

DETEKSI PENGENDARA MOTOR BERHELM DAN TIDAK BERHELM
BERBASIS ALGORITMA ZFNET DAN MOBILENET
SINGLE SHOT DETECTOR

SKRIPSI

JONATHAN SIMANJUNTAK

171402106



PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024

DETEKSI PENGENDARA MOTOR BERHELM DAN TIDAK BERHELM
BERBASIS ALGORITMA ZFNET DAN MOBILENET
SINGLE SHOT DETECTOR

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah Sarjana
Teknologi Informasi

JONATHAN SIMANJUNTAK

171402106



PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024

PERSETUJUAN

Judul : Deteksi Pengendara Motor Berhelm dan tidak Berhelm Berbasis Algoritma ZFNet dan MobileNet Single Shot Detector

Kategori : *Computer Vision*

Nama : Jonathan Simanjuntak

Nomor Induk Mahasiswa : 171402106

Program Studi : Teknologi Informasi

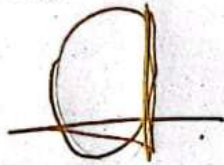
Fakultas : Ilmu Komputer dan Teknologi Informasi
Universitas Sumatera Utara

Komisi Pembimbing :

Medan, 4 Juli 2024

Komisi Pembimbing:

Pembimbing 2



Ainul Hizriadi S.Kom., M.Sc.
NIP. 198510272017061001

Pembimbing 1



Fahrurrozi Lubis B.IT., M.Sc.IT
NIP. 198610122018052001

Diketahui/disetujui oleh
Program Studi Teknologi Informasi
Ketua,



Dedy Arisandi S.T., M.Kom.
NIP. 197908312009121002

PERNYATAAN**DETEKSI PENGENDARA MOTOR BERHELM DAN
TIDAK BERHELM BERBASIS ALGORITMA ZFNET
DAN MOBILENET SINGLE SHOT DETECTOR****SKRIPSI**

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing – masing telah disebutkan sumbernya

Medan, 4 Juli 2024



Jonathan Simanjuntak

171402106

UCAPAN TERIMA KASIH

Penulis memanjatkan puji dan syukur kehadirat Tuhan Yang Maha Esa, karena berkat kesehatan dan rahmat yang dilimpahkan, penulis dapat menyelesaikan penyusunan skripsi yang berjudul “Deteksi Pengendara Motor Berhelm dan Tidak Berhelm Berbasis Algoritma Zfnet dan Single Shot Detector Mobilenet” sebagai salah satu syarat untuk memperoleh gelar Sarjana di Program Studi S1 Teknologi Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara. Dalam proses penyusunan skripsi ini tidak terlepas dari berbagai kendala yang telah dihadapi. Penyusunan skripsi ini tidak lepas dari bantuan dan dukungan dari beberapa pihak baik secara langsung maupun tidak langsung. Pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Keluarga Penulis, Ayah Joslan Simanjuntak, Ibu Karolina karokaro, Kakak Pebriani Simanjuntak serta Adik Pagar Simanjuntak yang selalu memberikan dukungan, doa, semangat senantiasa kepada penulis, dan juga mau menjadi bagian dari hasil pengujian didalam skripsi ini.
2. Bapak Dr. Muryanto Amin, S.Sos., M.Si selaku Rektor Universitas Sumatera Utara.
3. Ibu Dr. Maya Silvi Lydia, M.Sc selaku Dekan Fasilkom-TI USU.
4. Bapak Dedy Arisandi S.T., M.Kom selaku Ketua Program Studi S1 Teknologi Informasi Universitas Sumatera Utara.
5. Bapak Fahrurrozi Lubis B.IT, M.Sc.IT selaku Dosen Pembimbing I dan juga Bapak Ainul Hizriadi S.kom, M.SC selaku Dosen Pembimbing II yang telah meluangkan waktunya untuk memberikan bimbingan, petunjuk, dan semangat dalam proses penyelesaian skripsi ini.
6. Bapak/Ibu dosen penguji.
7. Seluruh Dosen Program Studi S1 Teknologi Informasi yang telah memberikan ilmu selama perkuliahan.
8. Para staff dan pegawai Fasilkom-TI Universitas Sumatera Utara yang telah membantu dalam segala urusan administrasi di masa perkuliahan dan penyelesaian skripsi.

9. Kakak/abang senior yang memberikan saran, kritik, maupun masukan dan bantuan kepada penulis selama masa perkuliahan maupun selama masa pengerjaan skripsi.
10. Teman-teman angkatan 2017 yang telah memberikan dukungan dan semangat selama masa perkuliahan dan juga selama masa pengerjaan skripsi yang tidak dapat penulis sebutkan satu - persatu.
11. Teman-teman di Kom A yang tidak dapat penulis sebutkan satu-persatu.
12. Para sahabat di Kom A, yaitu Egi Sebayang, Arlin Telaumbanua, Vincent Simanjuntak, Allen Tampubolon, dan Moris Martua, yang selalu memberikan dukungan, doa, dan semangat dimasa perkuliahan sampai selesai penyusunan skripsi ini.
13. Para sahabat DEKKE yaitu Yosie, Inez, Sarah, yang telah mendukung pengerjaan skripsi ini dan juga menemani waktu senggang penulis.
14. Anggota GULLY yaitu Frans, Riyo, Maulana, Syarfan, Irfan, Josua, Moris, Rico, Yonadab, Dicky, Alfi dan Yusman yang membantu penyelesaian skripsi.
15. Para penghuni KOS CIPTA yaitu Allen, Farras, Grace, Ciya, Moris, Riyo dan Frans yang telah membantu menyelesaikan skripsi.
16. Semua pihak yang terlibat langsung maupun tidak langsung yang tidak dapat penulis sebutkan satu-persatu yang telah membantu dalam penyelesaian skripsi ini

ABSTRAK

Dalam upaya meningkatkan keselamatan berkendara, pengembangan sistem deteksi pengendara motor berhelm dan tidak berhelm menjadi penting. Studi ini bertujuan untuk mengimplementasikan algoritma ZFNet dan MobileNet SSD untuk membedakan pengendara motor yang menggunakan helm dan yang tidak menggunakan helm berdasarkan analisis citra. Metode ini mengandalkan pendekatan *deep learning* untuk mengekstraksi fitur dari citra pengendara motor dan melakukan deteksi berdasarkan keberadaan helm. Evaluasi kinerja dilakukan menggunakan dataset citra yang bervariasi. Hasil eksperimen menunjukkan akurasi yang memuaskan, dengan ZFNet dan MobileNet SSD mencapai akurasi sebesar 86%. Sistem ini memiliki potensi untuk diimplementasikan dalam sistem keamanan lalu lintas untuk mendukung upaya pencegahan kecelakaan berkendara yang disebabkan oleh penggunaan helm yang tidak sesuai.

Kata Kunci: *ZFNet, MobileNet SSD, Pengendara, Helm, Deteksi*

*DETECTION OF MOTORCYCLISTS WITH HELMETS
AND NO HELMET BASED ALGORITHM
ZFNET AND MOBILENET SSD*

ABSTRACT

In an effort to improve driving safety, the development of a detection system for helmeted and unhelmeted motorcyclists is important. This study aims to implement the ZFNet and MobileNet SSD algorithms to distinguish helmeted and unhelmeted motorcyclists based on image analysis. The method relies on a deep learning approach to extract features from motorcyclist images and perform classification based on the presence of helmets. Performance evaluation is performed using a variety of image datasets. Experimental results show satisfactory accuracy, with ZFNet and MobileNet SSD achieving 86% accuracy. This system has the potential to be implemented in traffic safety systems to support efforts to prevent riding accidents caused by improper helmet use.

Keywords: ZFNet, MobileNet SSD, Rider, Helmet, Detecion

DAFTAR ISI

PERNYATAAN	ii
UCAPAN TERIMA KASIH	iii
ABSTRAK	v
<i>ABSTRACT</i>	vi
DAFTAR ISI	vii
DAFTAR TABEL	x
DAFTAR GAMBAR	xi
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	3
1.3. Batasan Masalah	3
1.4. Tujuan Penelitian	3
1.5. Manfaat Penelitian	3
1.6. Metodologi Penelitian	4
1.7. Sistematika Penulisan	5
BAB 2 TINJAUAN PUSTAKA	6
2.1. Streaming Video	6
2.2. Pengertian Citra Digital	8
2.2.1. Representasi Citra Digital	9
2.3. Deteksi Tepi	11
2.3.1. Operator Sobel	13
2.4. <i>Convolutional Neural Network (CNN)</i>	14
2.4.1. Konsep CNN	15
2.4.2. Arsitektur Jaringan CNN	16

2.4.3.	Arsitektur ZFNet	19
2.5.	MobileNet Single Shot Detector	20
2.6.	OpenCV	20
2.7.	Mengukur Kinerja Metode dengan <i>Confusion Matrix</i>	22
2.8.	Penelitian Terkait	24
BAB 3 ANALISIS DAN PERANCANGAN		28
3.1.	Data yang Digunakan	28
3.2.	Arsitektur Umum	29
3.2.1.	Pengumpulan Dataset	30
3.2.2.	<i>Preprocessing</i>	30
3.2.3.	<i>Labelling</i>	33
3.2.4.	<i>Splitting Data</i>	33
3.2.5.	<i>Convert to CSV</i>	34
3.2.6.	<i>Create TFRecord file</i>	34
3.2.7.	<i>Create Pipeline Config</i>	35
3.2.8.	<i>Training</i>	35
3.2.9.	Proses Deteksi	35
3.2.10.	Proses Deteksi	36
3.2.11.	<i>Output</i>	37
3.3.	Perancangan Antarmuka Sistem	37
BAB 4 IMPLEMENTASI DAN HASIL PENELITIAN		40
4.1.	Spesifikasi Kebutuhan Aplikasi	40
4.1.1.	Kebutuhan Perangkat Keras	40
4.1.2.	Kebutuhan Perangkat Lunak	40
4.2.	Implementasi Sistem	41
4.3.	Hasil Uji Sistem	44
BAB 5 KESIMPULAN DAN SARAN		65

5.1. Kesimpulan	65
5.2. Saran	65
DAFTAR PUSTAKA	66

DAFTAR TABEL

Tabel 2.1 Deteksi <i>confusion matrix</i>	23
Tabel 2.2 Penelitian Terdahulu	24
Tabel 4.1 Hasil Uji Sistem	45
Tabel 4.2 Hasil Seluruh Percobaan	59

DAFTAR GAMBAR

Gambar 2.1 Konfigurasi Perangkat Streaming Video (Kurniawati, 2019)	7
Gambar 2.2 Sistem Koordinat yang Dipergunakan untuk Mewakili Citra	10
Gambar 2.3 Matriks citra	10
Gambar 2.4 Proses deteksi tepi citra (Liantoni, 2015)	11
Gambar 2.5 Model tepi satu citra (Liantoni, 2015)	12
Gambar 2.6 Jenis – jenis tepi citra (Panetta <i>et al.</i> , 2015)	12
Gambar 2.7 Matriks konvolusi 3 x 3 Sobel (Israni & Jain, 2016)	13
Gambar 2.8 Matriks Operator Sobel (Gonzales, 2003)	13
Gambar 2.9 Hasil deteksi tepi	14
Gambar 2.10 Arsitektur MLP sederhana (Zapletal, 2017)	15
Gambar 2.11 Proses konvolusi pada CNN (Zapletal, 2017)	16
Gambar 2.12 Operasi Konvolusi (Syafeeza, 2014)	17
Gambar 2.13 Operasi max pooling (Syafeeza, 2014)	18
Gambar 2.14 Arsitektur ZFNet (Zeiler, <i>et al.</i> 2014)	19
Gambar 2.15 Arsitektur SSD	20
Gambar 2.16 Deteksi Haar digunakan dalam openCV (Cahu <i>et al.</i> , 2019)	21
Gambar 2.17 OpenCV digunakan untuk <i>object detection</i>	22
Gambar 3.1 Citra pengendara sepeda motor tanpa helm	28
Gambar 3.2 Citra pengendara sepeda motor menggunakan helm	28
Gambar 3.3 Arsitektur Umum	29
Gambar 3.4 Dataset yang telah diambil	30
Gambar 3.5 Proses <i>resizing</i>	31
Gambar 3.6 Proses <i>Grayscale</i>	32
Gambar 3.7 Proses pelabelan citra	33

