

BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

2.1.1 *Deep Learning*

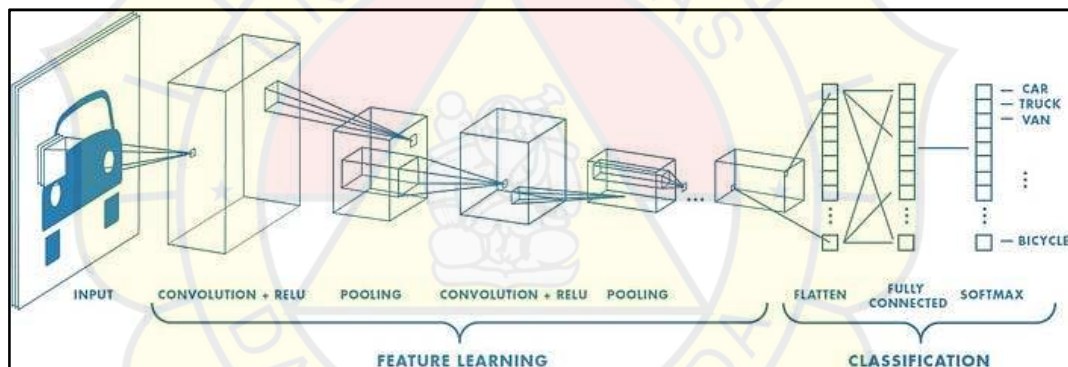
Deep Learning adalah bidang baru *machine learning* yang semakin populer belakangan ini (Arif M dkk., 2020) *Deep learning* adalah sub-bidang dari *machine learning* yang menggunakan jaringan saraf tiruan dengan banyak lapisan (*deep neural networks*) untuk memodelkan dan memahami data yang kompleks. Konsep utama dalam *deep learning* adalah penggunaan jaringan saraf tiruan yang terdiri dari banyak *neuron* yang saling terhubung (Goodfellow I dkk., 2016). Salah satu implementasi dari *deep learning* yaitu pada data citra. Klasifikasi citra bertujuan untuk mengkategorikan citra pada kelas-kelas tertentu (Setiawan W, 2021).

Deep Learning adalah teknik komputer untuk mengekstrak dan mengubah data dengan menggunakan beberapa lapisan jaringan saraf. Masing-masing lapisan ini mengambil input dari lapisan sebelumnya dan semakin menyempurnakannya. Lapisan dilatih oleh algoritma yang meminimalkan kesalahan mereka dan meningkatkan akurasi mereka (Howard J & Gugger, 2020). *Deep learning* memiliki tingkat akurasi lebih tinggi dibandingkan lainnya. Pada *deep learning*, semakin banyak data *training* yang digunakan, semakin tinggi akurasi pengenalan yang didapatkan. Salah satu metode *deep learning* adalah *Convolutional Neural Network* (CNN) (Setiawan W, 2021).

2.1.2 *Convolutional Neural Network (CNN)*

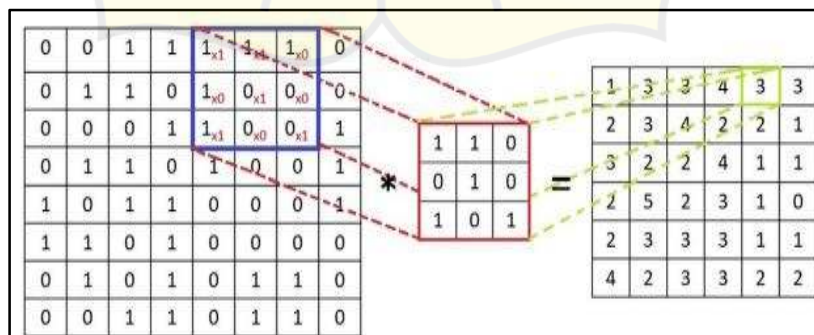
Pada buku yang ditulis oleh (Huawei Technologies Co., Ltd., 2023) yang berjudul “*ARTIFICIAL INTELLIGENCE TECHNOLOGY*”, dijelaskan tentang

Convolutional Neural Network (CNN) adalah jenis jaringan saraf *feed-forward* yang memiliki kinerja baik dalam pengolahan gambar. *Convolutional neural network* sendiri diambil dari operasi matematika yang bernama *convolutional*. Saat ini, CNN telah menjadi salah satu topik yang sangat dibahas di banyak bidang ilmiah, terutama dalam pengenalan pola. Struktur CNN telah banyak digunakan di banyak bidang karena dapat menghindari *preprocessing* gambar yang kompleks dan memasukkan gambar asli secara instan. Arsitektur CNN terdiri dari empat *layer* yaitu *convolution layer*, *pooling layer*, *fully connected layer*, dan *nonlinearity layer* Purwono dkk, (2022). Ilustrasi arsitektur metode CNN ditunjukkan pada Gambar 2.1.



Gambar 2.1 Arsitektur CNN (Purwono dkk., 2022)

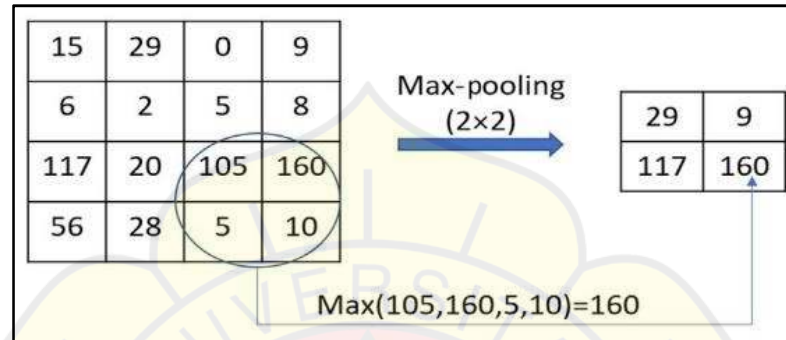
1. Convolutional Layer



Gambar 2.2 Convolutional Layer (Purwono dkk., 2022)

Convolutional Layer adalah operasi inti dalam CNN di mana filter diterapkan pada gambar input dengan menggesernya secara berurutan. Proses ini menghasilkan peta fitur yang menyoroiti keberadaan fitur-fitur yang dipelajari oleh filter di setiap lokasi dalam gambar.

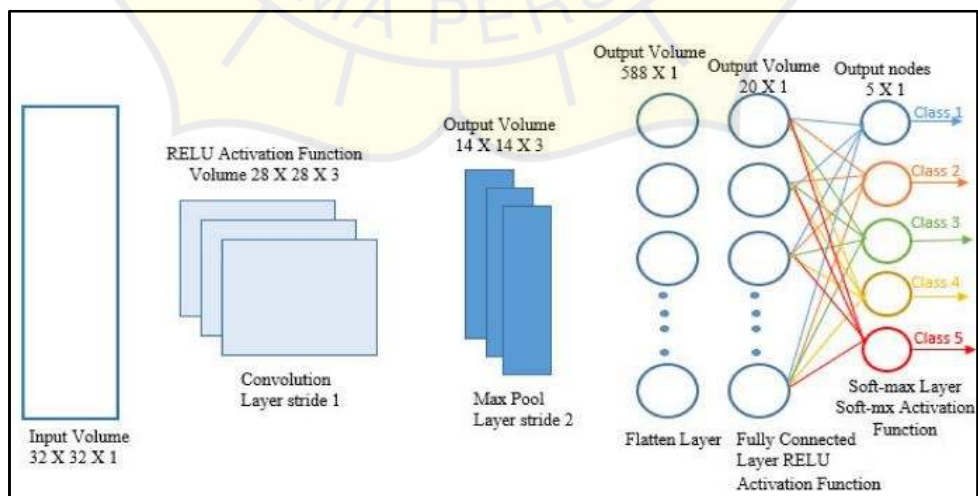
2. Pooling Layer



Gambar 2.3 *Pooling Layer* (Purwono dkk., 2022)

Setelah *Convolutional*, *pooling layer* diterapkan untuk mengurangi dimensi spasial dari *feature map* yang dihasilkan. Hal ini dilakukan dengan mengambil nilai maksimum (*max pooling*) atau rata-rata (*average pooling*) di dalam jendela tertentu pada *feature map*.

