BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Penelitian yang menerapkan *deep learning* untuk klasifikasi dan deteksi pada objek gambar akhir-akhir ini banyak dilakukan dan dimanfaatkan di berbagai bidang teknologi untuk menyelesaikan berbagai permasalahan yang ada. Berdasarkan penelitian terdahulu menunjukkan bahwa *deep learning* merupakan suatu perkembangan teknologi yang dalam penerapannya menggunakan konsep meniru otak manusia untuk menyelesaikan sebuah permasalahan dengan kemampuan tersebut dapat meningkatkan akurasi dalam pengenalan gambar pada sebuah aplikasi deteksi dan klasifikasi objek gambar. Berikut ini merupakan penjelasan dari hasil tinjauan dari penelitian sebelumnya.

Penelitian pertama tentang Penerapan *Convolutional Neural Network* dalam Pengenalan Citra Kebakaran Hutan yang dilakukan oleh [10] pada tahun 2019. Penelitian tersebut memiliki tujuan untuk mendeteksi kebakaran hutan pada suatu tempat menggunakan citra yang sudah dilakukan segmentasi pada area api. Metode yang dipakai pada penelitian tersebut menggunakan CNN yang merupakan model *deep learning* biasanya dipakai dalam klasifikasi objek gambar [14]. Data yang didapatkan berasal dari *corcisa fire database* yang dibuat oleh *University of Corsica*. Hasil pelatihan dan pengujian untuk proses klasifikasi gambar kebakaran dan tidak kebakaran dengan menerapkan metode *Convolutional Neural Network* (CNN) didapatkan akurasi 90,7% yang berasal dari skema percobaan dengan menggunakan data segmentasi area api pada sampel data latih dan uji.

Penelitian kedua tentang klasifikasi dan deteksi kebakaran hutan menggunakan *deep learning* yang dilakukan oleh [15] pada tahun 2019. Penelitian tersebut memiliki tujuan untuk melakukan deteksi dini kebakaran hutan agar dapat menghindari kerusakan dan bencana kebakaran hutan yang semakin besar [16] dengan menggunakan data citra satelit [9]. Metode yang digunakan teknik *deep learning* dengan model pembelajaran *transfer learning Inception v3* yang berbasis

Convolutional Neural Network (CNN). *Dataset* yang digunakan bersumber dari EOSDIS dan MODIS. Jumlah citra satelit yang digunakan sebanyak 534 data mencakup 239 data kelas kebakaran hutan dan 295 data kelas tidak kebakaran hutan. 481 data citra untuk proses *training* dan 53 data citra untuk proses *testing*. Hasil dari pengujian dari penelitian tersebut menggunakan citra satelit untuk diproses deteksi kebakaran hutan atau tidak menggunakan metode *Inception v3* yang berbasis *Convolutional Neural Network* (CNN) diperoleh akurasi 98%.

Penelitian ketiga tentang deteksi masker dan tidak menggunakan masker menggunakan *DenseNet201* yang dilakukan oleh [12] pada tahun 2021. Penelitian tersebut memiliki tujuan untuk mendeteksi penggunaan masker pada wajah untuk mencegah penularan *COVID-19* dan mendukung kebijakan 3M dari pemerintah Indonesia salah satunya memakai masker [17]. Pada penelitian tersebut menggunakan teknik *deep learning* dengan menggunakan model pembelajaran *transfer learning DenseNet201* dan *MobileNetV2* sebagai perbandingan. Data yang digunakan pada penelitian tersebut bersumber dari *Masked Face Recognition Dataset* (MFRD) dengan jumlah data yang digunakan untuk penelitian tersebut terdiri dari 90.000 gambar wajah tanpa menggunakan masker dan 2.203 gambar wajah menggunakan masker dari 525 orang. Dari hasil percobaan model *DenseNet201* menghasilkan nilai *F-Measure* 98% dan model *MobileNetV2* menghasilkan nilai *F-Measure* 67% dengan hasil tersebut model *DenseNet201* memiliki akurasi yang paling tinggi dan akurat dalam melakukan deteksi penggunaan masker.

Penelitian keempat tentang mengklasifikasi pasien yang terinfeksi *COVID-19* menggunakan *DenseNet201* yang dilakukan oleh [18] pada tahun 2020. Penelitian tersebut memiliki tujuan untuk melakukan klasifikasi diagnosa virus *COVID-19* menggunakan citra *CT Scan* pada pasien yang terkena virus *COVID-19* [19]. Penelitian tersebut memiliki kendala yaitu jumlah data citra *CT Scan* pasien yang terkena virus *COVID-19* sedikit sehingga percobaan tersebut menggunakan model pembelajaran *transfer learning DenseNet201*. Data yang dipakai bersumber dari *dataset* dari situs *kaggle SARS-COV-2 Ct-Scan Dataset* [20] dengan jumlah total 2492 data gambar *CT Scan* yang terdiri dari 1262 data gambar *CT Scan* positif

virus *COVID-19* dan sisanya 1230 data gambar yang negatif terinfeksi virus *COVID-19*. Hasil yang diperoleh bahwa metode *DenseNet201* menghasilkan akurasi pelatihan 99,82% dan pengujian 96,25% dengan hasil tersebut model *DenseNet201* memiliki akurasi yang tinggi dalam melakukan klasifikasi citra *CT Scan* yang terkena virus *COVID-19* atau tidak.

Penelitian kelima tentang pembuatan sistem deteksi huruf *rusia* menggunakan *framework flask* yang dilakukan oleh [21] pada tahun 2021. Penelitian tersebut memiliki tujuan untuk melakukan deteksi objek tulisan tangan dengan huruf Rusia untuk melakukan koreksi otomatis yang bermanfaat di bidang pendidikan bahasa asing yang di mana seseorang dapat belajar menulis huruf Rusia. Metode dipakai dalam proses pengembangan *website* ini menerapkan metode *waterfall* [22] kemudian untuk pembuatan *website* menggunakan *framework flask* dalam menangani setiap *request* model jaringan syaraf tiruan [23]. Data yang dipakai bersumber dari situs *kaggle* terdiri dari data objek gambar tulisan tangan huruf Rusia 403 data huruf untuk proses *training* dan 107 data huruf untuk *testing*. Hasil yang didapat pada penelitian ini adalah *website* dengan menggunakan *framework flask* yang dapat digunakan untuk menganalisa tulisan huruf Rusia yang dapat melakukan deteksi berbagai macam jenis huruf Rusia dari huruf *ah* sampai *zhe*.

Penelitian terakhir tentang pengimplementasian sistem deteksi Indonesia *Sing Language* yang menggunakan *framework flask* yang dilakukan oleh [24] pada tahun 2021. Penelitian tersebut memiliki tujuan untuk membatu masyarakat yang belum paham dan terbiasa dengan bahasa isyarat yang digunakan untuk berkomunikasi untuk penyandang tuna rungu maka diperlukan sebuah sistem deteksi Indonesa *Sign Language* yang berbasis *web* yang dapat digunakan untuk membatu masyarakat yang belum memahami dan terbiasa dengan bahasa isyarat. Penelitian tersebut menggunakan *framework flask* dalam proses implementasi model jaringan syaraf tiruan klasifikasi citra ke dalam bentuk *website* sehingga fungsionalitasnya dapat digunakan sepeti mengakses fitur-fitur pada sistem tersebut. Pada penelitian ini, digunakan 2080 data yang dibagi menjadi 26 kelas yang diawali dengan huruf A-Z. Pembagian data dilakukan menggunakan aturan *pareto*, dengan 80% data

digunakan sebagai data latih dan 20% sebagai data uji. Data ini diambil menggunakan kamera. Hasil yang diperoleh dari penelitian ini adalah *website* yang menggunakan *framework flask* yang dapat melakukan deteksi dalam bahasa isyarat Indonesia dari huruf A-Z. Perbandingan beberapa penelitian berdasarkan kajian pustaka diuraikan pada tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No.	Peneliti, Tahun	Judul	Metode	Hasil
1.	Y. Wang, L. Dang, and J. Ren. Tahun 2019	Pengenalan Citra Kebakaran Hutan Berdasarkan *Convolutional Neural Network*	*Convolutional Neural Network*	Hasil dari penelitian tersebut dengan menggunakan metode *Convolutional Neural Network* (CNN) memiliki akurasi 90.7% yang didpatkan dari hasil skema percobaan menggunakan sampel pelatihan dan percobaan segmentasi area api.
2.	R. Shanmuga Priya dan K. Vani. Tahun 2019	Klasifikasi dan Deteksi Kebakaran Hutan Berbasis *Deep Learning* menggunakan Citra Satelit	*Convolutional Neural Network Berbasis InceptionV3*	Hasil dari penelitian tersebut dengan menggunakan menggunakan *InceptionV3* yang berbasis *Convolutional Neural Network* (CNN) dengan menggunakan citra satelit dari hasil percobaan diperoleh nilai *F1-Score* yaitu 98%.
3.	Faisal Dharma Adhinata, Diovianto Putra Rakhmadani, Merlinda Wibowo, dan	*Deep Learning* Menggunakan *DenseNet201* Untuk Mendeteksi Masker	*DenseNet201*	Hasil dari penelitian tersebut dengan menggunakan *DenseNet201* diperoleh model hasil pelatihan yang memiliki akurasi 99%