Gambar 3.8 Potongan kode <i>split dataset</i>	34
Gambar 3.9 Tampilan Antarmuka Sistem (<i>Home</i>)	37
Gambar 3.10 Tampilan Video	38
Gambar 3.11 Tampilan <i>Live Camera</i>	38
Gambar 3.12 Tampilan <i>Information</i>	39
Gambar 4.1 Tampilan program (<i>Home</i>)	41
Gambar 4.2 Tampilan program (<i>Video</i>)	42
Gambar 4.3 Tampilan program (<i>Live Camera</i>)	43
Gambar 4.4 Tampilan program (<i>Information</i>)	44

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Helm adalah alat pelindung kepala yang berfungsi melindungi kepala dari benturan sehingga kepala tidak mengalami cedera. Beberapa kegiatan manusia yang menggunakan helm untuk melindungi kepala diantaranya adalah mengendarai kendaraan bermotor roda dua, melakukan pekerjaan konstruksi, beberapa kegiatan olahraga dan lain sebagainya. Helm sebagai perlengkapan kendaraan bermotor berbentuk topi pelindung kepala yang berfungsi untuk melindungi kepala apabila terjadi benturan. Helm terbagi menjadi 2 jenis yaitu *Open Face* yaitu bentuk helm yang menutup kepala sampai dengan bagian leher dan menutup depan telinga dan *Full Face* dimana bentuk helm yang menutup kepala atas, bagian leher, dan bagian mulut), yang memiliki bagian keras dan halus yang merupakan bagian paling luar dari helm dan bagian dalam yang dipasang untuk menyerap energi benturan, serta bagian muka helm yang dapat melindungi sebagian atau seluruh bagian muka dan terbuat dari lapisan bening (Lanes *et al.*, 2019).

Kesadaran akan pentingnya menggunakan helm sebagai perlengkapan keselamatan dalam berkendara semakin meningkat dalam masyarakat modern. Helm tidak hanya dianggap sebagai kepatuhan terhadap aturan lalu lintas, tetapi juga sebagai langkah proaktif untuk melindungi diri sendiri dari potensi cedera serius akibat kecelakaan. Berdasarkan data dari Kementerian Perhubungan, kecelakaan sepeda motor menjadi salah satu penyebab utama kematian di jalan raya, dan dalam banyak kasus, korban kecelakaan yang meninggal tidak mengenakan helm.

Penggunaan helm tidak hanya penting untuk melindungi kepala dari cedera, tetapi juga secara luas diakui sebagai salah satu langkah yang paling efektif dalam mengurangi risiko cedera serius atau fatal dalam kecelakaan sepeda motor. Helm dapat mengurangi dampak benturan pada kepala, meminimalkan risiko trauma kepala dan cedera otak,

serta meningkatkan peluang untuk bertahan hidup dan pulih sepenuhnya dari kecelakaan.

Di sisi lain, meskipun pentingnya penggunaan helm telah dipahami dengan baik, masih ada tantangan dalam memastikan kepatuhan pengguna jalan terhadap aturan tersebut. Faktor-faktor seperti kurangnya kesadaran, resistensi terhadap aturan, dan keengganan untuk mengenakan helm dalam situasi tertentu masih menjadi masalah yang harus diatasi. Berdasarkan data milik Kementerian Perhubungan, tercatat sebanyak 5 ribu orang tewas dalam kecelakaan sepeda motor akibat pengendara tidak mengenakan helm (Kementerian Perhubungan. 2020).

Pada penelitian Lanes *et al.*, 2019 yang berjudul Perancangan Dan Implementasi Aplikasi Sistem Pemantauan Helm Pendeteksi Kantuk Berbasis Website dilakukan pemantauan penggunaan helm. Sistem ini menggunakan sebuah chip yang dipasang pada satu helm yang akan dikenakan dan telah di modifikasi sedemikian rupa, sehingga saat helm dikenakan maka akan mengirimkan sinyal ke sepeda motor dan motor dapat dihidupkan, lalu ketika helm dilepas saat keadaan motor menyala, maka otomatis motor akan mati. Namun sistem tersebut hanya dapat digunakan dengan menggunakan satu helm saja, maka dari itu dilakukan pengembangan pada sistem tersebut dengan menggunakan Pengolahan Citra Digital serta library pendukung seperti OpenCV. Dari hasil pengujian sistem ini dinyatakan bahwa sistem mampu mendeteksi dengan akurasi yang cukup tinggi yaitu sebesar 93,7%. Secara keseluruhan dapat dinyatakan bahwa sistem ini dapat bekerja sesuai dengan perencanaan awal yaitu untuk dapat mendeteksi apakah pengendara mengenakan helm atau tidak, sehingga motor hanya dapat dihidupkan apabila pengendara mengenakan helm serta tujuan dari pembuatan sistem ini tercapai yaitu untuk mengurangi angka pelanggaran serta kecelakaan karena pengendara yang lalai melengkapi perlengkapan berkendara terutama helm.

Ramadhan, M. P. (2023) Pada penelitiannya yang berjudul *Deteksi Objek Robot Sepakbola secara Real Time dengan Mobilenet Single Shot Detector (SSD)* menggunakan algoritma SSD dan menghasilkan 90% akurasi pada model yang dibuatnya.

Pada penelitian Widodo *et al.* (2021) dengan judul Deteksi Pemakaian Helm Proyek Dengan Metode Convolutional Neural Network dimana dilakukan deteksi helm proyek untuk keselamatan pekerja. Proses deteksi pada sistem ini melakukan lokalisasi dan deteksi dengan sekali langkah proses sehingga hasil dari proses deteksi ini adalah

orang menggunakan helm proyek dan orang tidak menggunakan helm proyek. Pengujian sistem deteksi dilakukan secara individu maupun kelompok maksimal 5 orang dengan F1-score yang diperoleh sebesar 0,79.

Pada penelitian Sthevanie *et al.* (2020) yang berjudul Deteksi Helm pada Video Pengendara Sepeda Motor menggunakan Ekstraksi Ciri Histogram of Oriented Gradients dilakukan pendeteksian helm. Pada penelitian ini dibuat sebuah sistem yang dapat mendeteksi helm pada pengendara sepeda motor secara otomatis menggunakan ekstraksi fitur Histogram of Oriented Gradient. Sistem yang dibangun telah dapat mendeteksi pengendara yang menggunakan helm dan tidak menggunakan helm pada 100 citra yang diekstrak dari video dengan nilai f-measure sebesar 90.67% menggunakan metode ekstraksi fitur HOG dengan kondisi ukuran cell 8x8 piksel dan jumlah 9 bins dengan sudut 180° serta algoritma deteksi SVM dengan kernel polynomial derajat 3.

Sebagai tanggapan terhadap tantangan ini, berbagai upaya telah dilakukan untuk meningkatkan kepatuhan pengguna helm. Dari pengembangan sistem pemantauan otomatis hingga kampanye kesadaran masyarakat, berbagai pendekatan telah diambil untuk mengurangi angka kecelakaan yang disebabkan oleh ketidakpatuhan dalam menggunakan helm.

Dalam konteks ini, penelitian ini bertujuan untuk menyumbangkan kontribusi dalam pengembangan sistem deteksi otomatis yang dapat memantau penggunaan helm secara efisien dan akurat. Melalui penggunaan teknologi pengolahan citra digital dan algoritma kecerdasan buatan, penelitian ini bertujuan untuk merancang sistem yang dapat mendeteksi apakah pengendara mengenakan helm atau tidak melalui input video streaming, dengan harapan dapat meningkatkan kepatuhan pengguna helm dan mengurangi angka kecelakaan yang disebabkan oleh ketidakpatuhan tersebut.

Berdasarkan permasalahan tersebut diatas, maka dalam penelitian ini penulis bertujuan untuk merancang sebuah sistem yang mampu mendeteksi helm dengan input streaming video melalui camera dengan judul Deteksi Penggunaan Helm menggunakan Algoritma Convolutional Neural Network (CNN) berbasis Arsitektur ZfNet dan MobileNet SSD.

1.2. Rumusan Masalah

Pemantauan helm pada pengendara sepeda motor di jalan raya masih sulit dilakukan, hal ini diakibatkan oleh kurangnya petugas yang mengawasi kepatuhan berkendara. Sehingga dibutuhkan sebuah sistem yang dapat memantau penggunaan helm pada pengendara sepeda motor dengan akurat.

1.3. Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah:

1. Data yang digunakan untuk mendeteksi helm adalah streaming video melalui camera digital.
2. Penelitian ini dilakukan pada jarak maksimal 5 meter.
3. Penelitian ini menggunakan algoritma Mobilenet Single Shot Detector (SSD) dan Convolutional Neural Network (CNN) berbasis Arsitektur ZfNet.
4. Output sistem berupa helm, no_helm, dan motor.
5. Penelitian ini dikhususkan mendeteksi objek pada siang hari.
6. Penelitian ini mengabaikan situasi lingkungan seperti curah hujan, kabut, dan asap.

1.4. Tujuan Penelitian

Tujuan dalam penelitian ini adalah untuk menghitung kinerja algoritma CNN berbasis arsitektur ZFNet dan MobileNet SSD dalam mendeteksi penggunaan helm.

1.5. Manfaat Penelitian

Manfaat yang didapatkan dari penelitian ini adalah sebagai berikut:

1. Mempercepat pendeteksian helm pada pengendara sepeda motor berbasis streaming video.
2. Mempermudah pemantauan penggunaan helm pada sepeda motor sebagai sarana penegakan hukum di jalan raya.

1.6. Metodologi Penelitian

Adapun metodologi pada penelitian ini adalah sebagai berikut:

1. Studi Literatur

Pada tahap ini dilakukan studi kepustakaan yaitu proses mengumpulkan bahan referensi mengenai algoritma CNN dan SSD Mobilenet serta pengolahan citra dari berbagai buku, jurnal, artikel, dan beberapa referensi lainnya.

2. Analisis

Pada tahap ini dilakukan analisis terhadap studi literatur untuk mengetahui dan mendapatkan pemahaman mengenai kombinasi algoritma CNN dan SSD Mobilenet pada masalah pendeteksian helm.

3. Perancangan

Pada tahap perancangan sistem dilakukan perancangan arsitektur, pengumpulan data, pelatihan dan merancang antarmuka. Proses perancangan dilakukan berdasarkan hasil analisis studi literatur yang telah didapatkan.

4. Implementasi

Pada tahap implementasi dilakukan perancangan sistem deteksi helm berbasis desktop

5. Pengujian

Pada tahap ini dilakukan pengujian aplikasi yang telah dibuat guna memastikan aplikasi telah berjalan sesuai dengan apa yang diharapkan.

6. Dokumentasi

Pada tahap ini, penelitian yang telah dilakukan, didokumentasikan mulai dari tahap analisis sampai kepada pengujian dalam bentuk skripsi.

1.7. Sistematika Penulisan

Sistematika dalam penulisan skripsi ini akan dibagi menjadi beberapa bab sebagai berikut:

BAB 1 : PENDAHULUAN

Pada bab ini diuraikan hal-hal yang merupakan latar belakang masalah, pembuatan rumusan masalah, pembatasan masalah yang ada, pembuatan tujuan dan manfaat dari penelitian, metodologi penelitian yang dilakukan serta sistematika penulisan.

BAB 2 : LANDASAN TEORI

Berisi dasar teori citra digital, deteksi tepi, pendeteksian objek serta algoritma CNN.

BAB 3 : ANALISIS DAN PERANCANGAN

Dari Bab ini berisikan analisis terhadap pengolahan awal citra, deteksi helm dengan kombinasi algoritma CNN serta pemodelan sistem yang akan dirancang.