3. Fully Connected Layer



Gambar 2.4 *Fully Connected Layer* (Purwono dkk., 2022)

Fully Connected Layer adalah jaringan saraf *feed-forward*, sementara *fully connected layer* terletak di lapisan terendah jaringan. Keluaran (*output*) dari lapisan penyatuan (*pooling layer*) atau lapisan konvolusi (*convolutional layer*) terakhir diratakan (*flatten*) sebelum dikirim sebagai masukan (*input*) ke *fully connected layer*. Metode untuk mempelajari kombinasi *non-linear* dari fitur tingkat tinggi yang dihasilkan oleh *output convolutional layer* adalah dengan menerapkan *fully connected layer*, yang mempelajari fungsi dengan ruang yang berpotensi *non-linear*. Teknik ini berguna untuk mempelajari kombinasi *non-linear* dari fitur tingkat tinggi yang dihasilkan oleh *output convolutional layer*, yang memiliki potensi untuk mempelajari fungsi *non-linear*.

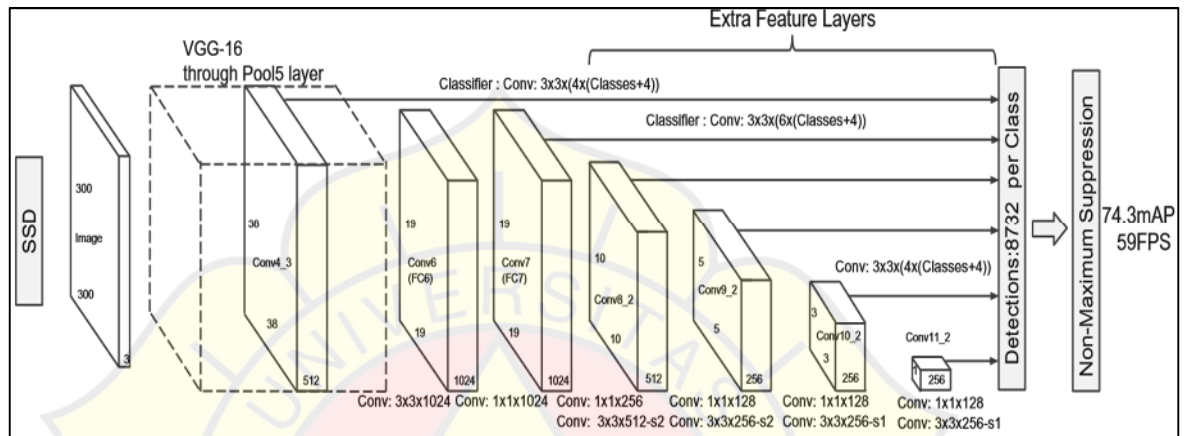
4. *Nonlinearity Layer (Activation Function)*

Aktivasi seperti ReLU (*Rectified Linear Activation*) diterapkan setelah setiap lapisan konvolusi untuk memperkenalkan *non-linearitas* ke dalam model. ReLU mengubah semua nilai negatif menjadi nol, memungkinkan jaringan untuk mempelajari pemetaan yang lebih kompleks.

2.1.3 *Single Shot Multibox Detector (SSD)*

Menurut Liu W (2016), SSD (*Single Shot MultiBox Detector*) adalah metode untuk mendeteksi objek dalam gambar menggunakan satu jaringan saraf dalam yang tunggal. Dengan mendiskritkan ruang *bounding box* menjadi seperangkat *default bounding box* yang memiliki rasio aspek dan skala yang berbeda berdasarkan lokasi *feature map*, SSD dapat mencapai deteksi objek dengan kecepatan *real-time* sambil mempertahankan akurasi yang kompetitif dengan teknik yang lebih kompleks. Arsitektur SSD berasal dari jaringan saraf VGG16,

yang dimodifikasi untuk menghasilkan berbagai *feature map* dengan berbagai resolusi. Beberapa *default bounding box* dengan berbagai rasio aspek dan skala menangani setiap *feature map*. Selanjutnya, dua lapisan regresi tambahan digunakan untuk mengklasifikasikan dan menempatkan *default bounding box* ini. Ilustrasi arsitektur SSD ditunjukkan pada Gambar 2.5.



Gambar 2.5 Arsitektur SSD (Liu W dkk., 2016)

1. Basic Network

Basic Network SSD didasarkan pada arsitektur VGG16. Arsitektur VGG16 terdiri dari beberapa *convolution block*, diikuti oleh *pooling layer*. *Convolution block* terdiri dari beberapa *convolution layer* dan *pooling layer*. *Convolution layer* mengekstrak fitur dari gambar, sedangkan *pooling layer* mengurangi dimensi *feature map*.

2. Feature Map

Basic Network SSD menghasilkan beberapa *feature map* dengan resolusi berbeda. *Feature map* ini dihasilkan oleh *pooling layer* dengan ukuran yang berbeda. *Feature map* dengan resolusi lebih tinggi memiliki ukuran spasial yang lebih kecil tetapi berisi informasi fitur yang lebih rinci. *Feature map*

dengan resolusi lebih rendah memiliki ukuran spasial yang lebih besar tetapi berisi informasi fitur yang lebih abstrak.

3. *Default Bounding Box*

Setiap *feature map* diproses oleh beberapa *default bounding box* dengan rasio aspek dan skala yang berbeda. *Default bounding box* ini adalah kotak persegi panjang yang ditempatkan di lokasi *grid* pada *feature map*. Rasio aspek *default bounding box* dipilih dari kumpulan nilai yang telah ditentukan sebelumnya. Skala *default bounding box* dipilih berdasarkan resolusi *feature map*.

4. Klasifikasi dan lokalisasi

Default bounding box diklasifikasikan dan dilokalkan oleh dua lapisan regresi tambahan. Lapisan klasifikasi memprediksi kelas objek untuk setiap kotak pembatas default. Lapisan lokalisasi memprediksi *offset default bounding box* untuk menyelaraskan dengan objek sebenarnya.

5. Deteksi objek

Hasil dari lapisan klasifikasi dan lokalisasi digabungkan untuk menghasilkan deteksi objek. Deteksi objek terdiri dari kotak pembatas dan kelas objek yang diprediksi.

