BAB II

LANDASAN TEORI

2.1 Tinjuan Pustaka

Dalam penelitian ini, penulis mengacu pada tujuan pustaka atau referensi dari penelitian sebelumnya mengenai Metode *Deep Learning* MobileNet-SSD. Penulis mempelajari tabel penelitian terkait sebagai bagian dari kajian literatur untuk menyelesaikan tugas akhir skripsi.

Tabel 2.1 Penelitian Sebelumnya

No	Penulis	Tahun	Metode	Judul	Hasil
1	Mohammad Heri Saputra, Danang Erwanto, Royb Fatkhur Rizal	2022	*Deep Learning* MobileNet-SSD	Penghitung Jumlah Pengunjung Objek Wisata dengan Metode *Deep Learning* MobileNet-SSD	Penelitian ini mengembangkan sistem untuk menghitung jumlah pengunjung dengan menggunakan webcam yang dipasang baik di pintu masuk maupun pintu keluar objek wisata. Metode yang digunakan dalam penelitian ini adalah memanfaatkan webcam untuk tujuan tersebut

					Dengan menggunakan model *deep learning* yang telah terlatih sebelumnya, yaitu MobileNet-SSD, penelitian ini berhasil mendeteksi pengunjung yang masuk dan keluar untuk menghitung jumlah pengunjung serta mengatur pembatasan jumlah pengunjung di objek wisata.
2	Maulia Rahman, Dedi Leman	2022	MobileNet-SSD	*Movidius Neural Compute Stick* untuk Pendeteksian Objek Manusia Secara Real Time dengan Metode MobileNet-SSD	Penelitian ini bertujuan untuk mengembangkan sistem yang dapat meningkatkan kinerja kamera pengintai dalam mendeteksi dan menghitung jumlah manusia menggunakan Movidius NCS pada Raspberry Pi. Tujuan utama adalah agar kamera aktif dan memberikan hasil optimal, sambil mengurangi penggunaan ruang penyimpanan yang berlebih. Sistem pendeteksi objek manusia dalam penelitian ini menggunakan teknik *Deep Learning* dengan model arsitektur jaringan Mobilenet-SSD. Uji coba dilakukan dalam berbagai kondisi cahaya dari 50 hingga 550 lux dan jarak objek antara 1 hingga 10 meter. Hasil penelitian menunjukkan bahwa sistem mencapai akurasi sebesar 91,67% dengan efisiensi penyimpanan mencapai 49,24%.

3	Rachmat Muwardi, Joe Mada Ranseda Permana, Hongmin Gao, Mirna Yunita	2023	MobileNet-SSD	*Human Object Detection for Real-Time Camera using MobileNet-SSD*	Penulis melakukan penelitian tentang deteksi objek manusia menggunakan sebuah perangkat sederhana. Dalam penelitian ini, penulis mengimplementasikan algoritma MobilenetV2-SSD, yang dikenal memiliki tingkat deteksi dan akurasi yang tinggi. Penggunaan MobilenetV2-SSD untuk pengenalan objek manusia menghasilkan tingkat deteksi sebesar 100% dengan kecepatan frame per detik (FPS) sebesar 5.

4	Yosia Pradeska Admaja	2021	*Single Shot Detector* (SSD)	Sistem Penghitung Jumlah Pengunjung di Restoran Menggunakan Kamera Berbasis *Single Shot Detector* (SSD)	Setelah pengujian dilakukan, aplikasi penghitung jumlah pengunjung ini berhasil membedakan manusia dari objek lain, sehingga mempermudah proses penghitungan pengunjung yang masuk, keluar, dan yang masih berada di ruangan yang sama. Hasil perhitungan ditampilkan di bagian kanan atas layar setelah menggunakan video rekaman dan juga disimpan dalam file *spreadsheet* (.csv). Tingkat akurasi perhitungan mencapai 86%. Selain itu, aplikasi ini mencatat akurasi 100% dalam mendeteksi objek yang sedang berjalan, 90% untuk objek yang bergerak cepat, dan 50% untuk objek yang sedang berlari.

5	Ari Kurnia Ramadan dan Sularso Budi Laksono	2023	*Computer Vision* (MobileNet SSD)	Rancang Bangun Aplikasi Deteksi Objek Untuk Menghitung Jumlah Pengunjung Restoran Berbasi Computer Vision	Restoran perlu memiliki sistem yang memudahkan mereka untuk mendapatkan informasi tentang jumlah pengunjung setiap harinya, agar data ini bisa digunakan dalam laporan mengenai minat masyarakat terhadap restoran tersebut. Oleh karena itu, diperlukan sebuah aplikasi deteksi objek yang menggunakan OpenCV dan metode Mobilenet-SSD untuk menghitung jumlah pengunjung. Aplikasi ini akan memungkinkan pihak restoran untuk secara efisien mendapatkan data pengunjung yang terdeteksi dan terhitung, serta menggunakan Laravel untuk antarmuka pengguna (UI) yang akan menampilkan data tersebut.