BAB 4 : IMPLEMENTASI DAN PENGUJIAN

Berisi bentuk jadi dari sistem yang telah dibangun berupa tampilan-tampilan serta hasil pengujian terhadap data serta nilai akurasi pengenalan.

BAB 5 : KESIMPULAN DAN SARAN

Bab ini berisi pemaparan hasil kesimpulan dari hasil pengujian yang telah dilakukan serta saran-saran untuk perbaikan dan pengembangan lebih lanjut.

BAB 2

TINJAUAN PUSTAKA

2.1. Streaming Video

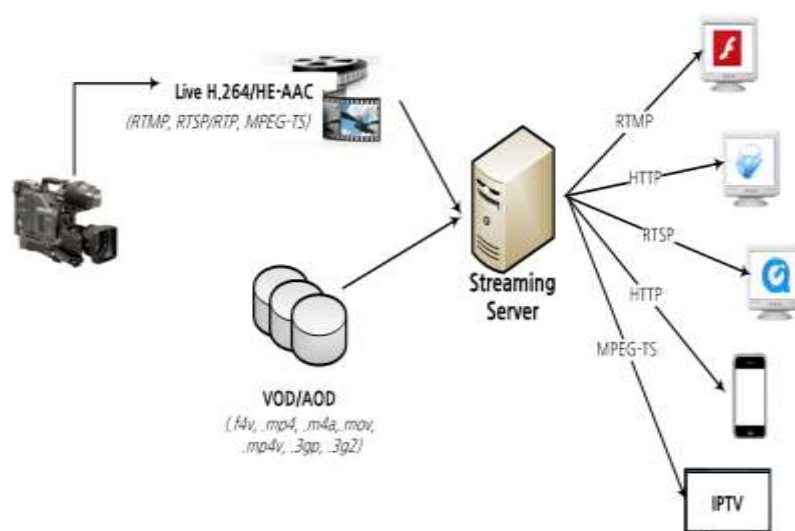
Pada era media digital seperti sekarang ini, ketika melihat konten yang ada di media sosial dapat dengan mudah memilih konten tersebut dan secara otomatis akan dimainkan, tanpa perlu melakukan pengunduhan terlebih dahulu. Streaming adalah proses pengiriman konten baik audio atau video yang dikirim dalam bentuk yang sudah terkompres melalui internet yang kemudian dimainkan secara langsung tanpa harus melakukan pengunduhan terlebih dahulu. Definisi lain streaming adalah sebagai proses mengalirkan atau mentransfer data dari server kepada host dimana data yang ditransfer merepresentasikan informasi yang harus disampaikan secara langsung. Data yang dikirimkan melalui proses ini berupa video, audio, grafik, web tours atau aplikasi real time lain (Christyono et al., 2017).

Dari dua definisi tadi, dapat dikatakan bahwa streaming adalah proses transfer data atau informasi dari satu pengguna ke pengguna lain, baik secara langsung atau melalui aplikasi tertentu, yang sifatnya tidak perlu melakukan pengunduhan dan akan secara langsung ditampilkan untuk data yang sudah berhasil dipindahkan. Jadi ketika menggunakan media sosial seperti YouTube atau Instagram, tidak perlu melakukan pengunduhan untuk menyaksikan video yang ada pada platform tersebut. Cukup dengan mengklik video yang akan disaksikan, dan secara langsung Anda akan dapat menikmati video tersebut secara bertahap sesuai dengan kecepatan transfer data yang dimiliki.

Streaming sendiri kini juga tak hanya dimanfaatkan pada media baru seperti YouTube dan Instagram, melainkan juga radio-radio modern yang mengandalkan jaringan internet untuk mengudara. Tidak sedikit yang kemudian memanfaatkan keberadaan radio online ini untuk mencari informasi berupa berita terkini atau sekedar konten hiburan yang dapat menemani ketika di perjalanan. Jika dilihat dari jenisnya,

setidaknya ada 2 macam streaming yang digunakan oleh masyarakat saat ini. Pertama adalah Prerecord Streaming dan kedua adalah Live Streaming. Tentu terdapat perbedaan pada kedua jenis streaming ini (Kurniawati, L. 2019).

Jenis yang pertama ini dijalankan dengan terlebih dahulu merekam video atau konten yang akan disiarkan, kemudian menyimpannya pada media tertentu (bisa media online atau media fisik seperti hardisk). Setelah disimpan, baru kemudian host dapat melakukan akses pada konten tersebut jika akses diberikan secara bebas oleh pemilik konten.



Gambar 2.1 Konfigurasi Perangkat Streaming Video (Kurniawati, 2019)

Sedikit berbeda dengan Prerecord Streaming, Live Streaming dilakukan dengan konsep kerja seperti siaran langsung pada televisi. Artinya, pemilik konten melakukan kegiatan perekaman pada suatu konten dan secara langsung disiarkan pada media yang dimilikinya. Nantinya host dapat melihat secara langsung setiap detik yang dilalui oleh pemilik konten tersebut, sehingga engagement yang terjalin dapat lebih erat.

Dalam era kekinian, streaming sendiri dapat menjadi satu cara penyiaran atau publikasi yang lebih efektif karena menghemat waktu dan dapat membuat host atau penonton merasa lebih terkait dengan pembuat konten. Bayangkan jika Anda harus mengunduh semua konten yang akan Anda lihat terlebih dahulu, akan berapa banyak waktu dan paket data yang dihabiskan dalam proses tersebut. Meski streaming memiliki kesan mengkonsumsi banyak data, nyatanya cara ini dipilih oleh sebagian besar masyarakat untuk menikmati konten dari berbagai media baru. Hal ini dikarenakan

dengan melakukan streaming, host dapat menyaksikan konten apapun tanpa perlu menunggu proses pengunduhan, yang disatu sisi akan menghabiskan waktu dan di sisi lain akan menghabiskan media penyimpanan.

Streaming sendiri banyak digunakan oleh tokoh publik untuk menyiarkan kegiatannya secara langsung, atau mendokumentasikan kegiatannya dalam format tertentu yang dapat dilihat lagi di lain waktu. Penerapan streaming yang paling terasa mungkin adalah pada industri sport, dimana hampir setiap turnamen yang diadakan akan disiarkan secara langsung pada media yang dipilih agar dapat dinikmati oleh penggemarnya.

2.2. Pengertian Citra Digital

Sistem olah citra digital merupakan salah satu sistem yang saat ini dikembangkan karena dalam alasan keamanan. Proses pengenalan citra digital akan dilakukan dengan pengolahan data wajah seseorang, namun yang menjadi pokok permasalahan adalah bagaimana hal tersebut dapat terwujud dalam sebuah sistem. Oleh karena itu, dibutuhkan sebuah algoritma yang mana nantinya dapat mengenali wajah dan dapat dipadukan dengan sistem keamanan dengan pengenalan citra digital (Gonzalez & Woods, 2003.), mendefinisikan citra adalah suatu gambaran atau kemiripan dari suatu objek. Citra analog tidak dapat direpresentasikan dalam komputer, sehingga tidak dapat diproses di komputer secara langsung. Tentu agar bisa diproses di komputer, citra analog harus dikonversi menjadi citra digital. Citra digital adalah citra yang dapat diolah oleh komputer. Sedangkan citra yang dihasilkan dari peralatan digital (citra digital) langsung bisa diolah oleh komputer karena, di dalam peralatan digital terdapat sistem sampling dan kuantisasi sedangkan peralatan analog tidak dilengkapi oleh kedua sistem tersebut.

Citra digital merupakan kesatuan dari berbagai elemen yang terdiri dari kecerahan (brightness), kontras (contrast), kontur (contour), warna (color), bentuk (shape), dan tekstur (texture). Secara garis besar citra dapat dibagi menjadi dua jenis, citra diam (still image) dan citra bergerak (motion image) (Gonzalez, 1992). Banyak peralatan elektronik yang menghasilkan citra digital misalnya scanner, kamera digital, mikroskop digital, dan pembaca sidik jari (fingerprint reader). Untuk dapat mengolah citra digital menjadi bentuk yang kita inginkan dapat digunakan beberapa macam Perangkat lunak,

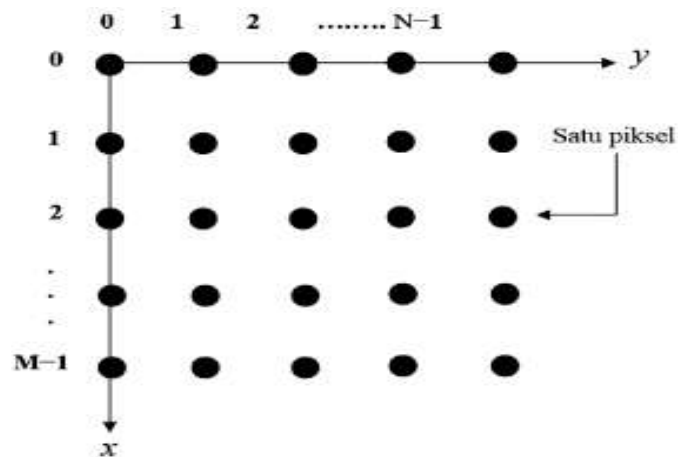
antara lain Adobe Photoshop dan GIMP (GNU Image Manipulation Program) menyajikan berbagai fitur untuk mengolah citra digital (Kadir et al., 2013).

Menurut Asmara (2018), citra digital adalah representasi numeric dari citra dua dimensi. Nilai numeric yang direpresentasikan umumnya adalah nilai biner 8 bit. Nilai biner ini disimpan pada elemen citra yang sering disebut sebagai pixel. Citra digital berisi pixel yang jumlah baris dan kolomnya tetap. Pixel adalah elemen gambar terkecil dari citra digital. Pixel disimpan pada memori komputer sebagai map raster, yaitu array dua dimensi bertipe integer.

Sistem sampling adalah sistem yang mengubah citra kontinu menjadi citra digital dengan cara membagi citra analog menjadi M baris dan N kolom, sehingga menjadi citra diskrit. Semakin besar nilai M dan N, semakin halus citra digital yang dihasilkan. Pertemuan antara baris dan kolom disebut piksel. Sistem kuantisasi adalah sistem yang melakukan pengubahan intensitas analog ke intensitas diskrit, sehingga dengan proses ini dimungkinkan untuk membuat gradasi warna sesuai dengan kebutuhan (Gonzalez & Woods, 2003).

2.2.1. Representasi Citra Digital

Hasil sampling dan kuantisasi dari sebuah citra adalah bilangan real yang membentuk sebuah matriks M baris dan N kolom. Ini berarti ukuran citra MxN. Secara umum, sistem koordinat yang dipergunakan untuk mewakili citra dalam teori pengolahan citra seperti digambarkan pada Gambar 2.2 adalah sebuah citra digital diwakili oleh matriks yang terdiri dari M baris dan N kolom, dimana perpotongan antara baris dan kolom disebut piksel. Piksel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y) adalah $f(x,y)$, yaitu besar intensitas atau warna dari piksel di titik itu (Gonzalez & Woods, 2003).



Gambar 2.2 Sistem Koordinat yang Dipergunakan untuk Mewakili Citra

Artinya, sebuah citra digital dapat ditulis dalam bentuk matriks Gambar 2.3 berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & \dots & \dots & f(1,M-1) \\ \dots & \dots & \dots & \dots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

Gambar 2.3 Matriks citra

Berdasarkan gambaran di atas, maka secara matematis citra digital dapat dituliskan sebagai fungsi intensitas $f(x,y)$, dimana harga x (baris) dan y (kolom) merupakan koordinat posisi dan $f(x,y)$ adalah nilai fungsi pada setiap titik (x,y) yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari piksel di titik tersebut.