Berdasarkan penelitian yang dilakukan Liu W dkk, (2016) arsitektur SSD mempunyai keuntungan dan kelemahan sebagai berikut:

Keuntungan dari arsitektur SSD

1. Kecepatan: SSD dapat mencapai deteksi objek dengan kecepatan *real-time* karena tidak memerlukan *region proposal*.

2. Akurasi: SSD mencapai akurasi yang sebanding dengan metode *state-of-the-art* lainnya.
3. Kesederhanaan: Arsitektur SSD relatif sederhana dan mudah diimplementasikan.

Kekurangan dari arsitektur SSD

1. Ketergantungan pada *feature map*: Kinerja SSD dapat bergantung pada kualitas *feature map*.
2. Sensitivitas terhadap skala: SSD mungkin kurang sensitif terhadap objek kecil.

2.1.4 Master Data

Master data adalah sebuah data yang menggambarkan entitas atau domain bisnis yang paling relevan dalam sebuah agensi, seperti pelanggan, produk, atau pemasok. Data ini mencakup karakteristik utama dari objek di dunia nyata. Entitas master data tunggal jarang diubah, contohnya sifat-sifat dari beberapa jenis material. Kelas data master cenderung lebih stabil dibandingkan dengan data transaksional. Master data menjadi referensi untuk data transaksional, di mana tidak ada data transaksional tanpa adanya master data. Selain itu, Dreibelbis mendefinisikan master data sebagai beberapa informasi paling berharga mengenai domain bisnis seperti pelanggan, pemasok, produk, akun, dan hubungan di antara mereka. Master data juga didefinisikan sebagai data perusahaan yang setiap domainnya mewakili informasi yang dibutuhkan dalam berbagai proses bisnis yang berbeda, di seluruh unit lembaga, dan antara sistem operasional serta sistem pendukung keputusan (Haneem F dkk., 2017).

2.1.5 React

Pada buku yang ditulis oleh (Gackenheimer C, 2015) dengan judul “*Introducing to React*” menjelaskan bahwa React adalah sebuah *framework* JavaScript. React pada awalnya dibuat oleh para insinyur di Facebook untuk mengatasi tantangan yang ada ketika mengembangkan antarmuka pengguna yang kompleks dengan *dataset* yang berubah dari waktu ke waktu. Ini bukanlah pekerjaan yang mudah karena, tidak hanya harus dapat dipelihara tetapi juga harus dapat diskalakan untuk bekerja pada skala Facebook. React sebenarnya lahir di organisasi iklan Facebook, di mana mereka telah menggunakan pendekatan *Model View Controller* (MVC) sisi klien tradisional. Aplikasi seperti ini biasanya terdiri dari pengikatan data dua arah bersama dengan *template rendering*. React mengubah cara pembuatan aplikasi seperti ini dengan membuat beberapa kemajuan yang berani dalam pengembangan web.

Ketika React dirilis pada tahun 2013, komunitas pengembangan web sangat tertarik dan tampaknya tidak senang dengan apa yang dilakukan React. Seiring dengan pergeseran mentalitas pengembangan *front-end*, React hadir dengan serangkaian fitur yang kaya yang membuat pembuatan aplikasi halaman tunggal atau antarmuka pengguna dapat didekati oleh para pengembang dengan berbagai tingkat keahlian, mulai dari mereka yang baru saja diperkenalkan dengan JavaScript, hingga mereka yang berpengalaman dalam dunia web. Ada beberapa hal penting yang perlu kita ketahui dari React seperti konsep *Components*, *virtual DOM*, *JSX*, dan *Flux*.

1. *Components*

Components adalah elemen dasar dari React, yang berfungsi sebagai fondasi untuk tampilan aplikasi.

2. *Virtual DOM (Document Object Model Virtual)*

Salah satu komponen utama React adalah *Virtual DOM (Document Object Model Virtual)*, yang memungkinkan aplikasi untuk *diupdate* sesuai dengan perubahan data atau interaksi pengguna. Meskipun Facebook menyadari bahwa *framework* tersebut tidak dapat memenuhi standarnya, Facebook berusaha untuk meningkatkan kinerjanya dengan membuat virtual DOM untuk *reconciliation*, yang digunakan untuk menghitung jumlah *minimum* perubahan yang diperlukan untuk mengubah DOM asli aplikasi.

3. JSX

JSX adalah sebuah lapisan transformasi yang digunakan untuk mengubah XML sintaks yang digunakan untuk menulis komponen React ke JavaScript sintaks yang digunakan untuk rendering elemen. Ini tidak diperlukan untuk React, tetapi dianggap bermanfaat untuk mempercepat proses pembuatan aplikasi. Itu menerima kelas React khusus dan tag HTML sederhana, mengubahnya menjadi elemen React yang sesuai.

4. Flux

Flux adalah arsitektur aplikasi Facebook yang memungkinkan data untuk berinteraksi secara bermakna dan sistematis dengan komponen-komponen React. Ini bukan arsitektur aliran data dua arah seperti *Model View Controller (MVC)*. Flux sangat penting untuk React karena menciptakan aliran data satu arah antara *dispatcher*, *store*, dan *view* React, mendorong penggunaan komponen React dengan cara yang tepat.

2.1.6 Python

Python adalah bahasa pemrograman tingkat tinggi yang diinterpretasikan dan digunakan dalam komputasi ilmiah dan teknik. Sintaksnya yang bersih dan mudah dibaca meningkatkan kemudahan pemeliharaan, mengurangi bug, dan memfasilitasi pengembangan kode yang cepat. Keterbacaan dan ekspresifitas ini sangat penting dalam komputasi eksploratif dan interaktif, yang menuntut perputaran cepat untuk menguji ide dan model (Johansson R, 2018).

2.1.7 Tensorflow

TensorFlow adalah *library* Python yang dikembangkan oleh Google untuk komputasi numerik yang cepat. Ini berfungsi sebagai fondasi untuk membuat model *deep learning* atau menggunakan pustaka pembungkus untuk menyederhanakan proses. Tidak seperti *library deep learning* lainnya seperti Theano, TensorFlow dirancang untuk penelitian, pengembangan, dan sistem produksi, termasuk proyek RankBrain dan DeepDream milik Google. TensorFlow dapat berjalan pada sistem CPU tunggal, GPU, perangkat seluler, dan sistem terdistribusi berskala besar (Brownlee J, 2016).

2.1.8 Firebase

Pada tahun 2014, Google menyelesaikan akuisisi sebuah perusahaan yang berbasis di San Francisco bernama Firebase, Inc. Firebase, Inc. menyediakan berbagai solusi pengembang yang dirancang untuk mempercepat integrasi fitur-fitur berbasis *cloud* ke dalam aplikasi seluler dan web. Setelah membeli perusahaan tersebut, Google menggabungkan layanan yang disediakan oleh Firebase dengan sejumlah fitur pelengkap yang sebelumnya telah disertakan sebagai bagian dari

Google Cloud Platform. Fitur gabungan dari kedua platform inilah yang sekarang dikenal dengan nama Firebase (Smyth N, 2017).

Firebase adalah *platform* pengembangan aplikasi seluler dan web yang didukung oleh Google. *Platform* Firebase membantu Anda mengembangkan aplikasi berkualitas tinggi dan fokus pada pengguna. Firebase mencakup berbagai produk, seperti *Realtime Database*, *Crash reporting*, *Cloud Firestore*, *Cloud Storage*, *Cloud functions*, *Authentication*, *Hosting*, *Test lab for Android*, and *Performance monitoring for iOS*, yang dapat digunakan untuk mengembangkan dan menguji aplikasi *Realtime* dengan berfokus pada kebutuhan pengguna, bukan pada kerumitan teknis (Tanna M & Sign H, 2018).