No.	Peneliti, Tahun	Judul	Metode	Hasil
	Akhmad Jayadi. Tahun 2021	dan Tidak Menggunakan Masker		dan hasil percobaan yang dilakukan diperoleh nilai *F1-Score* yaitu 98%.
4.	Aayush Jaiswala, Neha Gianchandania, Dilbag Singha, Vijay Kumarb dan Manjit Kaurc. Tahun 2021	Klasifikasi Pasien Terinfeksi COVID-19 Menggunakan *Deep Transfer Learning* Berbasis *DenseNet201*	*DenseNet201*	Hasil dari penelitian tersebut dengan menggunakan *DenseNet201* diperoleh model hasil pelatihan yang memiliki akurasi 99,82% dan hasil percobaan diperoleh hasil akurasi 96,25%.
5.	Kholilul Rachman N.M, Eka Prakarsa Mandyartha, dan Agung Mustika Rizki. Tahun 2021	Rancang Bangun Sistem Deteksi Huruf Rusia Berbasis *Web Flask*	*Framework Flask*	Hasil dari penelitian tersebut yaitu sebuah aplikasi berbasis *website* dengan menggunakan *framework flask* yang dapat mendeteksi dan mengklasifikasikan macam-macam huruf *rusia*
6.	Mohammad Idham Fachrurrozil, Yisti Vita Via, dan Wahyu S.J Saputra. Tahun 2021	Implementasi Sistem Pendeteksi *Indonesia Sign Language Bisindo* Berbasis *Web Flask*	*Framework Flask*	Hasil dari penelitian tersebut yaitu sebuah aplikasi berbasis *website* dengan menggunakan *framework flask* yang dapat mendeteksi dan mengklasifikasikan Indonesia *Sign Language Bidiondo*.

Berdasarkan tabel 2.1 dapat ditarik kesimpulan bahwa penelitian mengenai implementasi *website* deteksi kebakaran hutan dan lahan menggunakan *DenseNet201* belum pernah dilakukan sebelumnya. Dalam penelitian ini, penelitian nomor 1 dan 2 digunakan sebagai acuan karena memiliki tujuan yang sama. Namun, perbedaan yang ada pada kedua penelitian tersebut terletak pada *dataset* yang digunakan, algoritma yang digunakan, dan tidak diimplementasikan ke dalam bentuk *website*. Guna melengkapi literatur penelitian ini, penelitian nomor 3 dan 4 memiliki kemiripan yaitu menerapkan model *DenseNet201* dan penelitian nomor 5 dan 6 memiliki kemiripan dalam implementasi AI ke dalam *website* menggunakan *framework python flask*.

2.2 Landasan Teori

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

2.2.1 Kebakaran Hutan dan Lahan

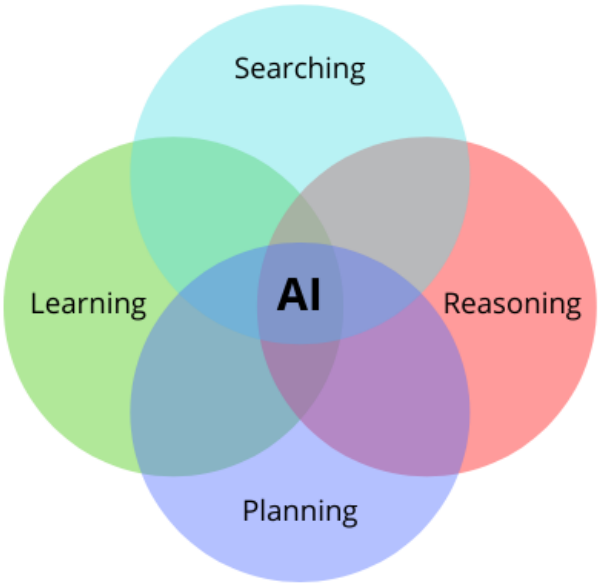
Kebakaran hutan dan lahan adalah kebakaran yang terjadi pada area hutan maupun lahan-lahan yang ditutupi oleh vegetasi alamiah. Ini bisa terjadi dikarenakan beberapa faktor, seperti kebakaran akibat sambaran petir, kecerobohan manusia seperti rokok atau pembakaran lahan, tindakan sengaja seperti pembersihan lahan. Kebakaran hutan memiliki dampak negatif yang sangat besar pada lingkungan, ekosistem, dan kesejahteraan manusia [16]. Ini menyebabkan kerusakan pada habitat hewan dan tumbuhan, mengurangi kualitas air dan udara, dan meningkatkan emisi gas rumah kaca. Kebakaran hutan juga menyebabkan kerugian ekonomi, termasuk hilangnya sumber daya alam dan pengaruh buruk pada sektor pariwisata.

2.2.2 *Deep Learning*

Perkembangan teknologi yang semakin meningkat di era *Society* 5.0 mendefinisikan bahwa teknologi dan manusia akan hidup untuk meningkatkan kualitas hidup manusia secara berkelanjutan. *Deep Learning* adalah aspek bagian

dari *Artificial Intelligence* dan *Machine Learning* dalam mengeksekusi dan menyelesaikan tugas-tugas yang sama seperti cara otak pada manusia bekerja.

Artificial Intelligence (AI) menjadi bagian dari ilmu komputer yang mengimitasi dari kecerdasan seperti otak manusia [25]. AI memiliki definisi lain yaitu sistem komputer yang dapat membantu kinerja manusia untuk menyelesaikan masalah tertentu dan penerapannya sudah banyak digunakan oleh berbagai bidang. AI dapat digunakan untuk memecahkan berbagai macam masalah, seperti menganalisis data, meningkatkan efisiensi bisnis, atau membantu dalam pengambilan keputusan. AI memiliki empat teknik dasar untuk melakukan pemecahan yang terdiri dari Pencarian, Penalaran, Perencanaan, dan Pembelajaran [26] seperti yang di tunjukan pada gambar 2.1.

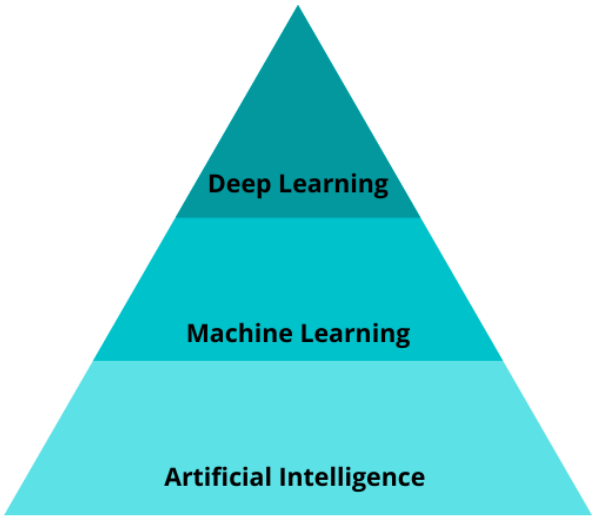


Gambar 2.1 Teknik Dasar Dalam AI

Berikutnya *Machine Learning* (ML) adalah pengembangan dari AI yaitu mesin yang dikembankan agar dapat belajar dengan otomatis tanpa adanya arahan dari pengguna [25]. ML dikembangkan dari disiplin ilmu lainya meliputi statistika, *data mining*, dan matematika yang dapat membuat mesin belajar dengan cara melakukan analisa data tanpa perlu diperintah ulang. ML memiliki kemampuan dalam memperoleh data yang didapatkan dengan perintahnya sendiri dan bisa

mempelajari data tersebut sehingga bisa melakukan tugas tertentu dan beragam, tergantung apa yang di pelajari oleh mesin.