2.1.1 Penghitung Jumlah Pengunjung Perpustakaan

Sebagian besar pengelola perpustakaan saat ini masih mengandalkan metode manual untuk menghitung jumlah pengunjung yang datang, baik menggunakan alat counter atau sistem kartu perpustakaan. Namun, metode ini dianggap kurang efektif dan efisien karena rentan terhadap kesalahan seperti kesalahan pencatatan atau gangguan teknis. Kesalahan ini dapat menyebabkan penghitungan ganda atau kehilangan data yang penting. Selain tidak akurat, penghitungan manual juga memakan waktu dan tenaga yang banyak, terutama di perpustakaan yang ramai.

Untuk mengatasi masalah ini, dibutuhkan sistem yang lebih canggih dan otomatis untuk mendeteksi serta menghitung jumlah pengunjung perpustakaan. Sistem ini menggunakan teknologi sensor video dengan metode deep learning MobileNet-SSD menggunakan OpenCV Python. Metode ini dipilih karena mampu mendeteksi objek langsung dan dengan cepat, serta dapat menghitung jumlah pengunjung secara otomatis. Implementasi sistem ini diharapkan dapat membantu pengelola perpustakaan dalam mengoptimalkan penggunaan sumber daya mereka berdasarkan data jumlah pengunjung yang lebih akurat..

2.1.1.1 Metode Mendeteksi Jumlah Pengunjung

Penelitian sebelumnya dengan judul "Pengembangan Sistem Penghitung Pengunjung Ruang Baca Departemen Teknik Elektro Menggunakan Teknologi Mikrokontroler" menggunakan metode penelitian eksperimen yang meliputi beberapa tahap. Tahap awal yang krusial dalam proses ini adalah desain dan pembuatan alat, dengan tujuan utama untuk menentukan komponen yang dibutuhkan agar hasil akhir sesuai dengan harapan. Proses perancangan dan

pembuatan alat ini mencakup pembuatan blok diagram, desain elektronik, desain mekanik, penjelasan prinsip kerja, serta analisis hasil pengujian.

Pengujian alat dilakukan dengan menguji sensor ultrasonik, sensor garis, dan fungsi keseluruhan alat. Alat ini terbagi menjadi tiga bagian: sistem input yang terdiri dari dua sensor ultrasonik dan sensor garis yang dikontrol oleh mikrokontroler Arduino Nano sebagai pusat pengontrolan masukan. Selain itu, terdapat Nodemcu ESP8266 sebagai mikrokontroler tambahan yang berfungsi sebagai penghubung ke jaringan internet. Nodemcu ESP8266 menghasilkan data jumlah pengunjung yang ditampilkan pada layar LCD dan juga dikirimkan ke perangkat lunak Ubidots.

Penelitian lain yang terkait, berjudul "Penghitung Jumlah Pengunjung Objek Wisata dengan Metode Deep Learning MobileNet-SSD," membahas tentang penggunaan model pra-terlatih MobileNet-SSD untuk mendeteksi dan menghitung jumlah pengunjung pada objek wisata. Sistem ini menggunakan webcam yang dipasang pada pintu masuk dan keluar objek wisata untuk penghitungan yang lebih akurat dan otomatis..

2.1.1.2 **Kecerdasan Buatan (*Artificial Intelligence)***

Komputer memandang dan memproses gambar dengan cara yang sangat berbeda dibandingkan manusia. Bagi komputer, gambar hanya terdiri dari kumpulan piksel – baik dalam format vektor atau raster. Pada gambar raster, setiap piksel diatur dalam grid, sedangkan pada gambar vektor, mereka diatur sebagai poligon dengan warna yang berbeda.

Selama proses pengolahan data, setiap gambar dikelompokkan ke dalam kategori, dan fitur-fitur fisik diekstraksi. Terakhir, representasi geometris dikonversi

menjadi label yang menggambarkan gambar tersebut. Tahap ini – pengumpulan, pengorganisasian, pelabelan, dan anotasi gambar – sangat penting untuk memaksimalkan kinerja model visi komputer.

Setelah dataset pelatihan yang komprehensif dikembangkan dengan baik, algoritma pengenalan gambar bekerja untuk mengekstrak pola-pola dari gambar.

1. Pengenalan Wajah

   AI dilatih untuk mengenali wajah dengan memetakan fitur-fitur wajah seseorang dan membandingkannya dengan gambar-gambar dalam database deep learning untuk menemukan kecocokan..

2. Identifikasi Objek

   Teknologi pengenalan gambar membantu dalam menemukan objek menarik dalam bagian gambar yang dipilih. Pencarian visual dimulai dengan mengidentifikasi objek dalam gambar dan membandingkannya dengan gambar-gambar di web..