2.1.9 Google Colaboratory (Google Colab)

Google telah membuat Google Colaboratory yang merupakan kerangka kerja online di mana seseorang dapat menulis, mengeksekusi kode *deep learning* dan *machine learning*. Di sini, berbagai versi python dan lingkungan *runtime* yang berbeda tersedia. Selain itu, ia dapat mengunduh kumpulan data yang lebih besar langsung dari server ke *drive* Google dengan kecepatan yang sangat tinggi. Anda dapat memasang *drive* Anda dengan Google Colab dan dapat mengambil file yang diperlukan setelah otentikasi Anda. Berapa lama jumlah memori dan komputasi yang disediakan oleh Google tersedia untuk pengguna juga dibahas. Ini menyediakan komputasi berkecepatan sangat tinggi dan kemudian menyimpan model Pembelajaran (Kanani P & Padole M, 2019).

2.1.10 Pemodelan Sistem UML

2.1.10.1 *Unified Modeling Language (UML)*

Pada buku yang ditulis oleh (Sundaramoorthy S, 2022) yang berjudul “*UML Diagramming: A Case Study Approach*” menjelaskan tentang *Unified Modeling Language (UML)* adalah bahasa grafis yang digunakan untuk merancang sistem perangkat lunak, membantu dalam menentukan, memvisualisasikan, membangun, dan mendokumentasikan sistem. UML mencakup beberapa diagram utama seperti *Use Case, Activity, Class, Sequence, Collaboration (Communication), State Machine, Component, dan Deployment Diagram*. Untuk mengetahui lebih detail silahkan lihat Tabel 2.1 dibawah ini.



Tabel 2.1 *List of UML Diagrams* (Sundaramoorthy S, 2022)

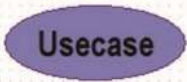
No	Diagram UML	Tujuan dari diagram UML
1	Diagram <i>Use Case</i>	Diagram <i>Use Case</i> ini bertujuan untuk menemukan kebutuhan fungsional sistem.
2	<i>Activity Diagram</i>	<i>Activity Diagram</i> berfokus pada aktivitas berurutan dan paralel yang terlibat dalam setiap kebutuhan fungsional sistem.
3	<i>Class Diagram</i>	<i>Class Diagram</i> berfungsi untuk menggambarkan struktur sistem dalam hal kelas dan objek.
4	<i>Sequence Diagram</i>	<i>Sequence Diagram</i> berfungsi untuk menggambarkan objek yang terlibat dalam skenario dan urutan pesan

		yang pertukarkan antara objek yang diperlukan untuk menjalankan fungsionalitas tersebut
5	<i>Collaboration</i> Diagram (<i>Communication</i> Diagram)	<i>Collaboration</i> Diagram (<i>Communication</i> Diagram) bertujuan untuk menampilkan interaksi antara objek menggunakan pesan yang disusun secara berurutan dalam susunan bebas.
6	<i>State Machine</i> Diagram	<i>State Machine</i> Diagram menggambarkan siklus hidup sebuah objek menggunakan tiga elemen utama: keadaan objek, transisi antar keadaan, dan peristiwa yang memicu transisi tersebut.
7	<i>Component</i> Diagram	<i>Component</i> Diagram berfungsi untuk menunjukkan hubungan antara berbagai komponen dalam suatu sistem.
8	<i>Deployment</i> Diagram	<i>Deployment</i> Diagram digunakan untuk menggambarkan komponen perangkat keras tempat komponen perangkat lunak ditempatkan. Tujuan utama diagram ini dapat dijelaskan sebagai berikut: <ol style="list-style-type: none"> 1. Memvisualisasikan topologi perangkat keras suatu sistem. 2. Menggambarkan komponen perangkat keras yang digunakan untuk menerapkan komponen perangkat lunak. 3. Menggambarkan node pemrosesan runtime.

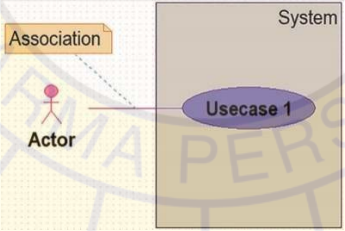
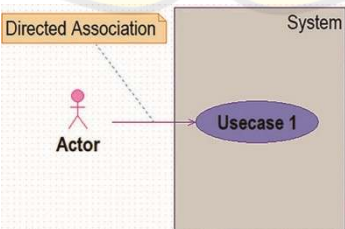
2.1.10.2 Use Case Diagram

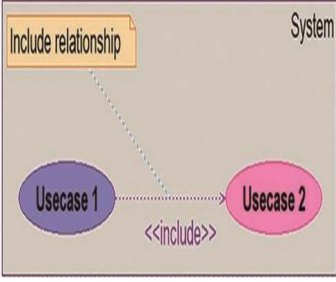
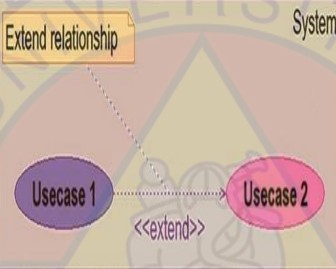
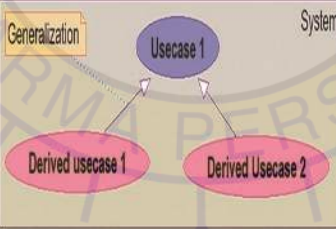
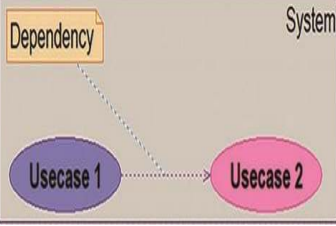
Tabel 2.2 Komponen *Use Case Diagram* (Sundaramoorthy S, 2022)

No	Nama Komponen	Notasi UML	Fungsi
1	Batas Sistem (<i>System Boundary</i>)		<ol style="list-style-type: none"> 1. Mendefinisikan ruang lingkup sistem. 2. Mencakup seluruh rangkaian fungsi yang ada di dalam sistem.
2	Aktor (<i>Actors</i>)		<ol style="list-style-type: none"> 1. Pengguna yang berinteraksi dengan suatu sistem. 2. Aktor dapat berupa orang, organisasi, atau sistem eksternal yang berinteraksi dengan aplikasi atau sistem. 3. Dalam diagram <i>use case</i>, aktor berinteraksi dengan <i>use case</i> (kasus penggunaan). 4. Terserah kepada <i>developer</i> untuk mempertimbangkan aktor mana yang berdampak pada fungsionalitas yang ingin mereka modelkan.