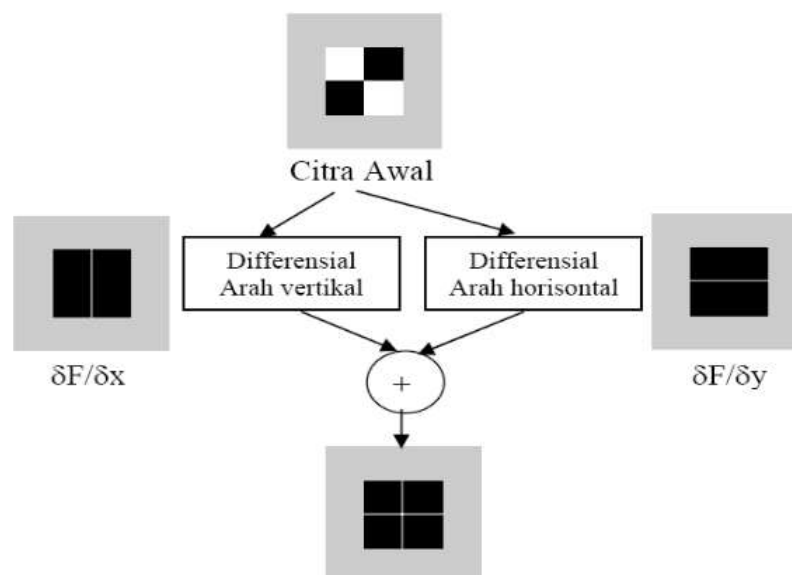
Citra digital dapat ditampilkan pada berbagai macam perangkat lunak penampil citra (*image viewer*). *Web Browser* dapat menampilkan standard citra format internet secara langsung seperti GIF, JPEG, dan PNG. Beberapa browser dapat menampilkan SVG yang merupakan format standard W3C. Beberapa citra sains saat ini berukuran sangat besar (sebagai contoh 46 giga pixel ukuran citra dari galaxy BimaSakti, berukuran 194 GB). Pada *Microsoft visual studio C#*, citra digital akan ditampilkan oleh *control IDE PictureBox*.

2.3. Deteksi Tepi

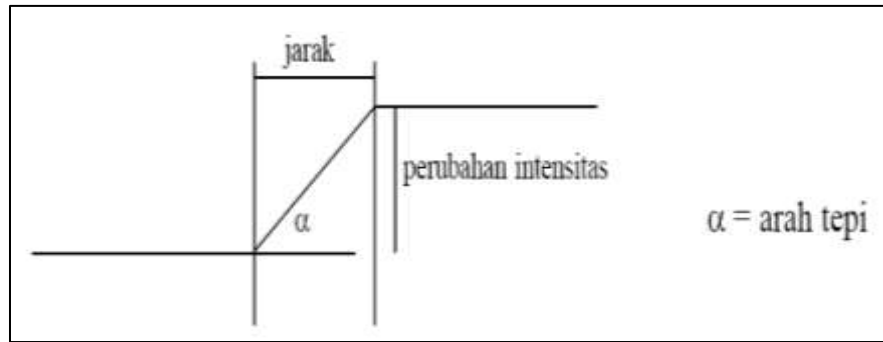
Tepi citra (*edge*) adalah perubahan nilai intensitas derajat keabuan yang cepat/tiba-tiba (besar) dalam jarak yang singkat. Sedangkan deteksi tepi (*Edge Detection*) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra. Deteksi tepi (*Edge detection*) adalah operasi yang dijalankan untuk mendeteksi garis tepi (*edges*) yang membatasi dua wilayah citra homogen yang memiliki tingkat kecerahan yang berbeda. Deteksi tepi pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, tujuannya adalah: (Venmathi, 2016).

1. Untuk menandai bagian yang menjadi detail citra.
2. Untuk memperbaiki detail dari citra yang kabur, yang terjadi karena error atau adanya efek dari proses akuisisi citra.
3. Serta untuk mengubah citra 2D menjadi bentuk kurva.

Suatu titik (x,y) dikatakan sebagai tepi (*edge*) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Gambar 2.4 berikut ini menggambarkan bagaimana tepi suatu citra diperoleh.



Gambar 2.4 Proses deteksi tepi citra (Liantoni, 2015)

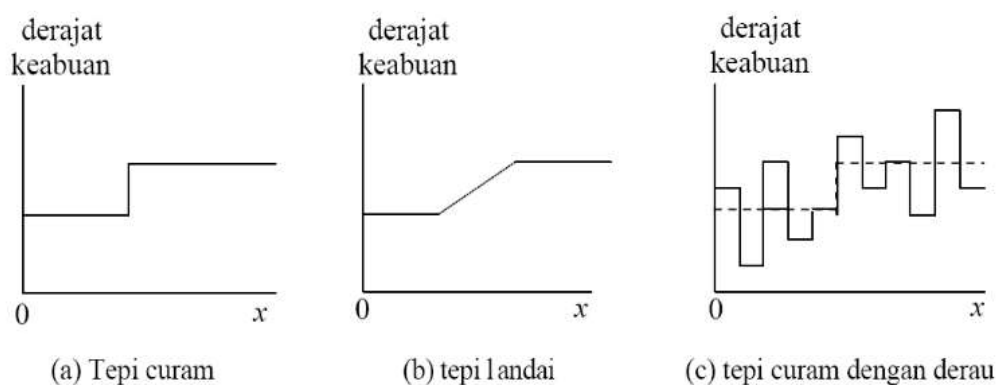


Gambar 2.5 Model tepi satu citra (Liantoni, 2015)

Pada Gambar 2.5, ada tiga macam tepi yang terdapat di dalam citra digital, ketiganya adalah (Panetta *et al.*, 2015):

1. Tepi curam, tepi dengan perubahan intensitas yang tajam. Arah tepi berkisar 90° .
2. Tepi landai, disebut juga tepi lebar, yaitu tepi dengan sudut arah yang kecil. Tepi landai dapat dianggap terdiri dari sejumlah tepi-tepi lokal yang lokasinya berdekatan.
3. Tepi yang mengandung derau (*noise*).

Umumnya tepi yang terdapat pada aplikasi *computer vision* mengandung derau. Operasi peningkatan kualitas citra (*image enhancement*) dapat dilakukan terlebih dahulu sebelum pendeteksian tepi seperti pada Gambar 2.6.



Gambar 2.6 Jenis – jenis tepi citra (Panetta *et al.*, 2015)

2.3.1. Operator Sobel

Operator Sobel merupakan pengembangan operator Robert dengan menggunakan High Pass Filter (HPF) yang diberi satu angka nol penyangga. Proses yang digunakan oleh operator Sobel merupakan proses dari sebuah konvolusi yang telah ditetapkan terhadap citra yang terdeteksi. Dalam operator Sobel digunakan matrik konvolusi 3 X 3 dan susunan piksel-pikselya di sekitar pixel (x, y) (Israni & Jain, 2016). Matriks konvolusi 3x3 piksel dapat dilihat seperti pada Gambar 2.7.

p1	p2	p3
p8	(x,y)	p4
p7	p6	p5

Gambar 2.7 Matriks konvolusi 3 x 3 Sobel (Israni & Jain, 2016)

P (1,2,3,4,5,6,7,8) = nilai piksel citra

Operator ini mengambil prinsip dari fungsi Laplacian dan Gaussian yang dikenal sebagai fungsi untuk membangkitkan HPF. Kelebihan dari operator Sobel ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi. Sehingga besar gradient (G) dapat dihitung dengan menggunakan persamaan:

$$S_x = (p_3 + cp_4 + p_5) - (p_1 + cp_8 + p_7) \dots\dots\dots (2.1)$$

$$S_y = (p_1 + cp_2 + p_3) - (p_7 + cp_6 + p_5) \dots\dots\dots (2.2)$$

Dengan nilai c konstanta bernilai dua, sehingga terbentuk matriks operator Sobel dapat digambarkan seperti pada Gambar 2.8.

S_x	-1	0	1
	-2	0	2
	-1	0	1

S_y	-1	-2	-1
	0	0	2
	1	2	1

Gambar 2.8 Matriks Operator Sobel (Gonzales, 2003)

Biasanya operator Sobel menempatkan penekanan atau pembobotan pada piksel - piksel yang lebih dekat dengan titik pusat jendela, sehingga pengaruh piksel-piksel tetangga akan berbeda sesuai dengan letaknya terhadap titik di mana gradien dihitung. Dari susunan nilai - nilai pembobotan pada jendela juga terlihat bahwa perhitungan

terhadap gradien juga merupakan gabungan dari posisi mendatar dan posisi vertikal. (Gonzales, 2003)

Pada Gambar 2.9 terlihat bahwa hasil deteksi tepi berupa tepi - tepi dari suatu gambar. Bila diperhatikan bahwa tepi suatu gambar terletak pada titik-titik yang memiliki perbedaan tinggi.



Gambar 2.9 Hasil deteksi tepi

Proses deteksi tepi (*edges detection*) sendiri masing dapat dikelompokkan berdasarkan operator atau metode yang digunakan dalam proses pendeteksian tepi suatu citra untuk memperoleh citra hasil.

2.4. Convolutional Neural Network (CNN)

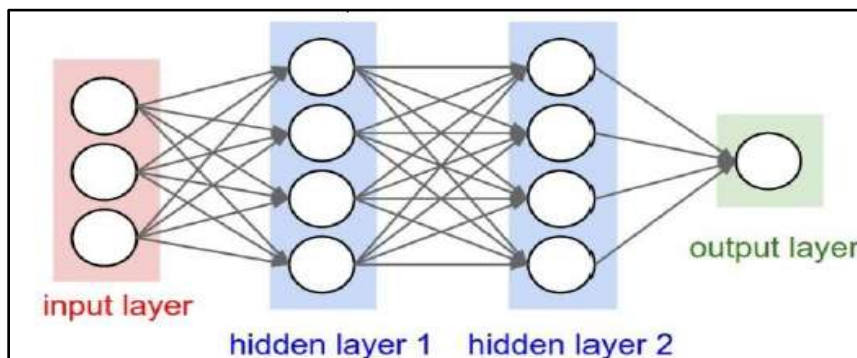
Convolutional Neural Network (CNN) adalah pengembangan dari Multilayer Perceptron (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis Deep Neural Network karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus deteksi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik (Zadeh et al., 2019).

CNN pertama kali dikembangkan dengan nama NeoCognitron oleh Kuniyiko Fukushima, seorang peneliti dari NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Jepang. Konsep tersebut kemudian dimatangkan oleh Yann LeCun, seorang peneliti dari AT&T Bell Laboratories di Holmdel, New Jersey, USA. Model CNN dengan nama LeNet berhasil diterapkan oleh LeCun pada penelitiannya mengenai pengenalan angka dan tulisan tangan. Pada tahun 2012, Alex Krizhevsky dengan penerapan CNN miliknya berhasil menjuarai kompetisi ImageNet Large Scale

Visual Recognition Challenge 2012. Prestasi tersebut menjadi momen pembuktian bahwa metode Deep Learning, khususnya CNN. Metode CNN terbukti berhasil mengungguli metode Machine Learning lainnya seperti SVM pada kasus deteksi objek pada citra.

2.4.1. Konsep CNN

Cara kerja CNN memiliki kesamaan dengan MLP, namun dalam CNN setiap neuron dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi yang dapat dilihat seperti pada Gambar 2.10.



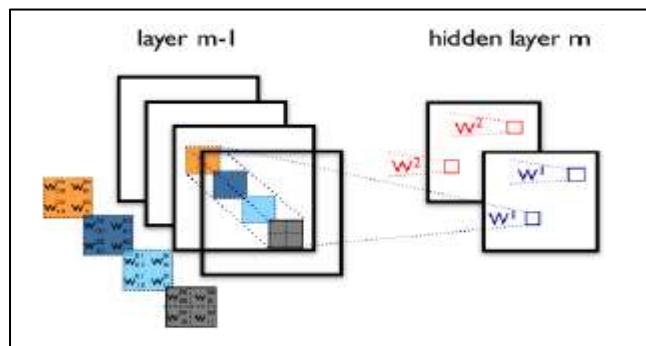
Gambar 2.10 Arsitektur MLP sederhana (Zapletal, 2017)

Sebuah MLP seperti pada Gambar 2.10 memiliki *I layer* (kotak merah dan biru) dengan masing-masing *layer* berisi *ji* neuron (lingkaran putih). MLP menerima *input* data satu dimensi dan mempropagasikan data tersebut pada jaringan hingga menghasilkan *output*. Setiap hubungan antar neuron pada dua *layer* yang bersebelahan memiliki parameter bobot satu dimensi yang menentukan kualitas mode (Zapletal, 2017). Disetiap data *input* pada *layer* dilakukan operasi *linear* dengan nilai bobot yang ada, kemudian hasil komputasi akan ditransformasi menggunakan operasi *nonlinear* yang disebut sebagai fungsi aktivasi.