3. Deteksi Teks

   Sistem pengenalan gambar juga membantu dalam menemukan teks dari gambar dan mengonversinya ke dalam format yang dapat diproses mesin menggunakan pengenalan karakter optik.

Gambar 2.1 Siswa Pengunjung Perpustakaan

2.1.2 Proses Sistem Pengenalan Gambar

Tiga langkah berikut membentuk latar belakang dimana gambar pengakuan bekerja.

1. Proses 1 : kumpulan data pelatihan

   Semua sistem pengenalan gambar dimulai dengan dataset pelatihan yang terdiri dari gambar dan video. Setelah itu, jaringan saraf membutuhkan dataset pelatihan untuk mengenali pola dan membentuk persepsi.

2. Proses 2 : pelatihan jaringan saraf

   Setelah dataset dikembangkan, dataset tersebut dimasukkan ke dalam algoritma jaringan saraf. Hal ini menjadi dasar untuk pengembangan alat pengenalan gambar, di mana algoritma tersebut memungkinkan jaringan saraf untuk mengenali kelas-kelas gambar.

3. Proses 3 : pengujian

   Model pengenalan gambar sebaiknya diuji dengan ketat sesuai dengan kualitasnya. Oleh karena itu, penting untuk menguji kinerja model menggunakan gambar yang tidak termasuk dalam dataset pelatihan. Secara umum, disarankan untuk menggunakan sekitar 80% dari dataset untuk melatih model dan menyisakan 20% untuk pengujian model. Kinerja model dinilai berdasarkan akurasi, prediktabilitas, dan kegunaannya.

2.1.2.1 Pengertian dan Cangkupan *Computer Vision*

Computer vision adalah cabang ilmu komputer yang meniru cara kerja mata manusia dan menerapkannya pada komputer. Tujuannya adalah agar komputer dapat melihat, mengidentifikasi, dan memproses gambar dengan cara yang mirip dengan manusia, serta menghasilkan output yang relevan. Ini melibatkan pengenalan dan interpretasi objek serta analisis gambar. Computer vision tidak

hanya fokus pada kemampuan melihat, tetapi juga pada pengolahan informasi yang diperoleh dari pengamatan, seperti membuat gambar 3D dari gambar 2D. Contohnya, pada mobil yang dilengkapi dengan teknologi computer vision, mobil dapat mengenali dan membedakan objek di jalan seperti lampu lalu lintas, rambu-rambu, atau pejalan kaki. Teknologi ini memungkinkan mobil untuk memberikan peringatan kepada pengemudi atau bahkan berhenti secara otomatis jika ada hambatan mendadak di jalan.

Gambar 2.2 Contoh hasil dari *Computer Vision*

2.1.2.2 OpenCV (Open Source *Computer Vision*)

OpenCV adalah sebuah library yang awalnya dikembangkan oleh pusat penelitian Intel di Nizhny Novgorod, Rusia. Library ini dapat digunakan dalam berbagai bahasa pemrograman, termasuk Python, C, C++, dan Java, serta dapat diimplementasikan di berbagai sistem operasi seperti Linux, Windows, iOS, macOS, dan Android. Pengembangan OpenCV memberikan dampak signifikan dalam meningkatkan efisiensi komputasi, khususnya untuk aplikasi real-time.

Tujuan utama OpenCV adalah untuk meniru cara kerja visual atau penglihatan manusia dalam sistem komputer.

2.1.2.3 Jenis Deep Learning dan *Convolutional Neural Network* (CNN)

Deep Learning merupakan bagian integral dari bidang kecerdasan buatan dan machine learning yang mengembangkan neural network dengan multiple layer untuk menyelesaikan berbagai tugas seperti deteksi objek, pengenalan suara, dan terjemahan bahasa, di antara lain. Teknik ini berbeda dari machine learning konvensional karena secara otomatis mampu merepresentasikan data seperti gambar, video, atau teks tanpa memerlukan aturan kode atau pengetahuan domain manusia (Pumsirirat, 2018).

Deep Learning menggunakan jaringan syaraf tiruan (Artificial Neural Networks atau ANN) yang belajar di berbagai tingkat abstraksi. Tingkat representasi yang lebih tinggi dibentuk dari tingkat konseptual yang lebih rendah, dan konsep pada tingkat yang lebih rendah membantu mendefinisikan banyak konsep tingkat yang lebih tinggi (Deng, 2014). Deep Learning juga terinspirasi dari struktur otak manusia, di mana ANN yang terdiri dari tiga atau lebih lapisan mampu belajar dan beradaptasi dengan data dalam jumlah besar serta menangani masalah yang sulit untuk diselesaikan dengan algoritma machine learning tradisional.