3	<i>Use Case</i>		<ol style="list-style-type: none"> 1. Representasi visual dari fungsionalitas bisnis tertentu dalam suatu sistem. 2. Memastikan proses bisnis bersifat diskrit. 3. Mencantumkan fungsi-fungsi bisnis yang terpisah dalam pernyataan masalah yang diberikan. 4. Mengidentifikasi <i>use case</i> adalah sebuah proses penemuan.
---	-----------------	---	--



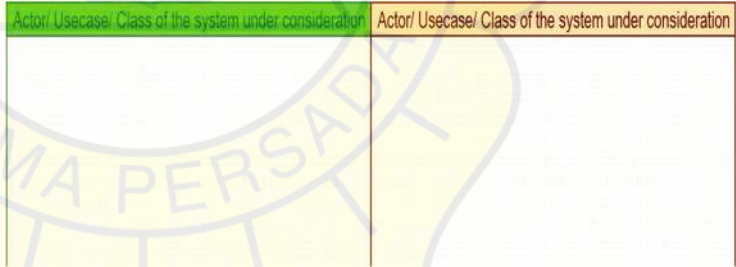
Tabel 2.3 *Use Case Relationships* (Sundaramoorthy S, 2022)



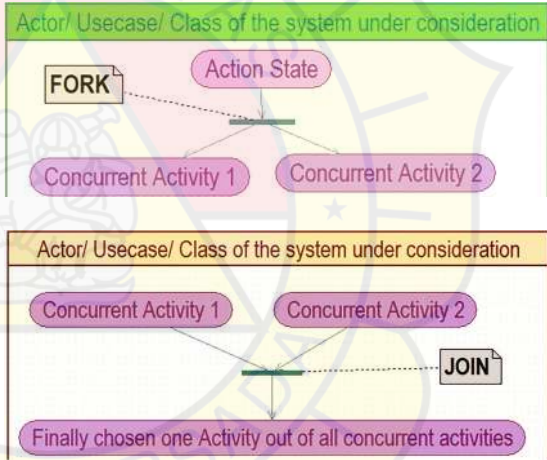
No	<i>Use Case Relationship</i>	Notasi UML	Fungsi
1	Asosiasi		Menunjukkan hubungan antara aktor dan <i>use case</i> .
2	Asosiasi Terarah		Menunjukkan hubungan satu arah dimana aktor mempengaruhi <i>use case</i> .

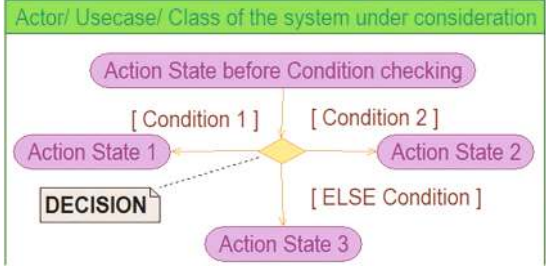
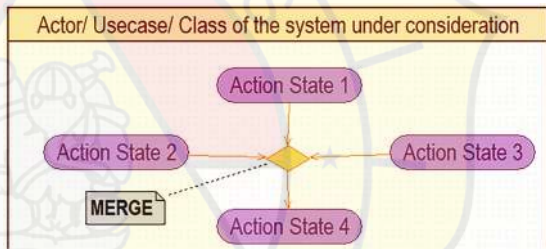

3	<i>Include</i>		Menunjukkan situasi dimana satu <i>use case</i> (<i>base usecase</i>) mencakup fungsionalitas dari <i>use case</i> lain (<i>inclusion usecase</i>).
4	<i>Extend</i>		Menunjukkan hubungan antara dua <i>use case</i> dimana <i>use case</i> yang diperluas (<i>extended use case</i>) menambahkan perilaku tambahan pada fungsionalitas yang ada dari <i>use case</i> dasar (<i>base use case</i>).
5	Generalisasi		Merupakan hubungan antara <i>use case</i> induk (<i>parent</i>) dan satu atau lebih <i>use case</i> anak (<i>child</i>).
6	Dependensi		Mendefinisikan hubungan di mana keberadaan satu <i>use case</i> bergantung pada keberadaan <i>use case</i> lainnya.

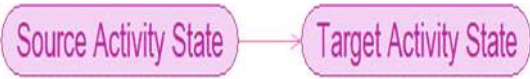



2.1.10.3 Activity Diagram

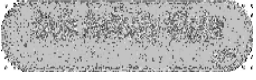
Tabel 2.4 Komponen *Activity Diagram* (Sundaramoorthy S, 2022)

No	Nama Komponen	Notasi UML dan Fungsinya
1	<i>Initial state</i>	 <p>Merupakan keadaan awal dari sistem yang sedang di kembangkan.</p>
2	<i>Final state</i>	 <p>Merupakan keadaan akhir dari sistem yang sedang di kembangkan.</p>
3	<i>Swimlanes</i>	 <p><i>Swimlane</i> adalah sistem yang membagi sistem menjadi dua partisi:</p> <ol style="list-style-type: none"> 1. mewakili entitas seperti aktor atau <i>use case</i>. 2. berfokus pada rangkaian aktivitas yang terlibat.

		<p><i>Swimlane</i> dapat berbentuk vertikal atau horizontal, dengan <i>swimlane</i> vertikal mewakili aktivitas paralel dan <i>swimlane</i> horizontal mewakili aktivitas berurutan.</p>
4	<i>Action state</i>	<div style="text-align: center;">  </div> <p><i>action state</i> mewakili operasi atau aktivitas bisnis atau proses.</p>
5	<i>Object</i>	<div style="text-align: center;">  </div> <p>Entitas yang membawa data di antara dua <i>action state</i>.</p>
6	<i>Synchronization</i>	<div style="text-align: center;">  </div> <p>Sinkronisasi adalah proses menjalankan dua atau lebih aktivitas pada waktu yang sama. Proses ini dapat dipisahkan menjadi dua atau lebih aktivitas bersamaan menggunakan pendekatan <i>Fork</i>, atau dapat digabungkan ke dalam satu <i>flow</i> menggunakan metode <i>Join</i>, untuk memastikan bahwa hanya satu aktivitas yang berlangsung pada satu waktu.</p>

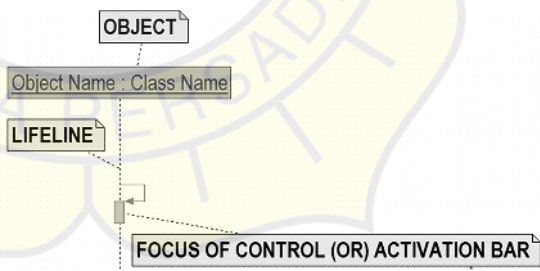
7	<i>Decision</i>	 <p>The diagram shows a decision node (yellow diamond) with an incoming flow from 'Action State before Condition checking'. Three outgoing flows are labeled with conditions: '[Condition 1]' leading to 'Action State 1', '[Condition 2]' leading to 'Action State 2', and '[ELSE Condition]' leading to 'Action State 3'. A callout box labeled 'DECISION' points to the diamond.</p> <p>Sebuah <i>decision node</i> memiliki satu <i>input</i> atau beberapa <i>output</i> berdasarkan kondisi yang dirancang. Setiap <i>output flow</i> memiliki kondisi yang melekat padanya, dan jika terpenuhi, <i>flow</i> tersebut akan menghasilkan <i>output</i> yang sesuai. <i>Output</i> 'lain' dapat didefinisikan jika tidak ada kondisi lain yang terpenuhi.</p>
8	<i>Merge</i>	 <p>The diagram shows a merge node (yellow diamond) with three incoming flows from 'Action State 1', 'Action State 2', and 'Action State 3'. One outgoing flow leads to 'Action State 4'. A callout box labeled 'MERGE' points to the diamond.</p> <p><i>Merge</i> memiliki tujuan untuk menggabungkan <i>input flow</i> tetapi tidak disinkronkan, sehingga memungkinkan sebuah <i>flow</i> untuk melanjutkan <i>output</i> tanpa menunggu <i>flow</i> lainnya.</p>
9	<i>Flow Final</i>	 <p>The symbol is a red circle with a diagonal cross through it, representing a flow final node.</p> <p><i>Flow final</i> merupakan penghentian jalur yang tidak normal dalam diagram aktivitas yang tidak dianggap sebagai bagian dari sistem yang sedang dikembangkan.</p>