Setelah memahami definisi dari AI dan ML kemudian *Deep Learning* yang memiliki pengertian sistem yang dapat mengadaptasi cara kerja struktur saraf otak manusia sehingga dapat membedakan objek. Hal tersebut yang diadaptasi oleh *deep learning* dengan membuat tiruan jaringan saraf yang disebut dengan *Artificial Neural Networks* (ANN). Algoritma pada *deep learning* memiliki kemampuan untuk belajar secara mandiri oleh karena itu dapat digunakan untuk masalah yang mudah hingga sulit. Gambar 2.2 adalah tingkatan pada AI, ML, dan *Deep Learning*, memberikan gambaran *level deep learning* merupakan bidang ilmu dari ML, sedangkan ML merupakan bidang ilmu dari AI.

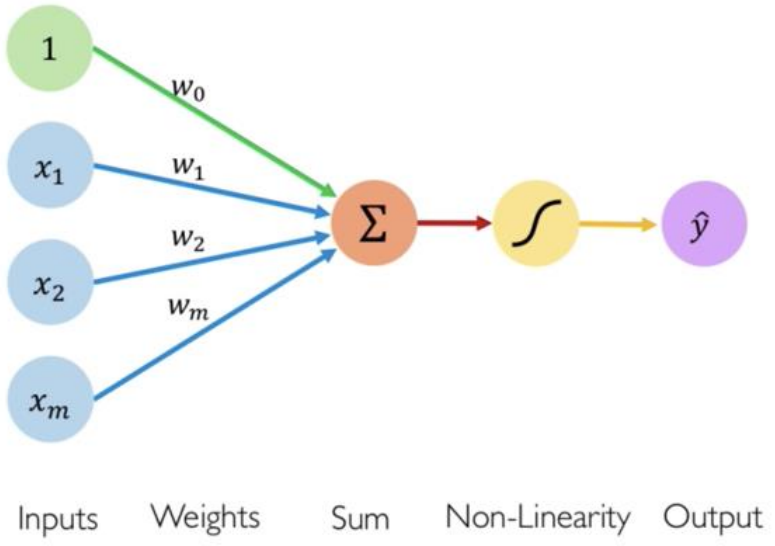


Gambar 2.2 *Level* Tingkatan Pada AI, ML, dan *Deep Learning*

2.2.3 *Artificial Neural Network*

Artificial Neural Network (ANN) adalah sebuah tiruan jaringan saraf yang didasarkan pada saraf biologis otak manusia bekerja dalam menerima dan merespons informasi [27]. ANN juga menjadi salah satu model *machine learning* yang multifungsi dan *powerfull*. Adanya kelebihan tersebut, ANN cocok digunakan untuk menangani permasalahan *machine learning* yang sangat sulit seperti melakukan klasifikasi gambar, video, dan lainnya.

Perceptron merupakan komponen dasar yang digunakan untuk membangun sebuah saraf tiruan yang pertama kali ditemukan oleh Frank Rosenblatt dari *Cornell Aeronautical Library* merupakan ilmuwan yang menemukan *perceptron* pada tahun 1957. *Perceptron* berfungsi untuk menerima masukan berupa bilangan numerik kemudian diproses yang nantinya akan menghasilkan sebuah keluaran. Gambar 2.3 merupakan model dari *artificial neuron*.



Gambar 2.3 Model *Artificial Neuron* [27]

Berikut ini merupakan penjelasan dari proses *perception* bekerja yang terdiri dari beberapa komponen yaitu *inputs, weights, sum, non-linearity*, dan *output* :

a. *Inputs* : Langkah pertama adalah masukan yang berfungsi untuk menerima masukan berupa angka-angka.
b. *Weights* : Langkah kedua adalah *weight* atau bobot berfungsi untuk memberikan bobot kepada masing-masing masukan yang nantinya akan di pelajari oleh *perception.*
c. *Sum* : Langkah ketiga adalah *sum* atau penjumlahan berfungsi untuk melakukan penjumlahan pada masukan yang nantinya setiap masukan akan dikalikan oleh bobotnya masing-masing kemudian ditambahkan dengan nilai bias yang berupa konstanta atau angka.

d. *Non-Linearity* : Langkah keempat adalah *non-linearlity* berfungsi untuk memberikan fungsi aktivasi atau *non-linearity function* agar perseptron dapat beradaptasi dengan pola data yang non linier.
e. *Output* : Terakhir ada *output* pada bagian ini berfungsi untuk memperoleh hasil *output* atau keluaran dari hasil perhitungan sebuah *perceptron*.

Fungsi matematis yang digunakan *perceptron* dapat di tunjukan pada persamaan di bawah. Rumus yang digunakan adalah simbol matematis yang mendefinisikan proses sebelumnya. *Output* (ŷ) merupakan bias (w_0), dijumlahkan dengan total setiap *input* (x_i) kemudian dikalikan dengan nilai masing-masing (w_i) sehingga diperoleh *weighted sum*, kemudian dimasukkan pada fungsi aktivasi (g).

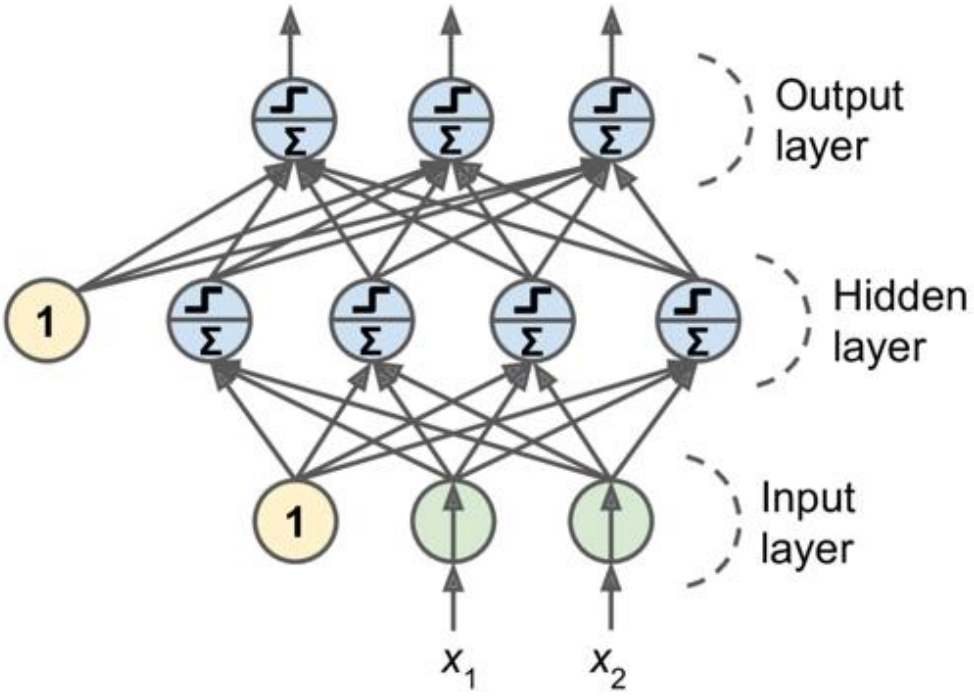
$$\text{Output} = \text{activation\_function}(\text{bias} + (\text{input1} * \text{weight1} + (\text{input2} * \text{weight2}) + \ldots + (\text{inputn} * \text{weightn})) \quad (2.1)$$

Di mana *output* adalah nilai yang dihasilkan oleh *perceptron* setelah melalui proses aktivasi. *Activation function* adalah fungsi aktivasi yang pakai dalam membatasi nilai output yang dihasilkan. Biasanya, fungsi aktivasi yang dipakai adalah step *function* atau *sigmoid function*. Bias adalah nilai yang ditambahkan ke *output* sebelum melalui proses aktivasi. *Input1, input2, ..., input-n* adalah nilai *input* yang diberikan ke *perceptron*. *Weight1, weight2, ..., weight-n* adalah bobot yang terhubung dengan *input-input* tersebut.