Sebagai contoh, implementasi *Deep Learning* yang umum adalah pada sistem AI untuk permainan catur, di mana kecerdasan buatan sulit dikalahkan bahkan oleh pemain berpengalaman. Hal ini dikarenakan AI mampu menganalisis jutaan langkah dari pertandingan sebelumnya dengan cepat, mencari solusi yang optimal dalam waktu singkat. Kemampuan ini memungkinkan model *Deep Learning* untuk belajar dari data dan memperbaiki kinerjanya sendiri, mirip dengan proses pengambilan keputusan manusia, menggunakan struktur algoritma berlapis yang disebut *artificial neural*

network (ANN).

Deep Learning merupakan bagian dari kecerdasan buatan dan machine learning yang menggunakan neural network dengan multiple layer untuk menyelesaikan berbagai tugas seperti deteksi objek, pengenalan suara, dan terjemahan bahasa. Deep Learning berbeda dari machine learning tradisional karena mampu merepresentasikan data seperti gambar, video, atau teks tanpa memerlukan aturan kode atau pengetahuan domain manusia (Pumsirirat, 2018).

Dalam prosesnya, Deep Learning menggunakan arsitektur jaringan yang terdiri dari berbagai lapisan tersembunyi. Setiap output dari lapisan tersebut dipantau menggunakan grafik khusus untuk setiap neuron output. Transformasi non-linear digunakan untuk mengombinasikan dan merekombinasikan neuron dari semua unit lapisan tersembunyi, sehingga menghasilkan bobot yang optimal untuk mencapai nilai target yang diinginkan (Deng, 2014). Namun, tambahan lapisan dalam jaringan saraf dapat mengurangi efisiensi proses penurunan gradien, yang dapat mempengaruhi output model.

Beberapa jenis algoritma dalam Deep Learning meliputi Convolutional Neural Networks (CNN), yang digunakan untuk memproses data gambar dan mendeteksi objek, serta Recurrent Neural Network (RNN) yang memiliki memori internal dan sering digunakan untuk menganalisis deret waktu dan memproses bahasa alami (Zailani, 2019). Selain itu, ada Long Short Term Memory Network (LSTM) yang merupakan jenis RNN yang dapat mengingat informasi jangka panjang, dan Self-Organizing Maps (SOM) yang memungkinkan pengurangan dimensi data melalui jaringan neural buatan yang mandiri.

Penerapan Deep Learning dalam kehidupan sehari-hari semakin luas, seperti pada smartphone dengan fitur Virtual Assistant yang mengenali suara dan bahasa,

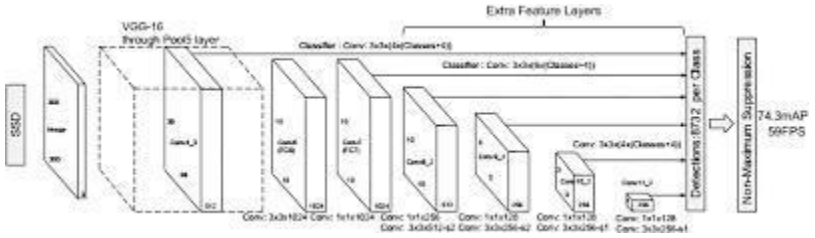
mobil otomatis yang menggunakan teknologi untuk mengemudi, serta chatbot yang digunakan dalam industri customer service. Teknologi ini juga digunakan dalam penerjemahan bahasa, pengenalan biometrik seperti wajah, dan aplikasi lainnya yang memanfaatkan kecerdasan buatan untuk meningkatkan kinerja dan efisiensi.

2.1.3 MobileNet-SSD sebagai Tahap Merancang Sistem Berbasis Deep Learning

Menurut Rachmat Muwardi dan rekan-rekannya (2022), MobileNet adalah sebuah arsitektur jaringan saraf konvolusi (CNN) yang dirancang untuk mengatasi masalah sumber daya komputasi yang berlebihan. Tim peneliti di Google mengembangkan MobileNet sebagai model jaringan yang dapat diimplementasikan pada perangkat ponsel. MobileNetV2 menggunakan konvolusi dalam kedalaman dan pointwise untuk mengekstraksi fitur-fitur yang akan diklasifikasikan. MobileNetV2 juga memperkenalkan dua fitur baru, yaitu kemacetan linear dan koneksi jalan pintas antar bottleneck.