Pada CNN, data yang dipropagasikan pada jaringan adalah data dua dimensi, sehingga operasi linear dan parameter bobot pada CNN berbeda. Pada CNN operasi linear menggunakan operasi konvolusi, sedangkan bobot tidak lagi satu dimensi saja, namun berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi seperti pada Gambar 2.11 Dimensi bobot pada CNN adalah:

Neuron Input X Neuron Output X Tinggi X Lebar

Karena sifat proses konvolusi, maka CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi seperti citra dan suara.



Gambar 2.11 Proses konvolusi pada CNN (Zapletal, 2017)

2.4.2. Arsitektur Jaringan CNN

JST terdiri dari berbagai layer dan beberapa neuron pada masing-masing layer. Kedua hal tersebut tidak dapat ditentukan menggunakan aturan yang pasti dan berlaku berbeda-beda pada data yang berbeda. Pada kasus MLP, sebuah jaringan tanpa hidden layer dapat memecahkan persamaan linear apapun, sedangkan jaringan dengan satu atau dua hidden layer dapat memecahkan sebagian besar persamaan pada data sederhana. Namun pada data yang lebih kompleks, MLP memiliki keterbatasan. Pada permasalahan jumlah *hidden layer* dibawah tiga *layer*, terdapat pendekatan untuk menentukan jumlah neuron pada masing-masing *layer* untuk mendekati hasil optimal. Penggunaan layer diatas dua pada umumnya tidak direkomendasikan dikarenakan akan menyebabkan *overfitting* serta kekuatan *backpropagation* berkurang secara signifikan (Syafeeza, 2014).

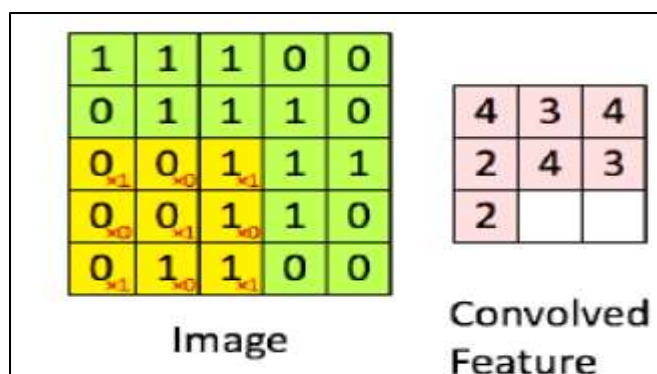
Dengan berkembangnya *deep learning*, ditemukan bahwa untuk mengatasi kekurangan MLP dalam menangani data kompleks, diperlukan fungsi untuk mentransformasi data input menjadi bentuk yang lebih mudah dimengerti oleh MLP. Hal tersebut memicu berkembangnya *deep learning* dimana dalam satu model diberi beberapa *layer* untuk melakukan transformasi data sebelum data diolah menggunakan metode deteksi. Hal tersebut memicu berkembangnya model neural network dengan jumlah *layer* diatas tiga. Namun dikarenakan fungsi *layer* awal sebagai metode ekstraksi fitur, maka jumlah *layer* dalam sebuah CNN tidak memiliki aturan universal dan berlaku berbeda-beda tergantung dataset yang digunakan. Karena hal tersebut,

jumlah *layer* pada jaringan serta jumlah neuron pada masing-masing *layer* dianggap sebagai *hyperparameter* dan dioptimasi menggunakan pendekatan *searching*. Sebuah CNN terdiri dari beberapa *layer*, berdasarkan arsitektur LeNet5 terdapat empat macam *layer* utama pada sebuah CNN namun yang umum diterapkan ada tiga macam lapisan antara lain:

a. *Convolution Layer*

Convolution Layer melakukan operasi konvolusi pada *output* dari *layer* sebelumnya. *Layer* tersebut adalah proses utama yang mendasari sebuah CNN. Konvolusi adalah suatu istilah matematis yang berarti mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Dalam pengolahan citra, konvolusi berarti mengaplikasikan sebuah *kernel* (kotak kuning) pada citra disemua *offset* yang memungkinkan seperti yang ditunjukkan pada Gambar 2.12. Kotak hijau secara keseluruhan adalah citra yang akan dikonvolusi. Kernel bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari citra tersebut dapat dilihat pada gambar disebelah kanannya.

Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input. Konvolusi akan menghasilkan transformasi *linear* dari data *input* sesuai informasi spasial pada data. Bobot pada *layer* tersebut menspesifikasikan kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan *input* pada CNN.

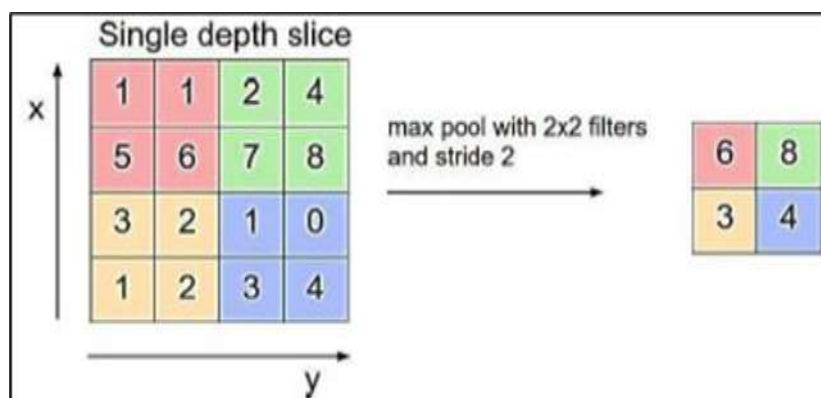


Gambar 2.12 Operasi Konvolusi (Syafeeza, 2014)

b. *Subsampling Layer*

Subsampling adalah proses mereduksi ukuran sebuah data citra dimana dalam pengolahan citra, *subsampling* juga bertujuan untuk meningkatkan invariansi posisi dari fitur. Dalam sebagian besar CNN, metode *subsampling* yang digunakan adalah *max pooling* yang berguna untuk membagi output dari *convolution layer* menjadi beberapa

grid kecil lalu mengambil nilai maksimal dari setiap *grid* untuk menyusun matriks citra yang telah direduksi seperti yang ditunjukkan pada Gambar 2.13. Grid yang berwarna merah, hijau, kuning dan biru merupakan kelompok *grid* yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan *grid* disebelah kanannya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi (pergeseran).



Gambar 2.13 Operasi max pooling (Syafeeza, 2014)

Menurut Springenberg *et al*, penggunaan *pooling layer* pada CNN hanya bertujuan untuk mereduksi ukuran citra sehingga dapat dengan mudah digantikan dengan sebuah *convolution layer* dengan *stride* yang sama dengan *pooling layer* yang bersangkutan.

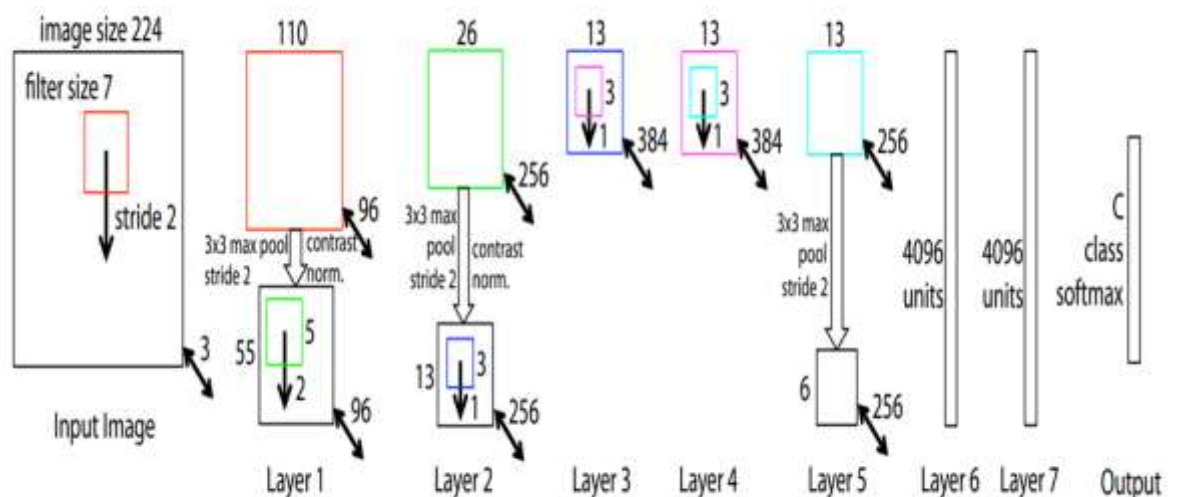
c. Fully Connected Layer

Layer tersebut adalah *layer* yang biasanya digunakan dalam penerapan MLP dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat dideteksi secara linear. Setiap neuron pada convolution layer perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan ke dalam sebuah fully connected layer. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversibel, fully connected layer hanya dapat diimplementasikan di akhir jaringan. Dalam sebuah jurnal oleh Lin *et al.* (2018) dijelaskan bahwa convolution layer dengan ukuran kernel 1 x 1 melakukan fungsi yang sama dengan sebuah fully connected layer namun dengan tetap mempertahankan karakter spasial dari data. Hal tersebut membuat penggunaan fully connected layer pada CNN sekarang tidak banyak dipakai.

2.4.3. Arsitektur ZFNet

ZFNet merupakan pemenang ImageNet Large Scale Visual Recognition Competition (ILSVRC) pada tahun 2013. Para peneliti mampu mengurangi tingkat kesalahan top-5 menjadi 14.8%. Hal tersebut diperoleh dengan mempertahankan struktur AlexNet namun melakukan perubahan pada hyperparameternya. ZFNet merupakan pemenang ImageNet Large Scale Visual Recognition Competition (ILSVRC) pada tahun 2013. Para peneliti mampu mengurangi tingkat kesalahan top-5 menjadi 14.8%. Hal tersebut diperoleh dengan mempertahankan struktur AlexNet namun melakukan perubahan pada hyperparameternya

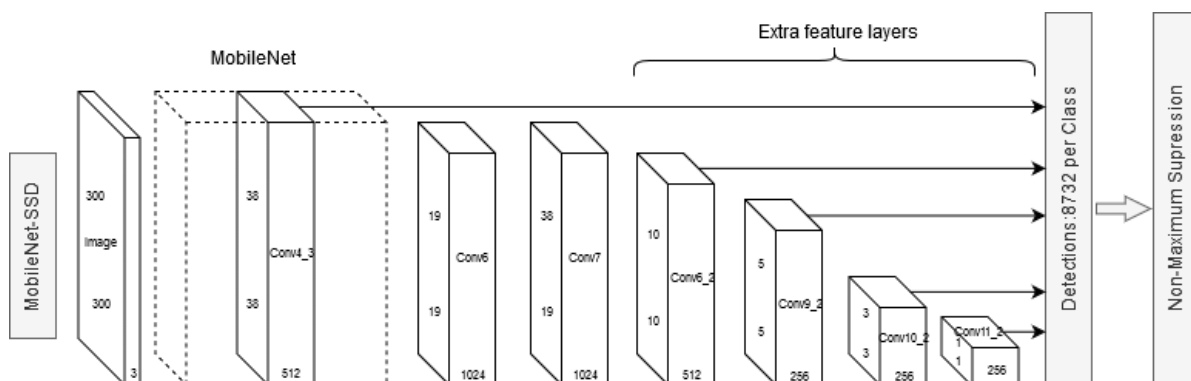
Pada arsitektur ZfNet dari algoritma CNN dilakukan pemasukan citra dengan dimensi 224x224x3 dan selanjutnya dilakukan 96 kali konvolusi sebanyak piksel 7x7 dengan 2 langkah, diikuti oleh aktivasi ReLU, 3x3 max pooling dengan langkah 2 dan normalisasi kontras lokal. Diikuti oleh 256 filter masing-masing 3x3 yang kemudian kontras lokal dinormalisasi dan dikumpulkan. Lapisan ketiga dan keempat identik dengan 384 kernel masing-masing 3x3. Lapisan kelima memiliki 256 filter 3x3, diikuti oleh penyatuan maksimal 3x3 dengan langkah 2 dan normalisasi kontras lokal. Lapisan keenam dan ketujuh masing-masing menampung 4096 unit padat dan selanjutnya memasukkan lapisan padat 1000 neuron yaitu jumlah kelas di ImageNet.