Berikut adalah contoh perhitungan menggunakan fungsi matematis perceptron dengan menggunakan step *function* sebagai fungsi aktivasi di mana input1 = 0.8, input2 = 0.1, weight1 = 0.5, weight2 = 0.9, dan bias = -0.3. Maka, *output* yang dihasilkan oleh *perceptron* adalah *output = activation_function* (-0.3 + (0.8 * 0.5) + (0.1 * 0.9)) kemudian *output = activation_function* (-0.3 + 0.4 + 0.09) dan didapatkan hasil *output = activation_function* (0.19). Karena step *function* hanya menghasilkan nilai 1 atau 0, maka *output* yang dihasilkan oleh *perceptron* adalah 1.

2.2.4 *Multi-Layer Perceptron*

Multi Layer Perceptron (MLP) adalah jenis jaringan saraf yang terdiri dari satu lapisan *input*, satu atau lebih lapisan tersembunyi, dan satu lapisan *output* yang merupakan pengembangan dari *perceptron* [28]. MLP berfungsi guna memberikan solusi dari permasalahan yang memerlukan pembelajaran serta penelitian pada ilmu saraf komputasi dan pemrosesan paralel. Gambar 2.4 merupakan contoh model MLP.

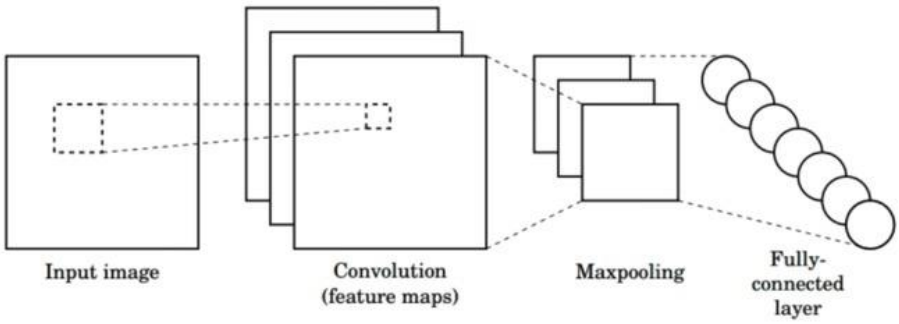


Gambar 2.4 Model *Multi Layer Perceptron* [28]

2.2.5 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah sebuah bagian dari jaringan saraf yang digunakan untuk mengidentifikasi objek pada gambar [29]. CNN merupakan perkembangan dari metode MLP yang dirancang khusus untuk memproses data dua dimensi. Cara kerja CNN pada umumnya mempunyai kemiripan dengan cara kerja MLP. Perbedaan antara CNN dengan MLP yaitu proses konvolusinya dan beberapa *hidden layer* pada CNN yang tidak dimiliki oleh MLP. CNN digunakan agar memungkinkan mesin atau komputer untuk melihat dan membedakan objek seperti yang dilakukan oleh mata manusia.

Arsitektur CNN mempunyai macam-macam jenis algoritma dan dapat dibedakan melalui jumlah lapisan dalam CNN. Algoritma CNN yang dibahas saat ini terdiri dari tiga bagian utama yaitu lapisan konvolusi, *max pooling*, dan lapisan *fully connected*. Gambar 2.5 di bawah merupakan contoh arsitektur CNN Sebuah gambar *input* dan dideteksi atribut atau fiturnya dengan menggunakan lapisan konvolusi 3 *filter*. Kemudian setelah melalui proses konvolusi akan dilakukan *Max Pooling* yang memperoleh 3 buah gambar dengan konvolusi yang memiliki resolusi kecil. Langkah selanjutnya hasil *max pooling* dapat diterapkan ke dalam sebuah *hidden layer* MLP yang nantinya akan dihubungkan menggunakan lapisan *fully connected* .



Gambar 2.5 CNN Menggunakan 3 *Layer* [30]

Berikut ini merupakan penjelasan mengenai fungsi dari lapisan konvolusi, *max pooling*, dan lapisan *fully connected* :

a. *Convolutional Layer*

Convolutional Layer atau layar konvolusi adalah suatu bagian dari arsitektur CNN yang memiliki fungsi utama untuk digunakan sebagai alat untuk mengenali gambar berdasarkan piksel atau atribut- atribut unik yang terdapat pada gambar nantinya layar konvolusi akan mengenali atribut-atribut unik pada sebuah gambar [31].

b. *Max Pooling*

Pada *max pooling* berfungsi untuk mengurangi ukuran gambar dari proses konvolusi dengan menggunakan *pooling layer* yang tersusun filter

yang mempunyai ukuran dan langkah yang akan berpindah - pindah di seluruh area peta fitur [31]. Lapisan *pooling* memiliki fungsi menurunkan tingkat resolusi gambar namun tetap menyimpan identitas unik yang ada di gambar sehingga dapat mempercepat proses komputasi pada proses *training* yang disebabkan oleh parameter diperbaharui semakin berkurang dan mengurangi *overfitting.*

c. *Fully Connected Layer*

Fully Connected Layer merupakan lapisan yang dapat menghubungkan *neuron* pada lapisan sebelumnya dengan *neuron* selanjutnya fungsi lapisan ini biasanya digunakan untuk menggabungkan *node* yang sebelumnya dua dimensi menjadi *node* satu dimensi [31].

2.2.6 *Transfer Learning*

Transfer learning adalah teknik yang digunakan dalam pembelajaran mesin untuk memindahkan pengetahuan yang telah diperoleh oleh model pembelajaran mesin dari satu tugas ke tugas lain yang terkait [32]. Tujuan dari *transfer learning* adalah untuk mempercepat proses pembelajaran model baru dengan menggunakan pengetahuan yang telah diperoleh oleh model lain yang telah terlatih pada *dataset* yang lebih besar atau lebih komprehensif. *Transfer learning* dapat membantu mengurangi kebutuhan akan *dataset* yang besar dan waktu pelatihan yang lama untuk membangun model pembelajaran mesin yang akurat. Ini sangat bermanfaat jika Anda ingin menerapkan model pembelajaran mesin ke dalam domain yang baru atau jika Anda hanya memiliki *dataset* yang kecil untuk dipelajari.

Cara kerja *transfer learning* adalah menggunakan model pembelajaran mesin yang telah terlatih pada *dataset* besar dan mengubahnya sedikit untuk sesuai dengan tugas baru yang akan diberikan. Biasanya, hanya sebagian kecil dari parameter model yang akan diubah dan dioptimalkan lagi melalui proses pelatihan ulang. *Transfer learning* sering digunakan dalam berbagai aplikasi pembelajaran mesin, seperti pengenalan objek dari gambar, pengenalan suara, dan pemrosesan bahasa natural. Dalam penerapan *transfer learning* terdapat dua faktor yang harus dilihat

terlebih dahulu yaitu ukuran atau jumlah *dataset* (banyak atau sedikit) dan kesamaan atau kemiripan *dataset* dengan sumber target yang digunakan berdasarkan dua faktor tersebut dapat disimpulkan aturan umum pada pendekatan *transfer learning* [33] seperti yang dijelaskan pada tabel 2.2 berikut.

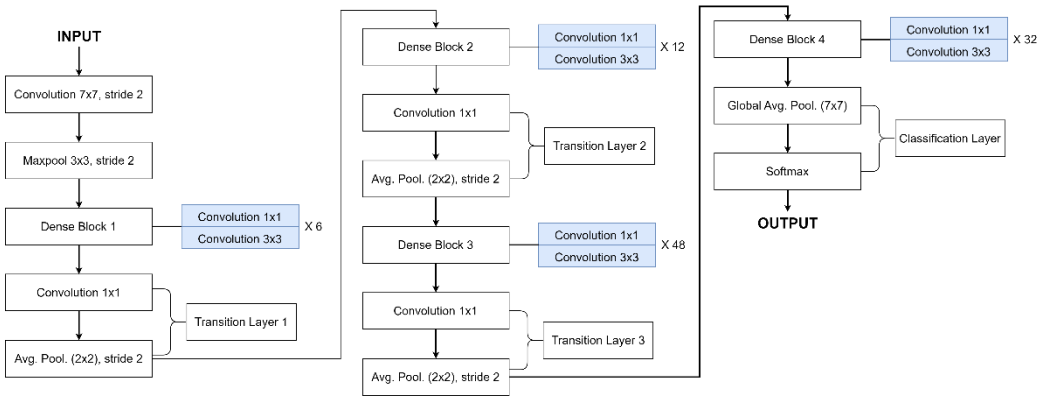
Tabel 2. 2 Pendekatan *Transfer learning*