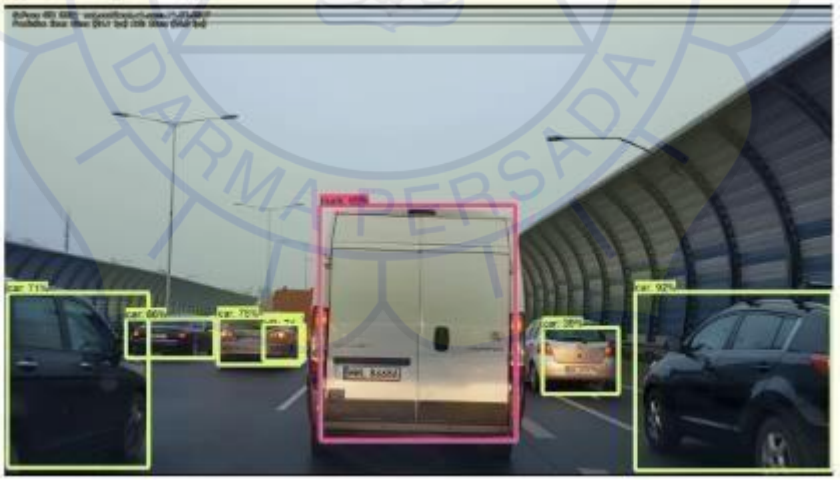
Konsep bottleneck dalam MobileNetV2 merujuk pada input dan output antar model, serta lapisan dalam yang merangkum kemampuan model untuk mentransformasi informasi dari konsep tingkat rendah (seperti piksel) ke konsep tingkat deskriptif yang lebih tinggi.

Single Shot Detector (SSD) adalah metode untuk mendeteksi objek dalam gambar atau video dengan menggunakan satu jaringan saraf mendalam (deep neural network) secara langsung. SSD adalah salah satu algoritma deteksi objek yang paling populer karena menggabungkan akurasi tinggi dengan kecepatan pemrosesan citra yang lebih baik dibandingkan dengan metode lain seperti Faster R-CNN dan YOLOv1 (Sik-Ho Tsang, 2018). Gambar 2 memperlihatkan arsitektur dari Single Shot Detector (SSD).



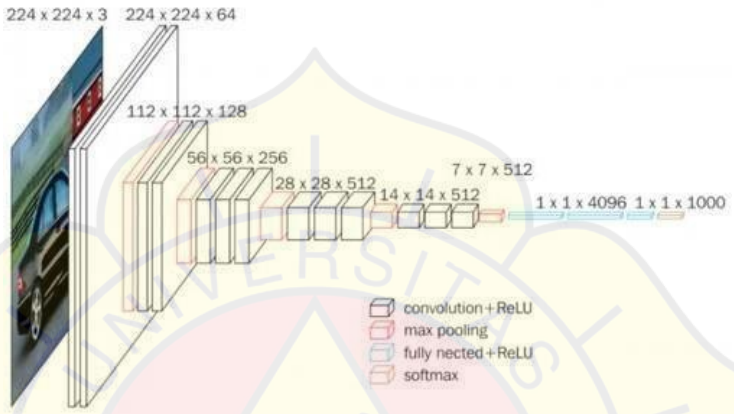
Gambar 2.3 Arsitektur *Single Shot Detector* (SSD)

Single Shot Detector (SSD) berfungsi dengan mendeteksi objek dalam gambar atau video melalui satu proses pemindaian untuk mengidentifikasi area dan objek yang ada di dalamnya. Metode ini berbeda dari Faster R-CNN dan YOLOv1, yang memerlukan dua proses pemindaian untuk mencapai hasil serupa. Sebagai contoh, gambar di atas menunjukkan hasil deteksi objek menggunakan metode Single Shot Detector (SSD).



Gambar 2.4 Contoh deteksi *object* dengan metode *Single Shot Detector* (SSD)

Pada Single Shot Detector (SSD), arsitektur menggunakan VGG-16 untuk ekstraksi fitur. Arsitektur ini terkenal karena kemampuannya dalam mengklasifikasikan gambar dengan resolusi tinggi, yang memungkinkan ekstraksi fitur dari berbagai skala dan secara bertahap mengurangi ukuran input di setiap lapisan berikutnya. Contoh arsitektur VGG-16 dapat dilihat pada Gambar 2.5.



Gambar 2.5 Arsitektur *Visual Geometry Group* (VGG)-16

Convolution Neural Network (CNN) merupakan salah satu jenis Neural Network yang umum digunakan untuk memproses data gambar. CNN dapat digunakan untuk mendeteksi dan mengenali objek dalam sebuah gambar.

Arsitektur CNN terdiri dari dua bagian utama: *Feature Extraction Layer* dan *Convolutional Layer*. Pada bagian Feature Extraction Layer, gambar diubah menjadi representasi numerik yang disebut fitur, yang mempresentasikan gambar dalam bentuk angka-angka. Sementara itu, bagian Convolutional Layer terdiri dari neuron yang tersusun secara terstruktur, membentuk filter yang menangani informasi berdasarkan ukuran dan bentuk piksel gambar. Secara matematis, *Convolutional Layer* atau yang bisa disebut dengan ‘konvolusi’ adalah integral yang mencerminkan jumlah lingkaran dari sebuah sudut fungsi $\boldsymbol{F}$ yang digeser atas fungsi

g, sehingga menghasilkan fungsi ***h***.