10	<i>Transition</i>	 <p>Transisi adalah anak panah yang merepresentasikan pergerakan dari status aktivitas sumber ke status aktivitas target yang dipicu oleh selesainya aktivitas pada status aktivitas sumber.</p>
11	<i>Self-Transition</i>	 <p><i>Self-Transition</i> mewakili transisi internal ke status tindakan itu sendiri.</p>
12	<i>Signal Send State</i>	 <p><i>Signal</i> adalah pengaruh eksternal pada aktivitas, biasanya berupa pasangan sinyal yang dikirim dan diterima. <i>Signal Send State</i> adalah tindakan di luar aktivitas, yang tidak menunggu respons dari penerima dan berakhir dengan sendirinya, menyerahkan kontrol eksekusi ke tindakan berikutnya. Hal ini dinotasikan sebagai segi lima cembung.</p>
13	<i>Signal Accept State</i>	 <p><i>Signal Accept State</i> adalah status aksi yang memicu peristiwa sinyal dan berhubungan dengan <i>Signal Send</i></p>

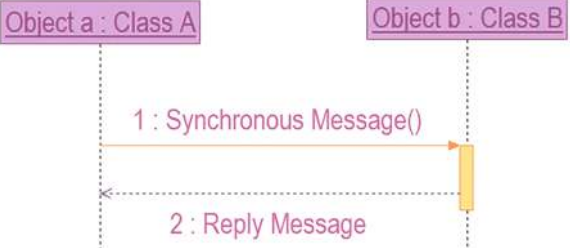
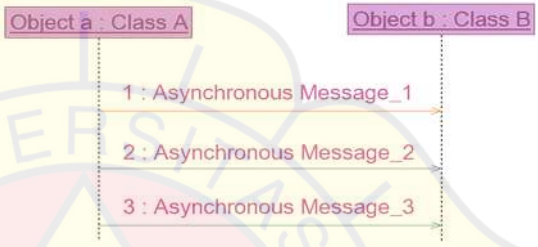
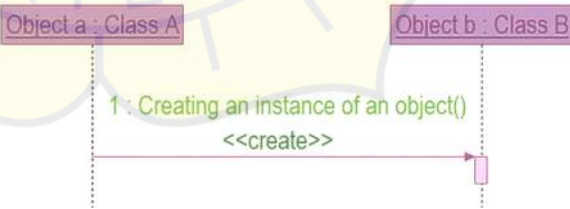
		<p><i>State</i>. Status ini dimulai setelah status aksi sebelumnya selesai, sementara <i>Status Signal Accept</i> tanpa sisi masuk tetap diaktifkan setelah menerima peristiwa. Status ini tidak berakhir setelah menerima sebuah peristiwa, tetapi terus menunggu peristiwa lainnya.</p>
14	<i>Subactivity</i>	 <p><i>Subactivity</i> adalah status tindakan yang dapat menjadi skenario aktivitas utama lainnya di masa mendatang.</p>

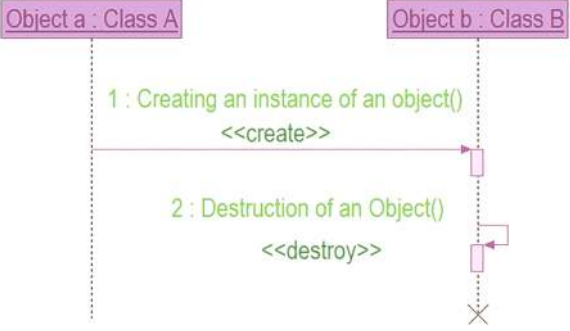
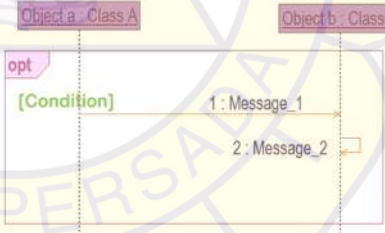
2.1.10.4 Sequence Diagram

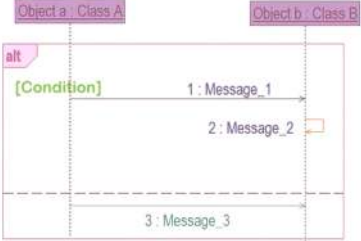

Tabel 2.5 Komponen *Sequence Diagram* (Sundaramoorthy S, 2022)

No	Nama Komponen	Notasi UML dan Fungsinya
1	<i>Object</i>	 <p>Objek adalah entitas sistem yang berinteraksi dengan entitas lain melalui pesan. Objek memiliki tiga informasi terkait: nama objek, <i>lifeline</i>, dan fokus kontrol.</p> <ol style="list-style-type: none"> Sintaks standar untuk menamai sebuah objek adalah <code>Object_Name: Class_Name</code>.

		<p>2. <i>Lifeline</i> merepresentasikan keberadaan objek secara lengkap,</p> <p>3. fokus kontrol merepresentasikan sesi aktifnya.</p>
2	<i>Message & its types</i>	<p>Komunikasi antar objek dalam skenario melibatkan pengoperan pesan menggunakan komponen transisi. Pesan diberi nomor menggunakan panah transisi dan memiliki sintaks Message_no: Message (). Dalam <i>sequence diagram</i>, ada lima jenis pesan yang dapat diterapkan pada setiap transisi antara dua objek.</p>
2.1	<i>Synchronous Message</i>	 <p>Pesan pengirim menggunakan semantik tunggu, yang mengharuskan penerima untuk mengakui setiap pesan sebelum menyiarkan pesan berikutnya. <i>Synchronous Message</i> direpresentasikan sebagai garis solid dengan kepala panah yang digelapkan, dan pesan tersebut harus diakui sebelum pesan berikutnya dapat dikirim.</p>