Skenario	**Ukuran Data**	**Perbedaan Domain *Dataset* Asli dan Baru**	**Pendekatan**
1.	Kecil	Serupa	Melatih *layer* klasifikasi atau *classifier*
2.	Besar	Serupa	Membekukan 60%-80% jaringan *pre-trained* dan *layer* lainnya dipakai untuk melatih *dataset* baru
3.	Kecil	Sangat Berbeda	Membekukan sepertiga atau setengah dari jaringan *pre-trained*.
4.	Besar	Sangat Berbeda	Melatih semua jaringan dari awal dan tidak menerapkan *transfer learning*

2.2.7 *DenseNet201*

DenseNet201 adalah salah satu jenis arsitektur *deep learning* yang dikembangkan oleh Gao Huang dan timnya pada tahun 2016 [34]. Nama "*DenseNet201*" mengacu pada jumlah lapisan yang terdapat dalam arsitektur tersebut. Arsitektur *DenseNet* menggunakan konsep lapisan padat (*dense layer*), yang berarti setiap lapisan terhubung dengan semua lapisan sebelumnya dalam jaringan. Ini berbeda dengan arsitektur *deep learning* lainnya, di mana hanya lapisan-lapisan terpilih yang terhubung dengan lapisan sebelumnya. Dengan menghubungkan semua lapisan sebelumnya ke lapisan saat ini, *DenseNet* dapat memanfaatkan informasi yang lebih banyak dari seluruh jaringan, sehingga dapat meningkatkan kemampuan prediksi jaringan. Arsitektur ini juga memiliki kelebihan dalam mengurangi parameter yang harus dioptimalkan, sehingga dapat

mengurangi resiko *overfitting* dan dapat memuat pra pelatihan model. *DenseNet201* merupakan salah satu varian *DenseNet* yang memiliki 201 lapisan. Arsitektur ini dapat digunakan untuk berbagai macam aplikasi *deep learning*, seperti pengenalan objek, pengenalan wajah, dan lain-lain. Gambar 2.6 adalah arsitektur *DenseNet201*.



Gambar 2.6 Arsitektur *DenseNet201* [12]

DenseNet201 menggunakan beberapa jenis lapisan untuk membangun arsitektur jaringannya, antara lain :

a. Lapisan Konvolusi

Lapisan ini digunakan untuk mengekstrak fitur dari data *input* dengan menggunakan filter yang ditentukan. Lapisan konvolusi merupakan lapisan yang sering digunakan pada arsitektur *deep learning.*

b. Lapisan Transisi

Lapisan ini digunakan untuk mengubah tingkat kompleksitas fitur yang telah diekstrak oleh lapisan sebelumnya. Lapisan ini terdiri dari sebuah lapisan konvolusi dan lapisan penurunan ukuran (*downsampling layer*), untuk mengurangi jumlah parameter dan meningkatkan efisiensi pada jaringan.

c. Lapisan *Dense*

Lapisan ini merupakan ciri khas arsitektur *DenseNet*. Lapisan ini terdiri dari beberapa lapisan konvolusi yang terhubung secara padat dengan lapisan sebelumnya. Setiap lapisan padat akan menambahkan beberapa

lapisan konvolusi ke dalam jaringan, sehingga jaringan terus bertambah besar selama proses pelatihan.

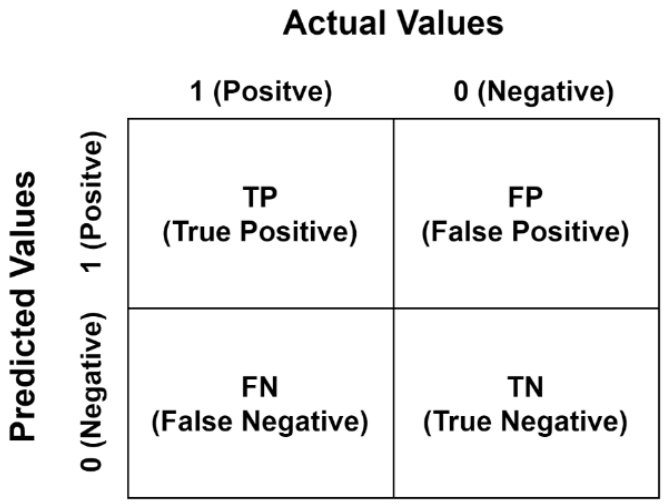
d. Lapisan Akhir

Lapisan ini merupakan lapisan terakhir pada jaringan, yang bertugas untuk melakukan prediksi atau klasifikasi terhadap data *input*. Lapisan ini terdiri dari sebuah lapisan penuh (*fully connected layer*) dan lapisan aktivasi (*activation layer*).

Selain lapisan-lapisan tersebut, *DenseNet201* juga mungkin menggunakan lapisan-lapisan tambahan seperti lapisan normalisasi (*normalization layer*) atau lapisan penurunan ukuran (*downsampling layer*) untuk meningkatkan performa jaringan.

2.2.8 *Confusion Matrix*

Confusion Matrix merupakan teknik yang dipakai untuk mengukur kinerja suatu model dalam klasifikasi [35]. Performa model dapat diukur menggunakan beberapa kriteria yang terdapat dalam matriks konfusi, yang memperolah empat nilai, yaitu *True Positif* (TP), *True Negatif* (TN), *False Positif* (FP), dan *False Negatif* (FN). Selain itu, matriks konfusi juga digunakan sebagai alat untuk mengevaluasi akurasi, *recall*, *precission* , dan skor F1 dari suatu model.



Gambar 2.7 Tabel *Confusion Matrix* 2 Kelas

Gambar 2.7 adalah ilustrasi dari tabel matriks konfusi yang terdiri dari dua kelas. Tabel tersebut terdiri dari nilai yang sebenarnya dan nilai yang akan diprediksi. Di dalam tabel memiliki beberapa elemen yang bisa dilakukan pengukuran nilainya, seperti yang akan dijelaskan selanjutnya :

1. *True Positif (TP)* merupakan jumlah prediksi yang benar dari kelas positif.
2. *True Negatif (TN)* merupakan jumlah prediksi yang benar dari kelas negatif.
3. *False Positif (FP)* merupakan jumlah prediksi yang salah dari kelas negatif.
4. *False Negatif (FN)* merupakan jumlah prediksi yang salah dari kelas positif.

Komponen tersebut dapat mengukur nilai *accuracy, recall, precision, F1-Score* dalam sebuah model yang dirancang untuk menyimpulkan bahwa model tersebut bagus atau tidak.

a. *Accuracy*

Accuracy adalah metrik yang berfungsi untuk mengukur seberapa baik suatu model dalam mengklasifikasikan data. Ini dihitung dengan menjumlahkan jumlah prediksi yang benar dari total jumlah prediksi yang dibuat oleh model. *Accuracy* memberikan gambaran umum tentang seberapa baik model dalam mengklasifikasikan data, namun tidak selalu menjadi metrik yang baik untuk digunakan dalam setiap kasus klasifikasi, terutama jika *dataset* tidak *balance* atau memiliki *imbalanced class*. Berikut ini merupakan rumus *accuracy* :

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.\ 2)$$

b. *Recall*

Recall adalah metrik yang berfungsi untuk mengukur seberapa baik suatu model dalam mendeteksi kelas positif. Ini dihitung dengan menjumlahkan jumlah t*rue positiv* dari total jumlah kelas positif yang sebenarnya. *Recall* memberikan informasi tentang seberapa baik model dalam mendeteksi kelas positif, dan sering digunakan dalam kasus-kasus

klasifikasi yang penting untuk mendeteksi kelas positif sebanyak mungkin. Rumus *recall* adalah sebagai berikut :

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

c. *Precision*

Precision adalah metrik yang berfungsi untuk mengukur seberapa baik suatu model dalam mengklasifikasikan data sebagai kelas positif. Dihitung dengan menjumlahkan jumlah *true positiv* dari total jumlah prediksi yang dibuat sebagai kelas positif oleh model, *Precision* memberikan informasi tentang seberapa baik model dalam mengklasifikasikan data sebagai kelas positif, dan sering digunakan dalam kasus-kasus klasifikasi yang penting untuk mengurangi jumlah prediksi yang salah sebagai kelas positif. Rumus *precision* adalah sebagai berikut :