Gambar 2.14 Arsitektur ZFNet (Zeiler, *et al.* 2014)

2.5. MobileNet Single Shot Detector

Single Shot MultiBox Detector (SSD) merupakan algoritma pendeteksi objek pada gambar yang menerapkan Single deep neural network. Detektor objek seperti Faster-RCNN memiliki akurasi yang cukup tinggi, namun mengorbankan waktu komputasi sehingga tidak bisa digunakan dalam sistem real time. Berbanding terbalik dengan Faster-RCNN, YOLO dapat digunakan secara real time, namun mengorbankan akurasinya. Liu et al. (2016) mengembangkan metode Single Shot MultiBox Detector sebagai perbaikan dari pertukaran antara waktu komputasi dan tingkat akurasi tersebut. Arsitektur SSD diawali dengan layers ekstraksi fitur dari arsitektur network yang biasanya ada pada deteksi citra lainnya (seperti Inception, ResNet, VGG, MobileNet) tetapi tidak termasuk layer untuk deteksi. Network ini disebut juga dengan base network/backbone. Kemudian di ujung base network ditambahkan beberapa convolutional feature layer yang ukurannya semakin kecil sehingga dimungkinkan untuk prediksi multi skala.



Gambar 2.15 Arsitektur SSD

2.6. OpenCV

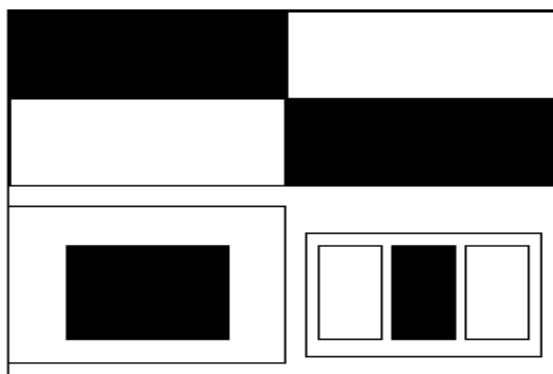
OpenCV merupakan singkatan dari Intel *Open Source Computer Vision Library* yang sekurang-kurangnya terdiri dari 300 fungsi-fungsi C, bahkan bisa lebih. *Software* ini gratis, dapat digunakan dalam rangka komersil maupun non komersil, tanpa harus membayar lisensi ke intel. OpenCV dapat beroperasi pada komputer berbasis Windows ataupun Linux. Library OpenCV adalah suatu cara penerapan bagi komunitas *open source vision* yang sangat membantu dalam kesempatan meng-*update* penerapan *computer vision* sejalan dengan pertumbuhan PC (*personal computer*) yang terus

berkembang. *Software* ini menyediakan sejumlah fungsi-fungsi *image processing*, seperti halnya dengan fungsi-fungsi analisis gambar dan pola (Kadir, 2013).

Beberapa contoh aplikasi dari OpenCV adalah pada *Human-Computer Interaction* (interaksi manusia komputer); *Object Identification* (Identifikasi Objek), *Segmentation* (segmentasi) dan *Recognition* (pengenalan); *Face Recognition* (pengenalan wajah); *Gesture Recognition* (pengenalan gerak isyarat), *Motion Tracking* (penjajakan gerakan), *Ego Motion* (gerakan ego), dan *Motion Understanding* (pemahaman gerakan); *Structure From Motion* (gerakan dari struktur); dan *Mobile Robotics* (robot-robot yang bergerak). Pengenalan wajah pada OpenCV menggunakan metode yang disebutkan oleh metode Viola-Jones, juga disebut sebagai *Haar cascade classifier*. Pendekatan ini untuk mendeteksi objek dalam gambar dengan menggabungkan empat konsep yaitu:

1. Segi empat sederhana, disebut dengan *Haar feature*.
2. Sebuah Integral gambar untuk mempercepat menemukan *feature*.
3. Metode *AdaBoost machine-learning*.
4. Deteksi bertingkat untuk menyatukan banyaknya *feature* secara efisien.

Bentuk yang Viola dan Jones gunakan adalah berdasarkan *Haar wavelets*. *klasifikasi* ini menggunakan gelombang segiempat tunggal (satu interval tinggi dan yang satunya interval rendah) dalam dua dimensi, gelombang persegi adalah pasangan dari segiempat yang berdekatan satu putih yang satunya hitam seperti pada Gambar 2.16.



Gambar 2.16 Deteksi Haar digunakan dalam openCV (Cahu *et al.*, 2019)

Pada gambar 2.17 dapat dilihat bahwa OpenCV dapat digunakan untuk *object detection*.



Gambar 2.17 OpenCV digunakan untuk *object detection*

2.7. Mengukur Kinerja Metode dengan *Confusion Matrix*

Pengukuran terhadap kinerja suatu sistem merupakan hal yang penting dimana kinerja sistem menggambarkan seberapa baik sistem dalam pengolahan data. *Confusion matrix* merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil deteksi yang dilakukan oleh sistem dengan hasil deteksi yang seharusnya.

Berdasarkan jumlah keluaran kelasnya, sistem dapat dibagi menjadi 4 (empat) jenis yaitu deteksi *binary*, *multi-class*, *multi-label* dan *hierarchical*. Pada deteksi *binary*, data masukan dikelompokkan ke dalam salah satu dari dua kelas. Jenis deteksi ini merupakan bentuk deteksi yang paling sederhana dan banyak digunakan. Contoh penggunaannya antara lain dalam sistem yang melakukan deteksi orang atau bukan, sistem deteksi kendaraan atau bukan, dan sistem deteksi pergerakan atau bukan.

Sementara itu, pada bentuk deteksi *multi-class*, data masukan dideteksi menjadi beberapa kelas. Sebagai contoh sistem yang dapat mendeteksi jenis kendaraan

seperti sepeda, sepeda motor, mobil, bus, truk, dan sebagainya. Bentuk deteksi *multi-label* pada dasarnya sama dengan *multi-class* dimana data dikelompokkan menjadi beberapa kelas, namun pada deteksi *multi-label*, data dapat dimasukkan dalam beberapa kelas sekaligus. Bentuk deteksi yang terakhir adalah *hierarchical*. Data masukan dikelompokkan menjadi beberapa kelas, namun kelas tersebut dapat dikelompokkan kembali menjadi kelas-kelas yang lebih sederhana secara hirarkis. Contohnya dalam penelitian ini, arah pergerakan dikelompokkan menjadi 12 arah pergerakan yang tentunya dapat disederhanakan menjadi 4 arah.

Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses deteksi. Keempat istilah tersebut adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN). Nilai *True Negative* (TN) merupakan jumlah data negatif yang terdeteksi dengan benar, sedangkan *False Positive* (FP) merupakan data negatif namun terdeteksi sebagai data positif. Sementara itu, *True Positive* (TP) merupakan data positif yang terdeteksi benar. *False Negative* (FN) merupakan kebalikan dari *True Positive*, sehingga data positif, namun terdeteksi sebagai data negatif. Pada jenis deteksi *binary* yang hanya memiliki 2 keluaran kelas, *confusion matrix* dapat disajikan seperti pada Tabel 2.1.

Tabel 2.1 Deteksi *confusion matrix*

Kelas	Terdeteksi Positif	Terdeteksi Negatif
Positif	TP (True Positive)	FN (False Negative)
Negatif	FP (False Positive)	TN (True Negative)

Berdasarkan nilai *True Negative* (TN), *False Positive* (FP), *False Negative* (FN), dan *True Positive* (TP) dapat diperoleh nilai akurasi, presisi dan *recall*. Nilai akurasi menggambarkan seberapa akurat sistem dapat mendeteksi data secara benar. Dengan kata lain, nilai akurasi merupakan perbandingan antara data yang terdeteksi benar dengan keseluruhan data. Nilai akurasi dapat diperoleh dengan Persamaan 2.16. Nilai presisi menggambarkan jumlah data kategori positif yang dideteksi secara benar dibagi dengan total data yang dideteksi positif. Presisi dapat diperoleh dengan Persamaan 2.17 Sementara itu, *recall* menunjukkan berapa persen data kategori positif

yang terdeteksikan dengan benar oleh sistem. Nilai *recall* diperoleh dengan Persamaan berikut.

$$Akurasi = (TP + TN) / (TP + TN + FP + FN) * 100\% \dots\dots\dots (2.16)$$

$$Precision = (TP / (TP + FP)) * 100\% \dots\dots\dots (2.17)$$

$$Recall = (TP / (TP + FN)) * 100\% \dots\dots\dots (2.18)$$

dimana:

1. TP adalah *True Positive*, yaitu jumlah data positif yang terdeteksi dengan benar oleh sistem.
2. TN adalah *True Negative*, yaitu jumlah data negatif yang terdeteksi dengan benar oleh sistem.
3. FN adalah *False Negative*, yaitu jumlah data negatif namun terdeteksi salah oleh sistem.
4. FP adalah *False Positive*, yaitu jumlah data positif namun terdeteksi salah oleh sistem

2.8. Penelitian Terkait

Adapun penelitian terdahulu diatas dapat dilihat seperti pada Tabel 2.2.

Tabel 2.2 Penelitian Terdahulu

Peneliti	Judul Penelitian	Metode	Keterangan
Lanes, Y. T. A, Saragih, Y., Lammada, I., Irawan, R. & Dewi, R. 2019.	Perancangan	Sensor EEG	Aplikasi yang dapat mengirimkan data kantuk pengendara sepeda motor ke <i>website</i> melalui jaringan internet sehingga dapat dilihat oleh keluarga pengendara.
	Dan		
	Implementasi		
	Aplikasi Sistem		
	Pemantauan		
	Helm		
Widodo, B., Armanto, H. & Setyati, E. 2021.	Pendeteksi	Metode Convolutional Neural Network	Proses deteksi pada sistem ini melakukan lokalisasi dan deteksi
	Kantuk Berbasis		
	Website		

	Dengan Metode Convolutional Neural Network		dengan sekali langkah proses sehingga hasil dari proses deteksi ini adalah orang menggunakan helm proyek dan orang tidak menggunakan helm proyek. Pengujian sistem deteksi dilakukan secara individu maupun kelompok maksimal 5 orang dengan <i>F1-score</i> yang diperoleh sebesar 0,79.
Sthevanie, F., Kurniawan, A. & Ramadhani, K. N. 2020.	Deteksi Helm pada Video Pengendara Sepeda Motor menggunakan Ekstraksi Ciri Histogram of Oriented Gradients	Histogram of Oriented Gradients	Sistem yang dibangun telah dapat mendeteksi pengendara yang menggunakan helm dan tidak menggunakan helm pada 100 citra yang diekstrak dari video dengan nilai f-measure sebesar 90.67%
Abidin, S. 2021.	Deteksi Wajah Menggunakan Metode Haar Cascade Classifier Berbasis	Haar Cascade Classifier	Metode haar cascade classifier sangat ideal digunakan untuk deteksi wajah secara realtime yang di capture dari webcam laptop.