Konvolusi dilambangkan dengan arsterik **(*)**, yang ditunjukkan dalam persamaan [1], sehingga ***F*g=h*** berarti fungsi ***F*** dikonvolusikan dengan fungsi g menghasilkan fungsi ***h***. konvolusi dua buah fungsi ***F(x)*** dan ***g(x)*** di definisikan sebagai berikut:

$$h(x) = F(x) * g(x) = \int F(a)g(x-a) \qquad [1]$$

Konvolusi dua buah fungsi ***F(x)*** dan ***g(x)***

Untuk fungsi diskrit, konvolusi didefinisikan sebagai berikut dengan rumus [2]:

$$h(x) = F(x) * g(x) = \sum_{-\infty}^{\infty} F(a)g(x-a) \qquad [2]$$

Konvolusi fungsi diskrit

Pada rumus [3], ***g(x)*** disebut dengan kernel konvolusi (filter). Kernel ***g(x)*** merupakan jendela yang dioperasikan secara bergeser pada sinyal masukan ***F(x)***. hasil konvolusi dinyatakan dengan keluaran ***h(x)***. Contoh, missal citra ***F(x,y)*** yang berukuran 5x5 sebuah kernel dengan 3x3 matriks sebagai berikut:

$$g(x,y)\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} F(x,y)\begin{bmatrix} 4 & 4 & 35 & 4 \\ 6 & 6 & 55 & 2 \\ 5 & 6 & 66 & 2 \\ 6 & 7 & 55 & 3 \\ 3 & 5 & 24 & 4 \end{bmatrix} \qquad [3]$$

Citra (x,y) berukuran 5x5 dan sebuah kernel dengan 3x3 matriks

Secara umum metode *Single Shot Detector* (SSD) mempunyai sebuah rumus sederhana dalam menentukan *default boxes* dan *scale default boxes*, dimana N merupakan jumlah $default\ boxes$, $L_{conf} = loss\ classification$, $L_{loc} = loss\ localization$, $L = prediction\ box\ dan\ g = truth\ ground\ box$. Untuk menentukan *default boxes* bisa mengunakan rumus (4) sebagai acuan:

$$L(x, c, l, g) = \frac{1}{N} L_{conf}(x, c) + L_{loc}(x, l, g) \quad [4]$$

Rumus *default box*

Sedangkan untuk menentukan *scale default boxes* bisa menggunakan rumus (5):

$$S_k = S_{min} \frac{S_{max} - S_{min}}{m-1} (k-1), ke[1, m] \quad [5]$$

Rumus *scale defaults box*

Dimana S_{min} adalah lapisan skala terendah S_{max} adalah lapisan skala tertinggi, dan S_k adalah input pixel.

2.1.4 Pemodelan Sistem UML

Menurut Destriana dan rekan-rekannya (2021), UML adalah bahasa yang digunakan untuk menggambarkan, memvisualisasikan, membuat, dan mendokumentasikan artefak sistem perangkat lunak. Artefak ini meliputi model, deskripsi, atau perangkat lunak yang dihasilkan dalam proses pembuatan perangkat lunak.

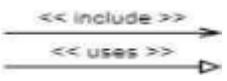
1. *Use case Diagram*

   Diagram use case adalah sebuah model yang mengilustrasikan perilaku sistem yang direncanakan (Sugiarti, 2018). *Diagram use case*

menggambarkan bagaimana aktor-aktor berinteraksi dengan sistem yang sedang dibangun. Dengan menggunakan diagram use case, pengguna dapat dengan mudah melihat fitur-fitur yang tersedia dalam sistem dan siapa yang berhak menggunakan fitur-fitur tersebut.

Simbol-simbol yang digunakan dalam diagram use case perlu dipahami..

Tabel 2.2 Simbol Use Case Diagram

No	Simbol	Deskripsi
1	Use case 	Fungsi yang diekspos oleh sistem sebagai entitas yang bertukar pesan antar entitas atau aktor, biasanya diekspresikan dengan kata kerja di awal frasa kata benda *use case.*
2	Aktor/actor 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat biasanya ditandai dengan, meskipun simbol aktor adalah gambar seseorang, aktor belum tentu kata benda pribadi di awal frase kata benda aktor.
3	Asosiasi/*association*	Komunikasi antara Aktor dan *Use Case* Partisipan dalam *Use Case* berinteraksi dengan Aktor.
4	Ektensi/*extend* 	Hubungan *use case* tambahan dengan *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri tanpa *use case* tambahan, seperti prinsip pewarisan pemrograman berorientasi objek, *use case* tambahan biasanya memiliki nama depan yang sama dengan *use case* yang ditambahkan.
5	*Include* 	Hubungan *use case* tambahan ke *use case* ketika *use case* yang ditambahkan membutuhkan *use case* tersebut untuk melakukan

		tugasnya atau sebagai syarat untuk kinerja *use case* tersebut. Ada dua aspek utama untuk menyertakan kasus penggunaan. Sertakan berarti bahwa *use case* yang ditambahkan akan dipanggil setiap kali *use case* yang ditambahkan dieksekusi.