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

d. *F1 Score*

F1 Score adalah metrik yang berfungsi untuk mengukur keseimbangan antara *precision* dan *recall*. *F1 Score* dihitung dengan menggabungkan *precision* dan *recall* menjadi satu skor dengan menggunakan *harmonic mean*. *F1 Score* mengambil kedua nilai *precision* dan *recall* ke dalam perhitungan, sehingga akan menjadi metrik yang baik digunakan jika kedua nilai ini diharapkan sama pentingnya. Rumus *F1 Score* adalah sebagai berikut :

$$F1\ Score = \frac{2(Recall * Precision)}{Recall + Precision} \tag{2.5}$$

2.2.9 *Python*

Python adalah bahasa pemrograman populer dan dipakai secara luas dalam bidang, termasuk pemrograman aplikasi, analisis data, pembelajaran mesin, pengembangan web, dan lainnya. *Python* memiliki kumpulan *library* yang kaya dan dukungan komunitas yang besar. *Python* juga dikenal dengan sintak yang intuitif dan mudah dipahami, sehingga membuatnya menjadi pilihan yang baik untuk pemula maupun profesional. Berikut ini merupakan beberapa *library* yang biasanya digunakan pengembang *machine learning* dan *data scientist* sebagai berikut :

a. *TensorFlow*

TensorFlow merupakan sebuah *platform end-to-end open source* dibuat dan dikembangkan oleh *Google Brain* dan banyak digunakan dalam pengembangan model *machine learning*. *TensorFlow* menyediakan ekosistem *tools, library,* dan sumber daya komunitas yang fleksibel dan *komprehensif* sehingga memudahkan para pengembang untuk membangun dan menerapkan aplikasi *machine learning* dengan mudah.

b. *Keras*

Keras merupakan API pada *deep learning* yang ditulis dengan bahasa pemograman *python* yang merupakan jaringan saraf buatan tahapan tertinggi dan dapat dijalankan di atas platform *tensorflow, CNTK,* dan *theano*. Penggunaan keras banyak diminati karena dalam pengembangan dan pembuatan *deep learning*.

c. *NumPy*

NumPy atau *Numerical Python* merupakan *library* pada bahasa pemograman *python* yang dapat dipakai untuk mengolah *array* dan juga memiliki fungsi untuk mengolah domain aljabar linier, transformasi *fourier*, dan matriks. *Library NumpPy* dibuat oleh Travis Oliphant pada tahun 2005 merupakan proyek *open source* sehingga dapat digunakan dengan bebas. Meskipun *python* memiliki fungsi yang dapat mengolah *array*, akan tetapi proses yang dibutuhkan memerlukan waktu yang cukup lama jika *array* yang

diproses memiliki ukuran yang besar dengan adanya *NumPy* dapat mempercepat proses pengolahan *array* agar lebih cepat.

d. *Sciket-Learn*

Scikit-learn merupakan *library* pada bahasa pemograman *python* yang dapat dipakai untuk data yang kompleks. *Library Sciket-Learn* ini bersifat *open source* sehingga dapat digunakan dengan bebas *library* ini juga mendukung *machine learning* dengan mendukung berbagai program klasifikasi, regresi linier, dan lainnya.

e. *Pandas*

Pandas merupakan *library* yang sering digunakan para *data scientist. Library Pandas* dapat digunakan untuk *machine learning* dan bersifat *open source* sehingga dapat digunakan dengan bebas. *Library* ini memiliki beberapa keunggulan yaitu fleksibel dalam melakukan analisis. Penggunaan *Pandas* dapat diterapkan untuk memudahkan dalam melakukan analisis, manipulasi, dan pembersihan data.

f. *Matplotlib*

Matplotlib merupakan *library* pada bahasa pemograman *python* Jenis library ini dapat digunakan untuk melakukan plot angka-angka berdefinisi tinggi seperti diagram lingkaran, histogram, *scatterplot*, grafik, dan lain-lain. *Matplotlib* juga bisa digunakan untuk merencanakan data numerik dengan fungsinya tersebut *Matplotlib* banyak digunakan untuk melakukan analisis data dan *library Matplotlib* ini bersifat terbuka sehingga bisa digunakan dengan bebas.

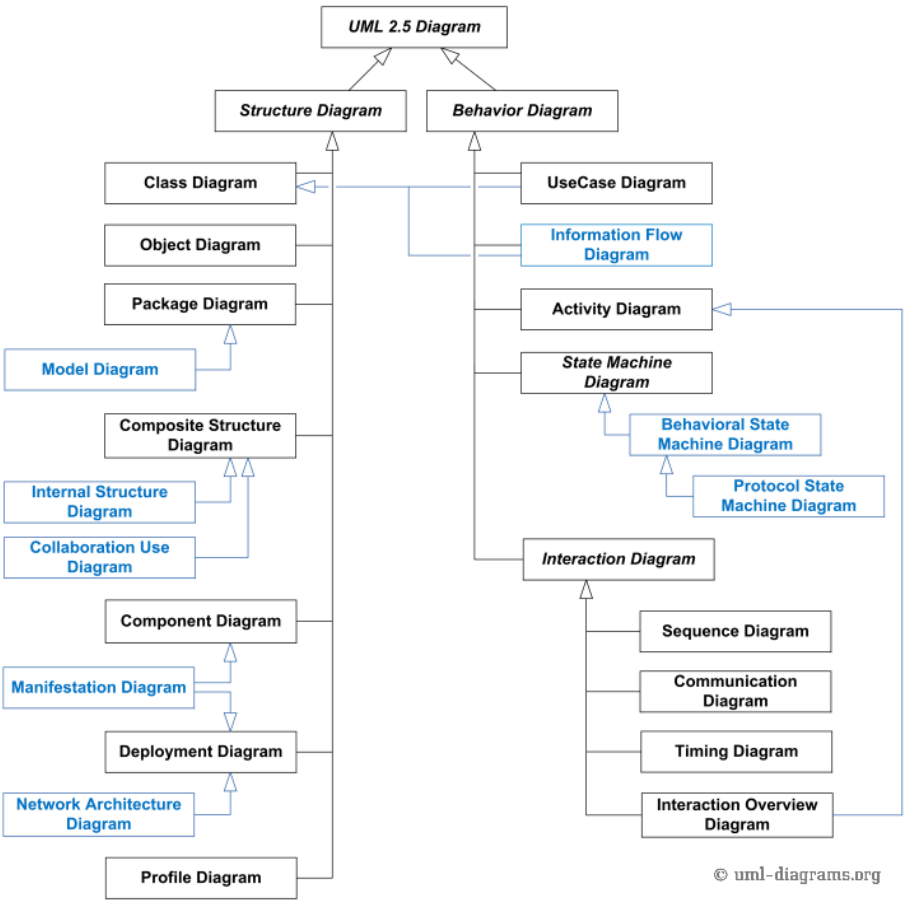
2.2.10 *Flask*

Flask merupakan *web framework* pada bahasa pemograman *python* yang dapat digunakan untuk pengembangan *website* berbasis *python* dan tergolong dalam *micro-framework* karena tidak membutuhkan *tool* dan *library* tertentu dalam

penggunanya sehingga fleksibilitas dan skalabilitas dari *flask* cukup baik jika dibandingkan dengan kerangka kerja lain. *Flask* digunakan sebagai kerangka kerja untuk membuat tampilan *website* dan juga dapat digunakan untuk menjalankan aplikasi kecerdasan buatan [24]. Dengan menggunakan *Flask*, *developer* dapat dengan mudah membuat *website* yang terstruktur dan menerapkan implementasi AI di dalamnya. Contoh aplikasi *web* yang mengimplementasikan AI dengan *Flask* adalah aplikasi yang menerima gambar sebagai *input* dan mengembalikan klasifikasi gambar tersebut atau aplikasi yang menerima teks sebagai *input* dan mengembalikan prediksi yang dibuat oleh model *deep learning*.