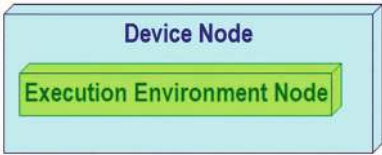
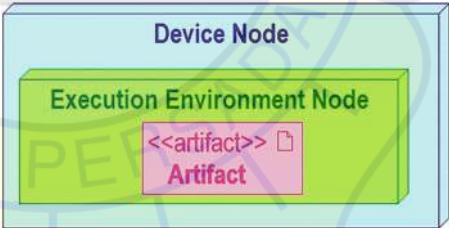
2.2	<p><i>Return Message</i></p>	 <p>The diagram shows two lifelines: Object a : Class A and Object b : Class B. Object a sends a synchronous message to Object b, labeled '1 : Synchronous Message()'. Object b then sends a return message back to Object a, labeled '2 : Reply Message'.</p> <p><i>Return Message</i> merupakan pesan balasan untuk <i>Synchronous Message</i>.</p>
2.3	<p><i>Asynchronous Message</i></p>	 <p>The diagram shows two lifelines: Object a : Class A and Object b : Class B. Object a sends three asynchronous messages to Object b, labeled '1 : Asynchronous Message_1', '2 : Asynchronous Message_2', and '3 : Asynchronous Message_3'.</p> <p><i>Asynchronous Message</i> dikirim oleh pengirim ke penerima tanpa memerlukan pengakuan atau tanggapan langsung dari penerima. Ini memungkinkan pengirim untuk mengirim sejumlah pesan ke penerima tanpa menunggu konfirmasi.</p>
2.4	<p><i>Create Message</i></p>	 <p>The diagram shows two lifelines: Object a : Class A and Object b : Class B. Object a sends a create message to Object b, labeled '1 : Creating an instance of an object()' with the stereotype '<<create>>'.</p> <p><i>Create Message</i> digunakan untuk menggambarkan pembuatan (<i>instansiasi</i>) objek baru dalam suatu skenario. Pesan ini ditandai dengan <i>stereotype</i> "<i><<create>></i>".</p>

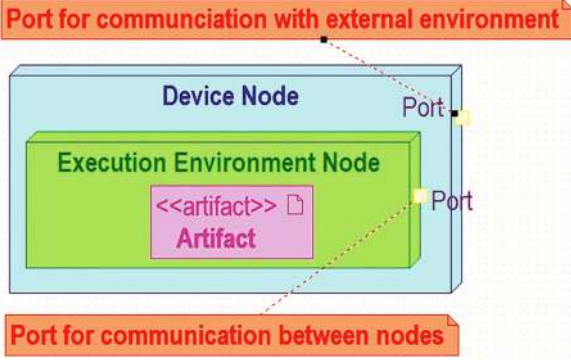
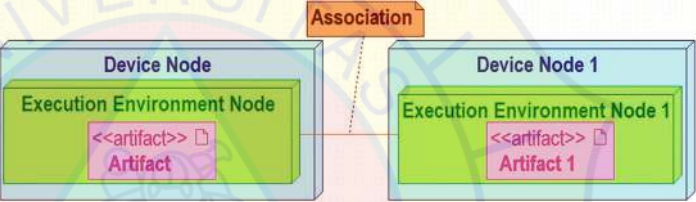
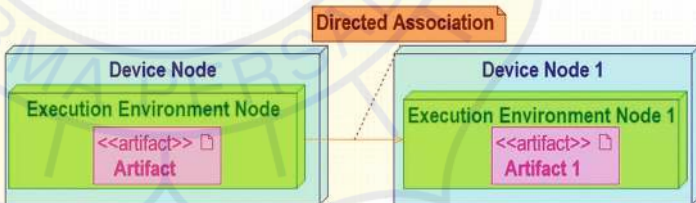
2.5	<p><i>Destroy Message</i></p>	 <p><i>Destroy Message</i> digunakan untuk menggambarkan penghentian siklus hidup suatu objek dalam suatu skenario. Pesan ini ditandai dengan <i>stereotype</i> "<i><<destroy>></i>".</p>
3	<p><i>Fragments</i></p>	<p><i>Fragments</i> n mewakili sekumpulan pesan bersyarat atau pesan berulang dalam suatu skenario. <i>Fragments</i> yang paling penting adalah <i>opt</i>, <i>alt</i>, dan <i>loop</i>.</p>
3.1	<p><i>Opt</i></p>	 <p><i>Fragments opt</i> adalah skenario <i>IF</i> sederhana di mana pesan dieksekusi hanya jika kondisi yang didefinisikan dalam <i>fragments opt</i> tetap benar, dan jika tidak, kontrol akan keluar dari <i>fragments</i>, yang secara teknis dikenal sebagai <i>Guard</i>.</p>

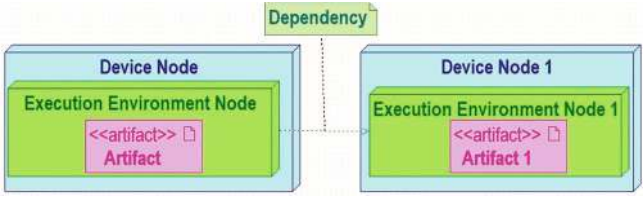
3.2	<i>Alt</i>	 <p>The diagram shows two objects, Object a: Class A and Object b: Class B. An 'alt' fragment is shown with a green header. It is divided into two vertical partitions. The top partition is labeled '[Condition]' and contains two messages: '1: Message_1' (a solid arrow from A to B) and '2: Message_2' (a dashed arrow from A to B). The bottom partition contains '3: Message_3' (a solid arrow from A to B). A dashed horizontal line separates the two partitions.</p> <p><i>Fragments alt</i> adalah skenario <i>IF ELSE</i> dengan dua partisi dalam area persegi panjang. Pesan di kompartemen pertama dieksekusi jika kondisi di partisi pertama tetap benar, sementara pesan di partisi kedua dieksekusi sebaliknya.</p>
3.3	<i>Loop</i>	 <p>The diagram shows two objects, Object a: Class A and Object b: Class B. A 'loop' fragment is shown with a green header. It contains three messages: '1: Message_1' (a solid arrow from A to B), '2: Message_2' (a dashed arrow from A to B), and '3: Message_3' (a solid arrow from A to B). A green box labeled '[condition]' is positioned above the messages, with a red arrow pointing to the start of the loop.</p> <p><i>Fragments loop</i> adalah sekumpulan pesan yang dieksekusi berulang kali sampai kondisi dalam <i>fragments</i> perulangan tetap benar. <i>Fragments</i> ini memiliki dua partisi dalam area persegi panjang, dengan pesan di kompartemen pertama hanya dieksekusi jika kondisi di partisi pertama tetap benar.</p>
4	<i>Nesting of fragments</i>	<p>Konsep <i>nesting</i> dapat diterapkan pada berbagai jenis <i>fragments</i> tergantung pada kebutuhan beberapa kondisi untuk verifikasi atau eksekusi berulang.</p>

2.1.10.5 Deployment Diagram

Tabel 2.6 Komponen *Deployment Diagram* (Sundaramoorthy S, 2022)

No	Nama Komponen	Notasi UML dan Fungsinya
1	Node	 <p><i>Node</i> mengacu pada sumber daya komputasi apa pun yang memiliki memori dan kemampuan pemrosesan, dan ada dua jenis: <i>Node</i> perangkat, yang mewakili perangkat keras seperti server aplikasi dan <i>workstation</i>, dan <i>node</i> lingkungan eksekusi, yang mewakili perangkat lunak atau program seperti sistem operasi, <i>workflow engine</i>, <i>browser</i>, server web, dan sistem basis data.</p>
2	<i>Artifact</i>	 <p><i>Artifact</i> terdiri dari beberapa file skrip seperti PHP, <i>database files</i>, <i>configuration files</i>, <i>rar files</i>, <i>executable files</i></p>