2. *Activity Diagram*

Menurut Rosa & Shalahuddin (2018), diagram aktivitas atau diagram tindakan mengilustrasikan urutan langkah atau operasi dari suatu sistem, proses bisnis, atau menu perangkat lunak. Diagram ini merencanakan fungsi-fungsi yang dapat dilakukan oleh sistem atau proses tersebut.

Tabel 2.3 Simbol Activity Diagram

No	Simbol	Deskripsi
1	Status Awal 	Keadaan awal operasi sistem, diagram operasi memiliki keadaan awal.
2	Aktivasi 	Tindakan yangdilakukan oleh sistem. Tindakan biasanya dimulai dengan kata kerja.
3	Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4	Penggabungan/*join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5	Status Akhir 	Keadaan akhir yang dijalankan sistem, diagram fungsional memiliki keadaan akhir

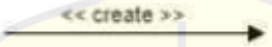
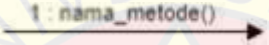
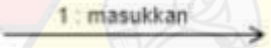
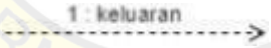
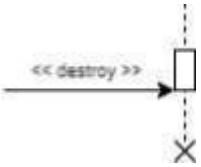
6	*Swimlane* 	Memisahkan aktivitas yang bertanggung jawab terhadap aktivitas yang terjadi dengan *swimlane*.

3. *Sequence Diagram*

Sequence diagram mengilustrasikan perilaku objek dalam suatu skenario penggunaan dengan menunjukkan siklus hidup objek dan pesan yang dikirim serta diterima antar objek. Oleh karena itu, untuk membuat sequence diagram, penting untuk mengetahui objek-objek yang terlibat dalam use case, serta metode-metode yang dimiliki oleh kelas-kelas yang digunakan oleh objek-objek tersebut. Pembuatan sequence diagram bertujuan untuk memvisualisasikan skenario yang terjadi pada use case tersebut (Rosa dan Shalahuddin, 2018).

Tabel 2.4 Simbol Sequence Diagram

No	Simbol	Deskripsi
1	Aktor 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, sehingga meskipun simbol aktor adalah gambar manusia, belum tentu aktor adalah orang yang menggunakan kata benda.
2	Garis Hidup/ *Lifeline* 	Menyatakan kehidupan suatu objek.

3	Objek 	Menyatakan objek yang berinteraksi pesan.
4	Waktu Aktif 	Mendeklarasikan objek aktif dan dapat berinteraksi, segala sesuatu yang berhubungan dengan waktu aktif itu adalah langkah yang dilakukan di dalamnya.
5	Pesan Tipe *Create* 	Artinya suatu objek menciptakan objek lain, arah panah menunjuk ke objek yang dibuat.
6	Pesan Tipe *Call* 	Menyatakan bahwa suatu objek memanggil fungsi/metode pada objek lain atau pada dirinya sendiri.
7	Pesan Tipe *Send* 	Menandakan bahwa objek sedang mengirim data/input/informasi ke objek lain, panah menunjuk ke objek yang dikirim.
8	Pesan Tipe *Return* 	Untuk menunjukkan bahwa suatu objek yang melakukan operasi atau metode menghasilkan pengembalian ke objek yang ditentukan, panah menunjuk ke objek yang menerima Pengembalian.
9	Pesan Tipe Destory 	Menunjukkan bahwa suatu objek mengakhiri kehidupan objek lain, tanda panah menunjukkan objek yang akan diakhiri, sebaiknya jika ingin dibuat, maka harus dihancurkan

2.1.5 Software dan Pemrograman Terkait

2.1.5.1 Jupyter

Penggunaan aplikasi Jupyter Notebook, yang diluncurkan pada tahun 2015, telah menjadi penting dalam analisis data. Ini menjadi alat utama bagi ilmuwan data untuk membuat dan berbagi kode, hasil perhitungan, visualisasi, dan narasi komputasi. Jupyter Notebook menggabungkan kelebihan dari bahasa pemrograman Julia (Ju), Python (Py), dan R, serta mendukung kolaborasi dengan aplikasi lain seperti Google Drive untuk analisis dan penyajian data yang mudah. Aplikasi ini juga memfasilitasi pembuatan software berbasis web dengan berbagai bahasa pemrograman, memungkinkan programmer untuk berinovasi dan membuat aplikasi yang bermanfaat. Kelebihan lainnya adalah kemampuannya untuk digunakan dalam kolaborasi dinamis, seperti dalam analisis data penjualan perusahaan, yang membutuhkan keputusan yang valid dan akurat untuk menghindari dampak negatif bagi perusahaan. Jupyter Notebook menjadi pilihan yang populer di perusahaan besar karena kemampuan analisisnya yang dapat dipercaya dan mudah dimengerti.