2.2.11 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) merupakan bahasa yang digunakan untuk menggambarkan, menganalisis, dan mendeskripsikan model sistem informasi atau aplikasi [36]. UML menyediakan seperangkat simbol dan notasi yang digunakan untuk menggambarkan perilaku dari sistem dan strukturnya yang akan dibuat. UML dikembangkan oleh *Object Management Group* (OMG) dan merupakan standar industri yang banyak digunakan dalam dunia pengembangan perangkat lunak. UML digunakan untuk membantu dalam proses perancangan sistem, memahami spesifikasi sistem yang sudah ada, dan mengkomunikasikan ide dan rencana pengembangan sistem kepada tim pengembangan.. Diagram UML versi 2.5 merupakan versi paling baru yang dirilis pada tahun 2015 terbagi menjadi dua kategori diagram yaitu diagram struktur dan diagram perilaku seperti yang ditampilkan pada gambar 2.8.



Gambar 2.8 Gambaran Umum Diagram UML Versi 2.5 [36]

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

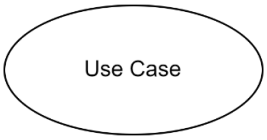
1. ***Use Case Diagram***

   Use Case Diagram adalah jenis diagram dalam UML yang digunakan untuk menggambarkan sebuah interaksi antara sistem dan pengguna. *Use case diagram* terdiri dari beberapa elemen sebagai berikut :

   a. *Actor* dapat digunakan untuk mengidentifikasi dan mendokumentasikan kebutuhan sistem, sehingga membantu dalam proses perancangan sistem dan mengkomunikasikan kebutuhan sistem kepada tim pengembangan. Gambar 2.9 merupakan elemen *actor*.

Gambar 2.9 Elemen *Actor*

b. *Use Case* adalah salah satu elemen yang digunakan untuk menggambarkan tindakan atau fungsi yang dapat dilakukan oleh sistem. Gambar 2.10 merupakan elemen *use case*.

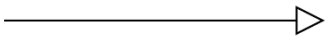


Gambar 2.10 Elemen *Use Case*

c. *Association Relationship* adalah salah satu elemen yang digunakan sebagai relasi yang menggambarkan bahwa dua *use case* saling terkait. Gambar 2.11 merupakan elemen *association relationship*.

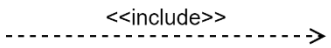
Gambar 2.11 Elemen *Association Relationship*

d. *Generalization Relationship* adalah salah satu elemen yang digunakan sebagai relasi yang menggambarkan bahwa suatu *use case* merupakan turunan dari *use case* lain. Gambar 2.12 merupakan elemen *generalization relationship*.



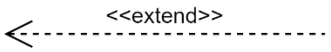
Gambar 2.12 Elemen *Generalization Relationship*

e. *Include Relationship* adalah salah satu elemen yang digunakan sebagai relasi yang menggambarkan bahwa suatu *use case* mencakup *use case* lain. Gambar 2.13 merupakan elemen *include relationship*.



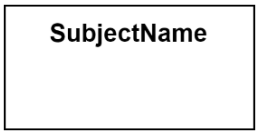
Gambar 2.13 Elemen *Include Relationship*

f. *Exclude Relationship* adalah salah satu elemen yang digunakan sebagai relasi yang menggambarkan bahwa suatu *use case* tidak dapat digabungkan dengan *use case* lain. Gambar 2.14 merupakan elemen *exclude relationship*.



Gambar 2.14 Elemen *Exclude Relationship*

g. *Use Case Subject* adalah salah satu elemen yang digunakan sebagai objek yang digambarkan dalam *use case*, yang mewakili sistem atau komponen yang digambarkan dalam *use case*. Gambar 2.15 merupakan elemen *use case subject*.



Gambar 2.15 Elemen *Use Case Subject*

2. *Activity Diagram*

Activity Diagram adalah jenis diagram UML yang digunakan untuk menggambarkan aliran aktivitas atau proses pada suatu sistem. *Activity diagram* menggambarkan bagaimana aktivitas yang berbeda terkait satu sama lain dan bagaimana sistem merespons aktivitas tersebut. *activity diagram* terdiri dari beberapa elemen sebagai berikut :

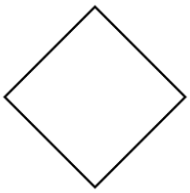
a. *Initial Node* adalah elemen yang digunakan untuk menggambarkan awal dari aktivitas. Gambar 2.16 merupakan elemen *initial node*.

Gambar 2.16 Elemen *Initial Node*

b. *Activity* adalah elemen yang digunakan untuk menggambarkan tindakan yang dilakukan dalam sistem. Gambar 2.17 merupakan elemen *activity*.

Gambar 2.17 Elemen *Activity*

c. *Decision* adalah elemen yang digunakan untuk menggambarkan kondisi atau pemilihan yang harus dilakukan dalam sistem. Gambar 2.18 merupakan elemen *decision*.



Gambar 2.18 Elemen *Decision*

d. *Join* adalah elemen yang digunakan untuk menggabungkan dua atau lebih aliran aktivitas. Gambar 2.19 merupakan elemen *join*.

Gambar 2.19 Elemen *Join*

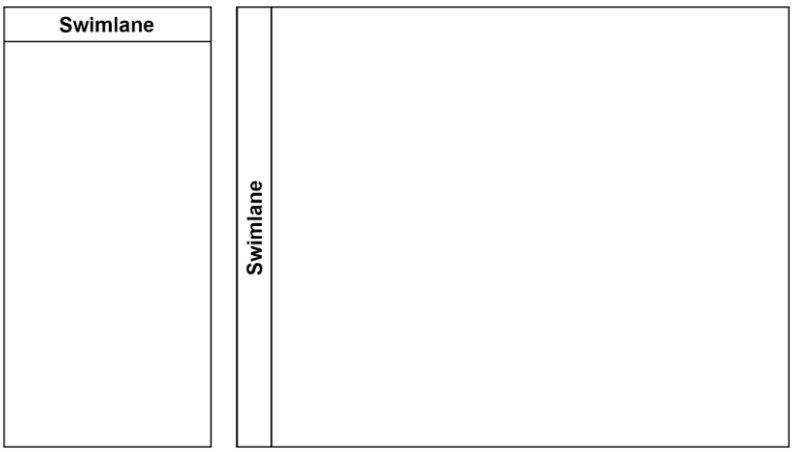
e. *Control Flow* adalah elemen yang digunakan untuk menggambarkan aliran aktivitas dari satu elemen ke elemen lain. Gambar 2.20 merupakan elemen *control flow*.

Gambar 2.20 Elemen *Control Flow*

f. *Final Node* adalah elemen yang digunakan untuk menggambarkan akhir dari aktivitas. Gambar 2.21 merupakan elemen *final node*.

Gambar 2.21 Elemen *Final Node*

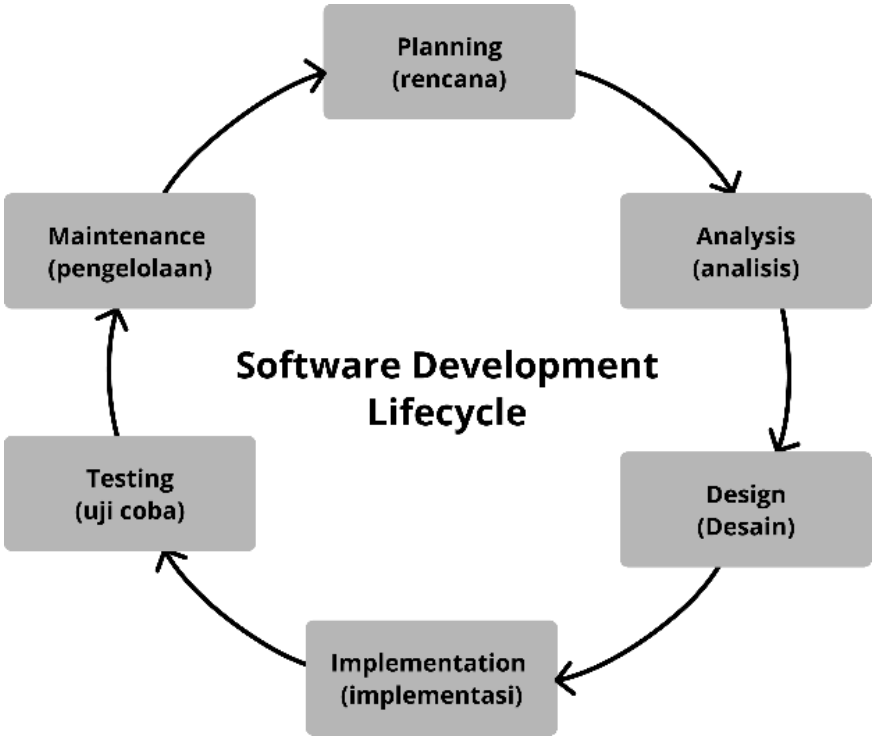
g. *Swimlane* adalah elemen yang digunakan untuk mengatur aktivitas dalam beberapa area yang disebut *swimlane*. *Swimlane* digunakan untuk membagi dan mengatur aktivitas dalam beberapa bagian dari sistem. Gambar 2.22 merupakan elemen *swimlane*.