Webcam Pada Matlab			
Dewi, I. S. & Chandra, A. Y. 2021.	Prototipe Sistem Deteksi Suhu Tubuh Dan Masker Wajah Menggunakan Algoritma Local Binary Pattern dan Arduino Nano.	AdaBoost	Hasil dari pengujian prototipe ini menunjukan tingkat akurasi skor F1 0.99 untuk deteksi masker dan pengukuran suhu tubuh secara real time setiap detik dengan satuan derajat celcius (C) dengan rata-rata presentase error tertinggi 3.10% pada jarak 4 meter.
	Implementasi Algoritma Haar Cascade pada Aplikasi Pengenalan Wajah	Haar Cascade	Nilai akurasi pengenalan wajah sebesar 80,5%
Said Si Kaddour, Yassir Aberni, Larbi Boubchir, Mohammed Raddadi and Boubaker Daachi, 2021	<i>Convolutional Neural Algorithm for Palm Vein Recognition using ZFNet Architecture</i>	<i>Convolutional Neural Network</i> berbasis <i>Zfnet</i>	Nilai akurasi sebesar 89,5%
Ramadhan, M. P. (2023)	Deteksi Objek Robot Sepakbola secara Real	MobileNet SSD	Nilai akurasi sebesar 90%

Time dengan
Mobilenet
Single Shot
Detector (SSD)

Pada penelitian deteksi helm diatas dilakukan dengan mengambil nilai ekstraksi ciri dari setiap data helm. Pada penelitian ini dilakukan implementasi algoritma CNN untuk mendeteksi helm pada pengendara sepeda motor.

BAB 3

ANALISIS DAN PERANCANGAN

3.1. Data yang Digunakan

Dalam penelitian ini data yang digunakan adalah data citra yang diekstrak dari video lalu lintas. Dataset adalah bahan uji suatu program untuk dijadikan sebagai media pembelajaran dan dataset dapat di tambahkan, dihapus atau memperbarui di dalam memori laptop. Citra dibagi menjadi tiga kelas, yaitu helm, tanpa helm dan motor. Total citra dalam dataset yang dikumpulkan penulis adalah sejumlah 200 citra.



Gambar 3.1 Citra pengendara sepeda motor tanpa helm

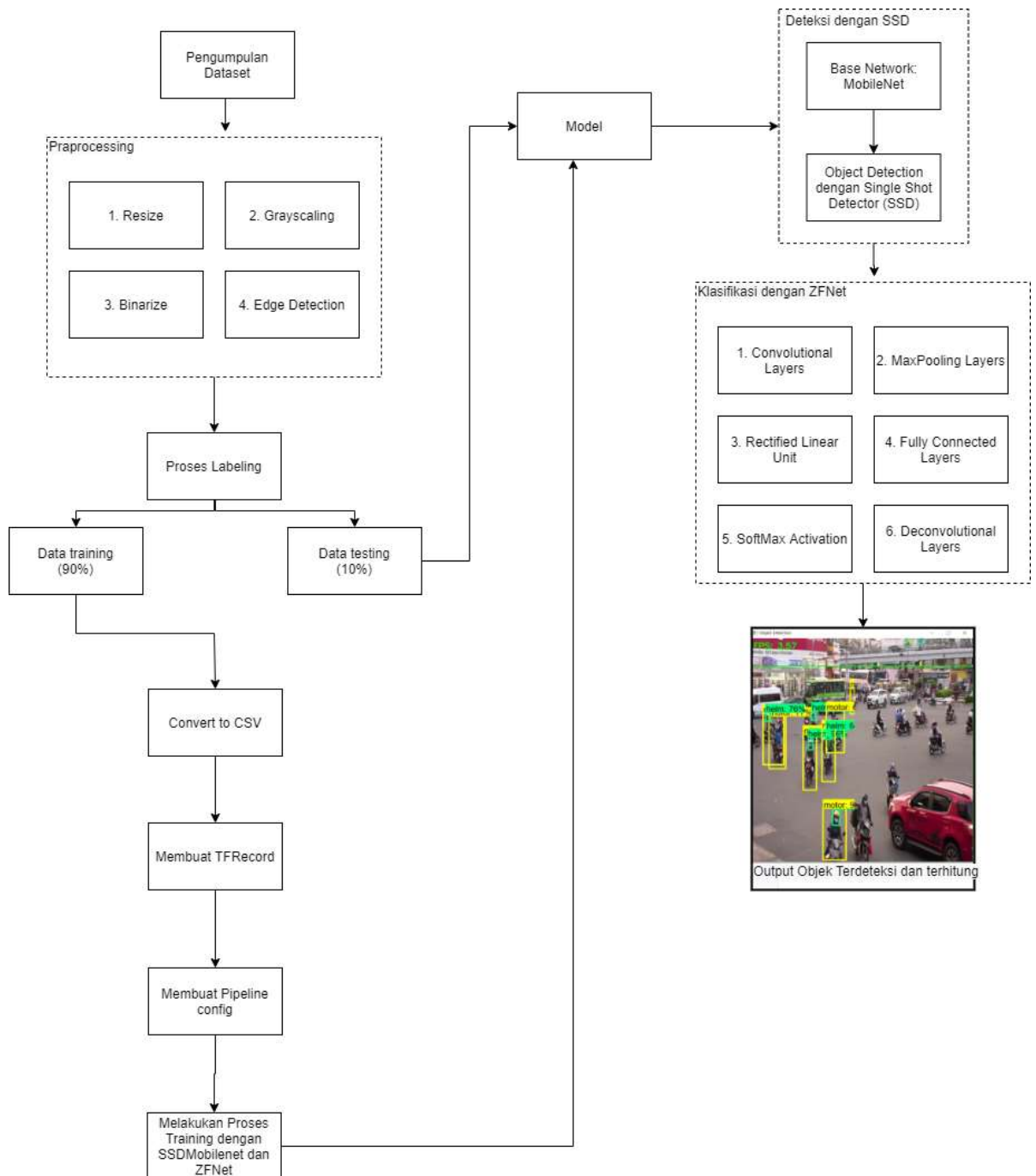
Berikut citra pengendara sepeda motor menggunakan helm.



Gambar 3.2 Citra pengendara sepeda motor menggunakan helm

3.2. Arsitektur Umum

Berikut arsitektur umum dari penelitian ini:



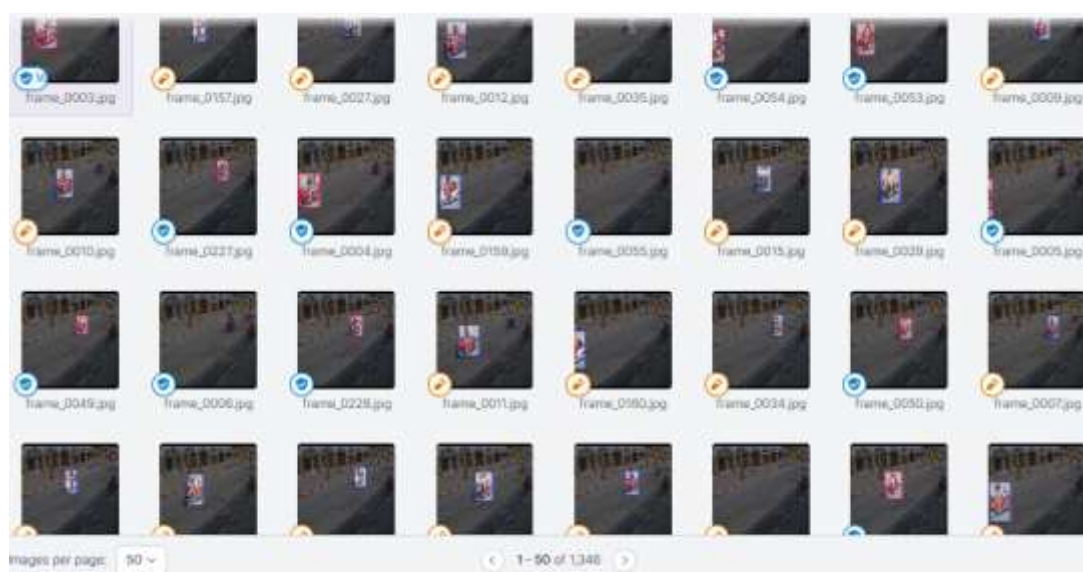
Gambar 3.3 Arsitektur Umum

Dataset berupa citra diambil dari video lalu lintas. Citra – citra tersebut kemudian dilakukan proses *preprocessing*. Tahapan *preprocessing* ini meliputi proses pengubahan ukuran citra (*resizing*) menjadi ukuran 640 x 640 piksel. Kemudian

dilakukan proses pengubahan warna citra menjadi keabuan (*grayscale*). Citra kemudian diubah *value*-nya menjadi satu (1) dan nol (0). Di akhir tahap *preprocessing*, citra dideteksi tepinya untuk menentukan batas antara objek dan *background*. Selanjutnya dilakukan proses labeling menjadi tiga kelas yaitu, kelas sepeda motor, kelas pengendara tanpa helm, dan kelas pengendara menggunakan helm. Selanjutnya dilakukan proses pelatihan data menggunakan algoritma MobileNet Single Shot Detector dan ZFNet. Hasil dari proses *training* ini berupa model yang akan digunakan dalam proses deteksi dan deteksi objek.

3.2.1. Pengumpulan Dataset

Proses ini merupakan proses awal dari penelitian. Penulis mengumpulkan citra secara manual dari video *traffic* lalu lintas yang diambil dari laman youtube. Citra yang dikumpulkan sebanyak 220 citra.



Gambar 3.4 Dataset yang telah diambil

3.2.2. Preprocessing

Tahapan ini meliputi empat (4) tahapan, yaitu:

1. *Resize*

Proses ini mengubah citra yang awalnya memiliki ukuran bervariasi menjadi satu ukuran yaitu 640 x 640 piksel. Proses ini dilakukan agar ukuran citra mengecil dan piksel yang tidak mempengaruhi hasil dari proses deteksi jumlahnya berkurang. Berikut pseudocode dari proses ini:

```
Function: Resizing Dataset
Input: Dataset
Output: Resized Dataset
BEGIN
    GET Dataset
    THEN Resize to 640 X 640
END
```

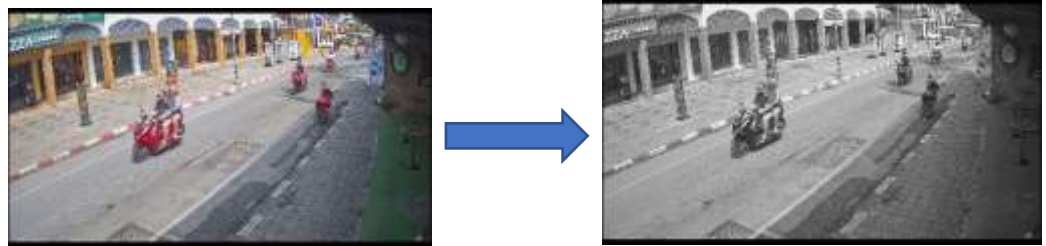


Gambar 3.5 Proses *resizing*

2. *Grayscale*

Citra yang awalnya memiliki berbagai macam warna diubah menjadi warna keabuan. Proses ini diperlukan karena warna tidak mempengaruhi dalam proses deteksi dan mempercepat proses *training* model. Proses ini dapat digambarkan pada pseudocode di bawah ini:

```
Function: Grayscale Dataset
Input: Colored Dataset
Output: Grayscaled Dataset
BEGIN
    GET Dataset
    THEN Grayscale
END
```