2.1.5.2 *Library* Flask

Menurut Rahadian Irsyad (2018), Flask adalah sebuah web framework yang ditulis dalam bahasa Python dan diklasifikasikan sebagai microframework. Flask berfungsi sebagai kerangka kerja untuk membangun aplikasi web dengan struktur yang terorganisir dan mengatur perilaku aplikasi web dengan lebih mudah menggunakan bahasa Python.

Flask termasuk dalam kategori microframework karena tidak memiliki alat atau pustaka tertentu yang terpasang secara bawaan. Fungsi umum seperti validasi form, database, dan sebagainya dapat diintegrasikan menggunakan ekstensi yang

disediakan oleh pihak ketiga, sehingga memungkinkan Flask untuk tetap sederhana namun fleksibel dalam pengembangan aplikasi. Meskipun disebut sebagai microframework, Flask tetap mampu menyediakan fitur yang lengkap dan dapat dengan mudah diperluas. Fitur bawaan Flask termasuk server pengembangan bawaan, debugger cepat, dukungan terintegrasi untuk pengujian unit, kompatibilitas dengan mesin aplikasi Google, dispatching request RESTful, templating Jinja2, dukungan untuk cookies yang aman, dan berbasis Unicode sesuai dengan standar WSGI 1.0.

Flask juga dikenal memiliki dokumentasi yang sangat baik dan komunitas yang aktif di internet, yang mendukung pengguna untuk memecahkan masalah dan berbagi pengalaman terkait pengembangan dengan Flask.

Keunggulan Flask antara lain adalah ringan, modular, mudah dalam penanganan HTTP request, memiliki API yang baik, dokumentasi yang lengkap, mudah untuk dipasang dan di-deploy, mudah untuk diuji (unit testability), dan sangat fleksibel dalam konfigurasi. Namun, Flask juga memiliki kekurangan seperti tidak memiliki ORM dan layer database bawaan, serta tidak mendukung secara native asinkronisitas. Ketergantungan Flask pada aplikasi dari pihak ketiga untuk beberapa fitur juga dapat menjadi tantangan tersendiri dalam penggunaannya.

2.1.5.3 *Library* PyTorch

PyTorch adalah library yang dikembangkan oleh Facebook AI (sekarang Meta AI) pada tahun 2016, dan digunakan untuk machine learning, deep learning, serta natural language processing. Pada dasarnya, PyTorch adalah library terkompilasi atau objek bersama yang berisi fungsi-fungsi yang dapat dieksekusi secara native di berbagai sistem operasi. Format file yang digunakan bervariasi sesuai dengan sistem operasi, seperti .dll untuk Windows dan .so untuk Linux. Meskipun

dikembangkan menggunakan C++, PyTorch menyediakan API frontend berbasis Python yang sangat memudahkan penggunaannya.Selain frontend API Python, PyTorch juga memiliki frontend API C++ yang disebut LibTorch, yang digunakan dalam proyek-proyek yang tidak bergantung pada lingkungan Python. Misalnya, dalam bahasa pemrograman Rust, penggunaan PyTorch dapat dilakukan melalui wrapper bernama tch-rs (https://github.com/LaurentMazare/tch-rs). Model yang dapat dimuat pada LibTorch disebut Torchscript, yang merupakan kumpulan fungsi yang dapat dieksekusi tanpa ketergantungan pada lingkungan Python, cocok untuk aplikasi seperti proyek berbasis C++ atau Rust (seperti yang dijelaskan oleh Arsil Qodryantha, 2023).

2.1.5.4 *Library* TensorFlow

Menurut B. Setiawan (2021), Tensorflow merupakan sebuah library sumber terbuka yang dikembangkan oleh Google Brain untuk pembelajaran mesin dan jaringan saraf. Tensorflow memiliki kemampuan untuk menjalankan berbagai algoritma dalam machine learning. Tujuan utama pengembangan Tensorflow adalah untuk keperluan riset, pemrograman, dan pengembangan dalam bidang pembelajaran mesin. Library ini mendukung beberapa bahasa pemrograman seperti Python, Java, dan C/C++ (Rucci & Casile, 2015).

Google juga mengembangkan Tensorflow Lite, sebuah versi Tensorflow yang dirancang untuk berjalan di perangkat kecil atau perangkat seluler. Tensorflow Lite menyediakan solusi bagi pengguna yang ingin menjalankan model pembelajaran mesin pada perangkat mobile. Untuk menggunakan Tensorflow Lite, diperlukan proses konversi dari model Tensorflow ke format Tensorflow Lite. Mode Tensorflow Lite dapat diimplementasikan pada perangkat Android dan iOS.