Gambar 2.22 Elemen *Swimlane*

2.2.12 *Software Development Life Cycle*

Software Development Life Cycle (SDLC) adalah proses yang digunakan dalam pengembangan perangkat lunak untuk mengelola dan mengendalikan aktivitas pengembangan perangkat lunak dari awal sampai akhir [21]. Setiap tahap dari SDLC harus dilakukan secara teratur dan sistematis untuk memastikan bahwa sistem yang dikembangkan sesuai dengan kebutuhan pengguna dan dapat digunakan dengan aman dan efektif seperti pada gambar 2.23.



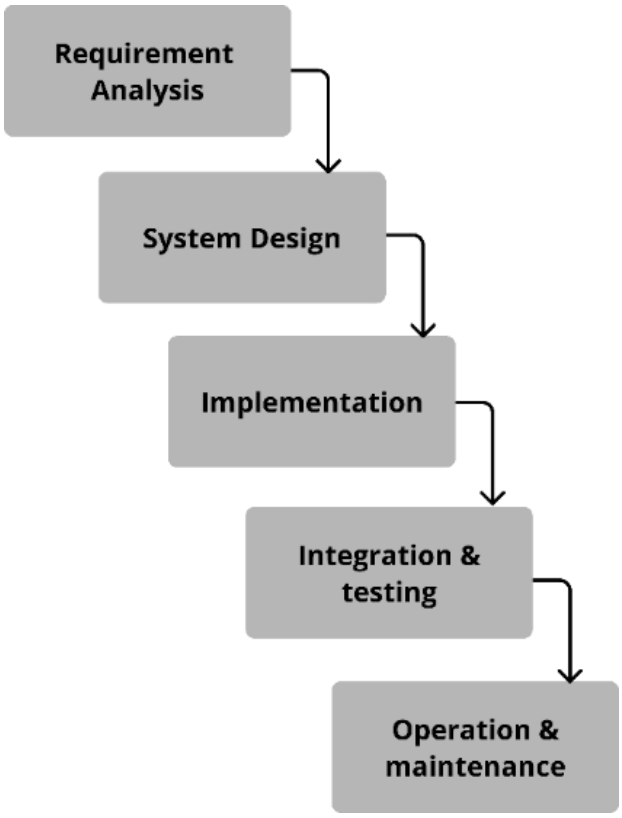
Gambar 2.23 Tahap *Software Development Life Cycle* (SDLC)

SDLC terdiri dari beberapa tahap yang harus dilalui dalam proses pengembangan perangkat lunak, yaitu :

1. Analisis Kebutuhan : Mengumpulkan kebutuhan pengguna dan mendefinisikan spesifikasi sistem yang akan dikembangkan.
2. Desain : Mendesain arsitektur sistem dan menentukan teknologi yang digunakan.
3. Implementasi : Menulis kode program dan menguji komponen sistem.

4. Uji coba : Melakukan pengujian sistem untuk menemukan dan memperbaiki kesalahan.
5. *Deployment* : Mengimplementasikan sistem ke lingkungan produksi.
6. *Maintenance* : Melakukan pemeliharaan dan perbaikan sistem setelah diimplementasikan.

SDLC memiliki sejumlah tahapan metode yang umum digunakan untuk menjalankan proses SDLC salah satunya yaitu *waterfall*. *Waterfall* adalah salah satu metode dalam *Software Development Life Cycle* (SDLC) yang mengikuti proses linier dan berurutan [21]. Setiap tahap dalam proses pengembangan perangkat lunak harus diselesaikan sebelum tahap berikutnya dimulai. Ini berarti bahwa setiap tahap dalam proses pengembangan perangkat lunak harus benar-benar selesai sebelum tahap berikutnya dimulai. Gambar 2.24 merupakan tahapan dari metode *waterfall*.



Gambar 2.24 Tahap Model *Waterfall*

a. *Requirement Analysis*

Tahapan pertama model *waterfall* adalah mempersiapkan dan menganalisis apa saja kebutuhan dari *software* yang akan dibuat yang sesuai dengan kebutuhan pengguna dan dapat mengetahui batasan dari *software* yang akan di buat. Informasi tersebut dapat diperoleh dengan cara wawancara, survei, mencari studi literatur, observasi, dan diskusi.

b. *System Design*

Tahapan kedua model *waterfall* adalah membuat desain dari *software* yang akan dibuat sebelum masuk ke dalam proses *coding*. Tujuannya dilakukannya tahap tersebut untuk memudahkan *programmer* dalam mengimplentasikan atau mengeksekusi tampilan dan antarmuka *software* yang dibuat.

c. *Implementation*

Tahapan ketiga dari model *waterfall* adalah melakukan implementasi atau eksekusi program dengan menggunakan bahasa pemograman dan *tools* yang sesuai dengan kebutuhan dari *software* yang akan dibangun. Oleh karena itu pada tahap ini berfokus membuat hasil desain yang telah dibangun sebelumnya untuk diterjemahkan ke dalam bahasa pemograman.

d. *Integration & Testing*

Tahapan keempat dari model *waterfall* adalah melakukan integrasi dan pengujian *software* yang telah dibuat sebelumnya di mana sistem tersebut sudah sesuai dengan desain dan fungsi pada *software* berjalan dengan benar.

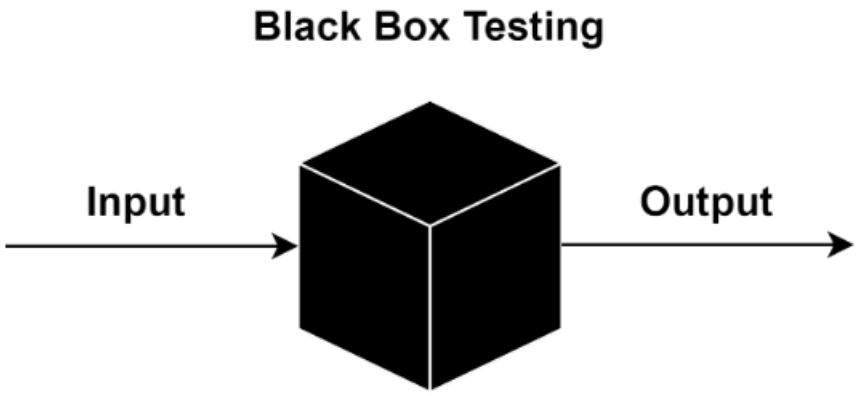
e. *Operation & Maintenance*

Tahapan terakhir pada model *waterfall* yaitu melakukan pengoprasian dan perbaikan *software* yang telah dibuat setelah melewati tahap pengujian *software* tersebut akan masuk ke tahap pemakaian oleh pengguna dan jika

terjadi kesalahan dalam *software* yang dibuat nantinya dapat dilakukan perbaikan dari kesalahan yang di temukan oleh pengguna.

2.2.13 Pengujian Sistem

Pengujian sistem adalah proses verifikasi dan validasi suatu sistem untuk memastikan bahwa sistem tersebut memenuhi spesifikasi yang ditentukan, sesuai dengan kebutuhan pengguna, dan dapat digunakan dengan aman dan efektif [21]. Pengujian sistem dapat dilakukan melalui berbagai metode salah satunya yaitu *blackbox testing.* Gambar 2.25 menampilkan konsep dari *blackbox testing*.



Gambar 2.25 Model *Black Box Testing*

Blackbox testing adalah metode pengujian sistem di mana *tester* tidak memiliki informasi tentang implementasi internal sistem yang diuji. *Tester* hanya mengetahui *input* yang diberikan ke sistem dan *output* yang diharapkan dari sistem. Tujuan dari *blackbox testing* adalah untuk memastikan bahwa sistem dapat memproses *input* yang valid dan menghasilkan *output* yang sesuai dengan spesifikasi.