengidentifikasi objek pada gambar yang diambil dari berbagai sudut dan juga intensitas cahaya yang berbeda.

2.2.23 Mobile Application Development Lifecycle (MADLC)

Untuk menghasilkan aplikasi yang sesuai kebutuhan pengguna, diperlukan metode yang sesuai dengan kaidah pengembangan perangkat lunak. Salah satu metode pengembangan perangkat lunak adalah dengan menggunakan metode *Software Development Lifecycle* (SDLC). Menurut Katarzyna Kubis[39], metode SDLC memiliki beberapa fase seperti yang ditunjukkan pada gambar 2.15 yaitu *fase planning, development, testing, deployment, maintenance,* dan *archiving*.



Gambar 2.15 Tahapan *Software Development Lifecycle (SDLC)* [39]

Salah satu pengembangan dari model SDLC adalah *Mobile Application Development Lifecycle* (MADLC) yang dikhususkan untuk pengembangan aplikasi mobile dan memiliki fungsi yang lebih kompleks dan berbeda dari aplikasi jenis lainnya. Menurut Tejas Vithani dan Anand Kumar [40] MADLC memiliki tujuh fase seperti yang ditunjukkan pada gambar 2.16:

Gambar 2.16 Fase *Mobile Application Development Lifecycle (MADLC)* [40]

a. Fase Identifikasi (*Identification Phase*)

Fase identifikasi merupakan fase pengumpulan dan pengkategorian ide. Tujuan utama pada fase identifikasi adalah mendapatkan ide baru atau perbaikan pada aplikasi yang sudah ada. Ide bisa berasal dari pengembang ataupun dari calon pengguna. Ide didokumentasikan kemudian diteruskan ke fase desain.

b. Fase Desain (*Designing Phase*)

Pada fase desain, dilakukan pembuatan desain awal aplikasi berdasarkan ide yang didapatkan pada fase identifikasi. Hal yang sangat penting pada fase desain ini adalah membuat *storyboard* untuk interaksi antarmuka pengguna (*user interface*). *Storyboard* ini akan menggambarkan mengenai alur aplikasi. Hasil dari fase desain didokumentasikan untuk kemudian diteruskan pada tahap pengembangan.

c. Fase Pengembangan (*Development Phase*)

Pada fase pengembangan, mulai dilakukan proses pengkodean aplikasi berdasarkan hasil desain pada tahap desain. Proses pengkodean dapat dibagi menjadi dua bagian, yaitu tahap pengkodean untuk persyaratan fungsional (*Functional Requirements*) dan persyaratan antarmuka (*User Interface Requirements*). Hasil dari fase pengembangan kemudian diteruskan ke fase pembuatan prototipe.

d. Fase *Prototyping* (*Prototyping Phase*)

Pada fase *prototyping*, dilakukan analisis semua persyaratan fungsional prototipe. Prototipe diuji dan diberikan kepada calon pengguna untuk diberikan umpan balik (*feedback*). Setelah umpan balik diterima, perubahan diimplementasikan melalui fase pengembangan. Fase pengembangan, prototipe dan pengujian adalah fase yang terus diulang sampai prototipe akhir siap. Hasil dari prototipe didokumentasikan dan diteruskan ke fase pengujian.

e. Fase Pengujian (*Testing Phase*)

Fase pengujian merupakan fase yang paling penting. Pada fase ini, prototipe diuji pada *emulator* atau *simulator* dan kemudian diuji pada perangkat nyata. Hasil didokumentasikan dan diteruskan kepada calon pengguna untuk dimintai umpan balik.

f. Fase Penyebaran (*Deployment Phase*)

Fase penyebaran merupakan fase terakhir dari proses pengembangan aplikasi. Fase penyebaran ini dilakukan setelah seluruh pengujian telah selesai dilakukan dan telah mendapatkan umpan balik (*feedback*) dari pengguna. Aplikasi pada tahap pengembangan telah siap untuk disebarkan melalui toko aplikasi seperti Google Play Store ataupun App Store.

g. Fase Pemeliharaan (*Maintenance Phase*)

Fase pemeliharaan merupakan tahap yang paling akhir dari metode MADLC dan merupakan proses yang akan terus berkelanjutan. Umpan balik dari pengguna dikumpulkan dan dianalisis untuk dilakukan perbaikan aplikasi dalam bentuk peningkatan keamanan, perbaikan bug, penambahan fitur, perubahan antarmuka aplikasi ataupun peningkatan aplikasi harus dilakukan secara berkala dalam bentuk pembaruan aplikasi.

Berikut merupakan perbandingan metode MADLC dengan metode SDLC yang lain dapat dilihat pada tabel 2.6.

Tabel 2.6 Perbandingan metode pengembangan aplikasi [41]

Variabel	Spiral Model	Iterative Model	Agile Model	MADLC	Model Driven
Environment	Stabil	Adaptif	Adaptif	Adaptif	Adaptif
Kecocokan	Proyek besar, rumit, dan mahal	Aplikasi yang dinamis dan rumit	Untuk organisasi kecil, proyek pengembangan dan non-sekuensial	Untuk aplikasi yang memiliki ide serupa dan sudah ada di pasar	Non-pakar dapat dengan mudah membuat aplikasi seluler khusus.
Arsitektur	Dirancang untuk kebutuhan saat ini dan yang akan datang	Dirancang ketika persyaratan sistem lengkap didefinisikan dan dipahami dengan	Dirancang untuk kebutuhan saat ini	Didesain untuk kebutuhan pengguna dan pengguna sendiri memberikan ide	Didesain untuk lebih fokus pada desain dan logika aplikasi

Variabel	Spiral Model	Iterative Model	Agile Model	MADLC	Model Driven
		jelas.		bagaimana mengembangkan ide tersebut lebih lanjut dan dinamis	
Ukuran Aplikasi	Besar	Besar	Kecil	Kecil	Kecil
Waktu untuk Dipasarkan	Panjang	Pendek	Pendek	Pendek	Pendek
Beragam Platform	N/A	N/A	N/A	Ya	Tidak
Dokumentasi	Berat	Tinggi	Rendah	Tinggi	Rendah
Keterlibatan pengguna	Sepanjang *life cycle*	Pada akhir tiap iterasi	Konstan umpan balik dari pengguna	Konstan feedback dari pengguna	Tidak banyak

Seperti yang dapat dilihat pada tabel 2.6, metode MADLC merupakan metode yang bersifat adaptif sehingga metode ini bisa menerima perubahan secara berkala. Arsitektur MADLC cocok digunakan untuk pengembangan perangkat lunak terutama mobile dimana pengguna dapat ikut serta dalam menentukan ide aplikasi dan ide tersebut akan dianalisis sehingga metode ini cocok untuk pengembangan aplikasi yang sesuai dengan kebutuhan pengguna. Metode MADLC juga baik dalam hal keberlanjutan pengembangan aplikasi karena proses dokumentasi yang lengkap sehingga akan memudahkan dalam proses pengembangan aplikasi selanjutnya.