3	<i>Port</i>	 <p>The diagram shows a light blue box labeled "Device Node" containing a green box labeled "Execution Environment Node". Inside the Execution Environment Node is a pink box labeled "<<artifact>> Artifact". Two yellow circles labeled "Port" are shown on the right side of the Device Node. A red callout box at the top points to the upper port with the text "Port for communication with external environment". A red callout box at the bottom points to the lower port with the text "Port for communication between nodes".</p> <p><i>Port</i> berfungsi untuk memfasilitasi komunikasi antara <i>node</i> perangkat dan <i>node</i> lingkungan eksekusi, serta antara <i>node</i> dan lingkungan eksternal.</p>
4	<i>Association</i>	 <p>The diagram shows two light blue boxes. The left one is labeled "Device Node" and contains a green "Execution Environment Node" with a pink "<<artifact>> Artifact". The right one is labeled "Device Node 1" and contains a green "Execution Environment Node 1" with a pink "<<artifact>> Artifact 1". A red line connects the Execution Environment Node of the first device to the Execution Environment Node of the second device. A red callout box labeled "Association" points to this line.</p> <p><i>Association</i> mengacu pada komunikasi dua arah antara dua <i>node</i> yang terlibat dalam suatu hubungan.</p>
5	<i>Directed Association</i>	 <p>The diagram shows two light blue boxes. The left one is labeled "Device Node" and contains a green "Execution Environment Node" with a pink "<<artifact>> Artifact". The right one is labeled "Device Node 1" and contains a green "Execution Environment Node 1" with a pink "<<artifact>> Artifact 1". A red arrow points from the Execution Environment Node of the first device to the Execution Environment Node of the second device. A red callout box labeled "Directed Association" points to this arrow.</p> <p><i>Directed Association</i> mengacu pada komunikasi satu arah antara dua <i>node</i> yang terlibat dalam suatu hubungan.</p>

6	Dependency	 <p><i>Dependency</i> mengacu pada hubungan antara satu <i>node</i> dan <i>node</i> lain dalam suatu hubungan dengan menekankan bagaimana keberadaan suatu <i>node</i> bergantung pada keberadaan yang lain.</p>
---	------------	--

2.2 Kajian Penelitian Terdahulu

2.2.1 Paper 1

Pada penelitian yang berjudul “*PENGEMBANGAN APLIKASI BAHASA ISYARAT INDONESIA BERBASIS REALTIME VIDEO MENGGUNAKAN MODEL MACHINE LEARNING*” (Ambarak A & Falani A, 2023), Penelitian ini bertujuan untuk mengembangkan aplikasi yang memungkinkan pengenalan Bahasa Isyarat Indonesia (BISINDO) secara *real-time* melalui video menggunakan model *machine learning*.

Dalam penelitian ini, arsitektur *pre-trained* model yang digunakan adalah *EfficientDet-lite0*. Model ini memanfaatkan *pre-trained* model pada platform *machine learning TensorFlow Lite* dan diimplementasikan dalam bahasa pemrograman Java untuk aplikasi Android. Model AI yang dirancang memiliki empat kelas dan telah berhasil mendeteksi isyarat bahasa menggunakan aplikasi yang dibangun. Nilai *mean average precision* yang optimal sebesar 77% menunjukkan performa yang baik.

Namun, seperti setiap penelitian, ada beberapa kelemahan yang perlu diperhatikan:

1. Ukuran Dataset: Penelitian ini tidak menyebutkan secara rinci tentang ukuran dataset yang digunakan untuk melatih model. Dataset yang terbatas dapat mempengaruhi performa model dan generalisasi ke situasi yang lebih kompleks.
2. Keterbatasan Data: Model AI yang dirancang hanya memiliki empat label kelas. Ini mungkin tidak mencakup semua isyarat bahasa yang digunakan dalam komunikasi sehari-hari oleh kelompok tuli.

Evaluasi Lebih Lanjut: Nilai *mean average precision* sebesar 77% menunjukkan performa yang baik, tetapi evaluasi lebih lanjut diperlukan. Misalnya, bagaimana model berperilaku dalam kondisi pencahayaan yang berbeda atau dengan variasi gerakan tangan yang lebih kompleks.

2.2.2 Paper 2

Dalam penelitian yang berjudul "*PENGEMBANGAN APLIKASI PENGENALAN BAHASA ISYARAT ABJAD SIBI MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN)*" (Sholawati M dkk., 2022), Penelitian ini bertujuan mengklasifikasikan peragaan bahasa isyarat abjad SIBI (Sistem Bahasa Isyarat Indonesia) berdasarkan peragaan langsung oleh guru maupun murid di depan kamera secara *real-time*. Aplikasi ini dibuat berbasis web dengan proses klasifikasi menggunakan metode *Convolutional Neural Network* (CNN). Peneliti menggunakan 416 citra digital peragaan abjad SIBI untuk melatih model. Model yang telah dirancang tersebut telah berhasil mendeteksi abjad SIBI dengan hasil nilai akurasi sebesar 80.76%.

Namun masih terdapat beberapa kelemahan yang perlu di perhatikan:

1. Kesalahan Deteksi: Terdapat kesalahan dalam pengenalan bahasa isyarat abjad B, F, M, N, dan S. Hal ini disebabkan karena bentuk peragaan bahasa isyarat abjad SIBI memiliki kemiripan yang membuat sistem mengalami kesulitan dalam membedakannya secara akurat.
2. Ketidakmampuan Mengenali Gerakan Aktif: Peragaan bahasa isyarat abjad J dan Z belum berhasil dikenali oleh sistem secara sempurna. Ini disebabkan karena peragaan abjad J dan Z melibatkan gerakan aktif yang lebih kompleks dan sulit untuk diidentifikasi.

Kurangnya Informasi tentang Model: Peneliti tidak menyebutkan secara rinci model CNN yang digunakan dalam penelitian ini. Sehingga arsitektur model tidak bisa dijadikan parameter yang digunakan akan membantu memahami keandalan dan performa aplikasi yang dikembangkan.

2.2.3 Paper 3

Dalam penelitian yang berjudul “*Implementasi Deep Learning pada Pengenalan Angka dalam Sign Language*” (Celsia F & Sandag G, 2021), Penelitian ini menggunakan metode *Deep Learning*, khususnya mengevaluasi empat model algoritma seperti *Convolutional Neural Network*, *Artificial Neural Network*, *Logistic Regression*, dan *L Layer Neural Network* untuk mengklasifikasikan bahasa isyarat berdasarkan angka. Penelitian ini menggunakan dataset *sign language digits*, yang terdiri dari gambar-gambar yang mewakili setiap digit dalam bahasa isyarat. Totalnya ada 2.062 gambar yang mencakup angka 0 hingga 9. Hasilnya menunjukkan bahwa algoritma *Artificial Neural Network* dan *Convolutional Neural Network* memiliki performa terbaik dengan akurasi sebesar 99,7%. Kedua

algoritma ini lebih baik dalam memprediksi bahasa isyarat dibandingkan dengan algoritma lainnya, dengan perbedaan sekitar 1% hingga 6,9%.

Meskipun penelitian ini berhasil mengklasifikasikan angka dalam bahasa isyarat, terdapat beberapa kelemahan yang perlu diperhatikan:

1. Data Abjad: Penelitian hanya menggunakan angka 0 hingga 9 dalam American Sign Language. Jika ingin mengaplikasikan model untuk mendeteksi abjad ataupun kata, perlu dilakukan adaptasi lebih lanjut.

Kurangnya Penjelasan Model dan *Output*: Sayangnya, penelitian ini tidak memberikan penjelasan mendalam tentang model yang digunakan dan *output* yang dihasilkan