Gambar 3.6 Proses *Grayscale*

3. *Binarize*

Citra yang sudah dibuat keabuan kemudian di-*binarize*. Proses ini mengubah *value* dari piksel citra sebelumnya menjadi biner atau nol (0) dan satu (1). Satu bermakna piksel dari citra tersebut merupakan *foreground* (objek yang ingin dideteksi) sedangkan nol merupakan *background* (latar belakang dari objek yang ingin dideteksi). Proses ini dapat digambarkan pada pseudocode di bawah ini:

```
Function: Binarizing Dataset
Input: Dataset
Output: Binarized Dataset
BEGIN
    GET Dataset
    THEN Binarize
END
```

4. *Edge Detection*

Proses deteksi tepi (*edge detection*) dikelompokkan berdasarkan operator atau metode yang digunakan dalam proses pendeteksian tepi suatu citra untuk memperoleh citra hasil. Proses ini dapat digambarkan pada pseudocode berikut:

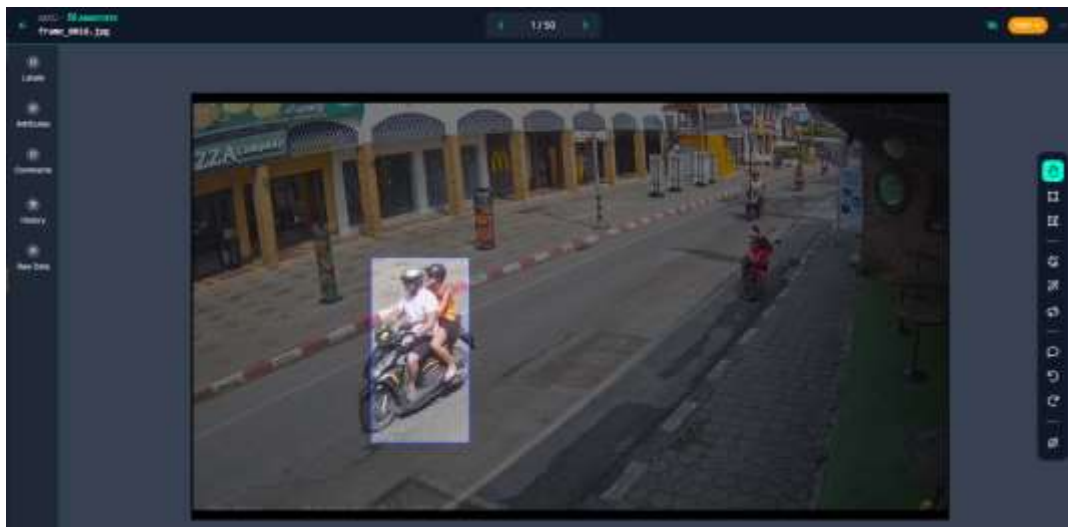
```

Function: Edge Detecting Dataset
Input: Dataset
Output: Edge Detected Dataset
BEGIN
    GET Dataset
    THEN Detect Edge Dataset
END

```

3.2.3. Labelling

Proses ini dilakukan dengan cara melabeli atau membuat *bounding box* satu persatu dataset berupa citra yang telah di-*preprocessing* oleh peneliti. Proses ini akan menghasilkan file berformat xml yang akan menandai setiap posisi *bounding box* yang telah dibuat. Proses pelabelan citra ini dapat digambarkan gambar 3.7.



Gambar 3.7 Proses pelabelan citra

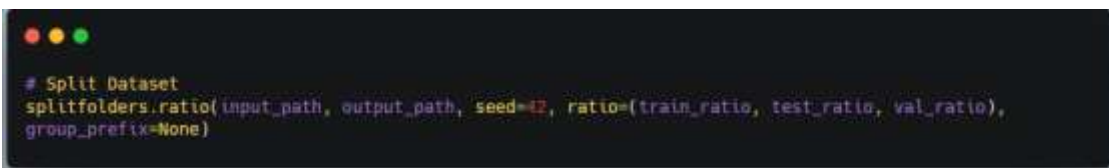
3.2.4. Splitting Data

Setelah citra dilabeli, kemudian dilakukan proses pembagian data menjadi data *training* (latih) dan data *testing* (uji). Diambil sebanyak 90% dari total citra pada dataset untuk digunakan sebagai data latih dan sisanya berupa 10% digunakan sebagai data uji. Proses ini dapat digambarkan pada pseudocode berikut:


```

Function: Splitting Dataset
Input: Dataset
Output: Splitted Dataset
BEGIN
    GET Dataset
    THEN Split to train 90% and test 10%
END

```



```

# Split Dataset
splitfolders.ratio(input_path, output_path, seed=42, ratio=(train_ratio, test_ratio, val_ratio),
group_prefix=None)

```

Gambar 3.8 Potongan kode *split dataset*

3.2.5. *Convert to CSV*

Proses ini mengubah setiap file xml yang berisi *bounding box* setiap citra menjadi satu file CSV. Hal ini berguna agar model dapat membaca posisi setiap objek ketika akan di-*training*. Setelah selesai proses *splitting*, dilakukan proses selanjutnya yaitu membuat file CSV `train_labels.csv` yang berisi data seluruh nama file citra yang akan digunakan dalam proses training beserta kelas dan anotasinya, lalu file `test_labels.csv` yang berisi data seluruh nama file citra yang akan digunakan dalam proses testing beserta kelas dan anotasinya dan `label_map.pbtxt` yang berisi nama dari seluruh kelas yang ada dalam file XML.

3.2.6. *Create TFRecord file*

Selanjutnya peneliti membuat file TFRecord. File TFRecord berguna untuk membuat record *training* dan *testing* agar model dapat di-*training*. Proses ini dapat digambarkan pada pseudocode berikut:

```

Function: Generate train.record and test.record
Input: Labels in CSV and label_map
Output: train.record and test.record file
BEGIN
    GET labels and label_map
    THEN Generate train.record and test.record
END

```

3.2.7. Create Pipeline Config

Pipeline merupakan file konfigurasi yang diatur sesuai tujuan *training* masing – masing. File ini akan mengatur bagaimana proses *training* akan dilakukan. Konfigurasi pipeline ini berfungsi untuk mengatur *hyperparameter* untuk proses *training* nantinya.

3.2.8. Training

Setelah semua persiapan selesai, penulis melakukan proses *training*. Proses latih ini melibatkan dua algoritma, yaitu algoritma MobileNet SSD dan ZFNet. Proses ini akan menghasilkan model *training custom* yang akan digunakan untuk mendeteksi penggunaan helm nantinya.

```

Function: Training Model
Input: Previously Trained Model and pipeline config
Output: Tensorflow model (.pb)
BEGIN
    GET Previously Trained Model and pipeline config
    THEN train model
END

```

3.2.9. Proses Deteksi

Proses deteksi objek dilakukan menggunakan algoritma MobileNet SSD. Algoritma ini akan menggunakan MobileNet sebagai *base network*-nya. MobileNet ini berfungsi sebagai ekstraktor dari fitur – fitur yang membedakan tiap objek dalam citra. Selanjutnya pendeteksian objek dilakukan oleh SSD MultiBox. SSD MultiBox

melakukan proses lokalisasi dan dengan feature map yang berbeda – beda. Citra input pertama-tama diubah ukurannya menjadi 224x224 piksel.

3.2.10. Proses Deteksi

Proses deteksi objek dilakukan menggunakan algoritma ZFNet. Algoritma ini memiliki 6 tahapan, yaitu:

1. Convolutional layers

Dalam lapisan ini, filter konvolusional diterapkan untuk mengekstrak fitur penting, ZFNet terdiri dari beberapa lapisan konvolusional untuk mengekstrak fitur penting. Citra kemudian diproses oleh lima lapisan konvolusional. Setiap lapisan konvolusional menggunakan filter 3x3 dengan sejumlah tertentu filter. Filter ini digulirkan di atas gambar, menghasilkan peta fitur baru.

2. MaxPooling Layers

Lapisan MaxPooling digunakan untuk menurunkan sampel dimensi spasial peta fitur. Ini terdiri dari fungsi agregasi yang dikenal sebagai maxima. Setelah setiap dua lapisan konvolusional, ada lapisan pooling maksimum. Lapisan pooling maksimum ini mengurangi dimensi peta fitur dengan mengambil nilai maksimum dalam jendela 2x2.

3. Rectified Linear Unit

Relu digunakan setelah setiap lapisan konvolusi untuk memperkenalkan non linearitas ke dalam model yang sangat penting untuk mempelajari pola yang kompleks. Ini memperbaiki peta fitur untuk memastikan peta fitur selalu positif. Peta fitur ini kemudian diaktifkan menggunakan fungsi aktivasi non-linear.

4. Fully Connected Layers

Relu digunakan setelah setiap lapisan konvolusi untuk memperkenalkan non linearitas ke dalam model yang sangat penting untuk mempelajari pola yang kompleks. Ini memperbaiki peta fitur untuk memastikan peta fitur selalu positif. Peta fitur dari lapisan pooling maksimum terakhir kemudian diratakan menjadi vektor. Vektor ini kemudian diproses oleh dua lapisan terhubung penuh. Lapisan terhubung penuh ini menggunakan bobot untuk memetakan input ke output.

5. SoftMax Activation

Aktivasi SoftMax digunakan pada lapisan terakhir untuk mendapatkan probabilitas gambar yang termasuk dalam 1000 kelas. Output dari lapisan terhubung penuh terakhir adalah vektor probabilitas. Probabilitas ini menunjukkan kemungkinan input milik setiap kelas. Kelas dengan probabilitas tertinggi adalah kelas yang diprediksi untuk gambar input.

6. Deconvolution Layers

ZFNet memperkenalkan teknik visualisasi yang melibatkan lapisan dekonvolusional (Lapisan Transposisi). Lapisan ini memberikan wawasan tentang apa yang telah dipelajari jaringan dengan memproyeksikan aktivasi fitur kembali ke ruang piksel masukan.

3.2.11. Output

Setelah proses deteksi berhasil, Output yang dimunculkan adalah *bounding box* objek yang dideteksi, nama kelas, dan penghitungan jumlah dari objek yang dideteksi.

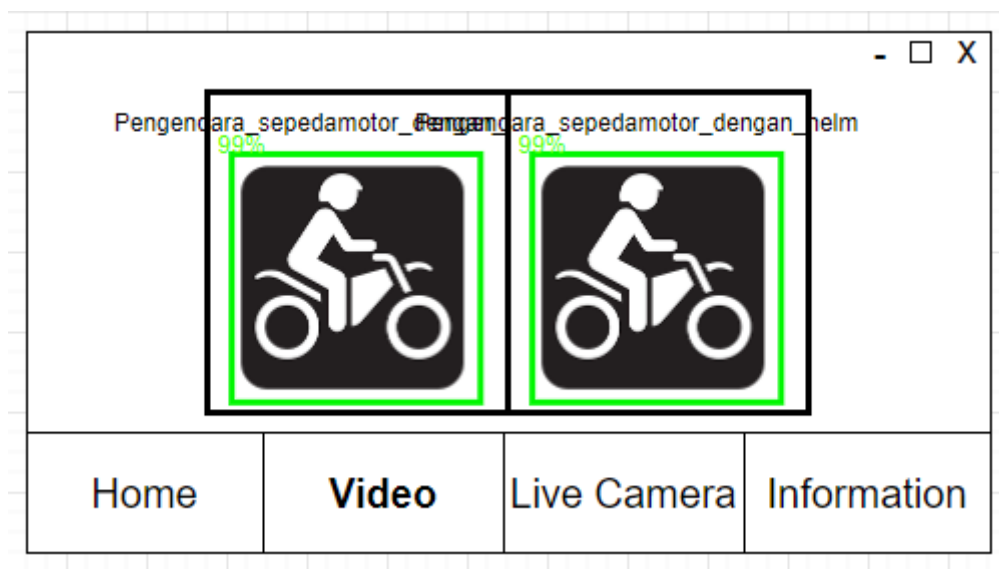
3.3. Perancangan Antarmuka Sistem

Terdapat 4 *section* pada antarmuka sistem, yaitu tab *home*, *video*, *live camera*, dan *information*. Tab Home ini dapat dilihat pada gambar 3.9.



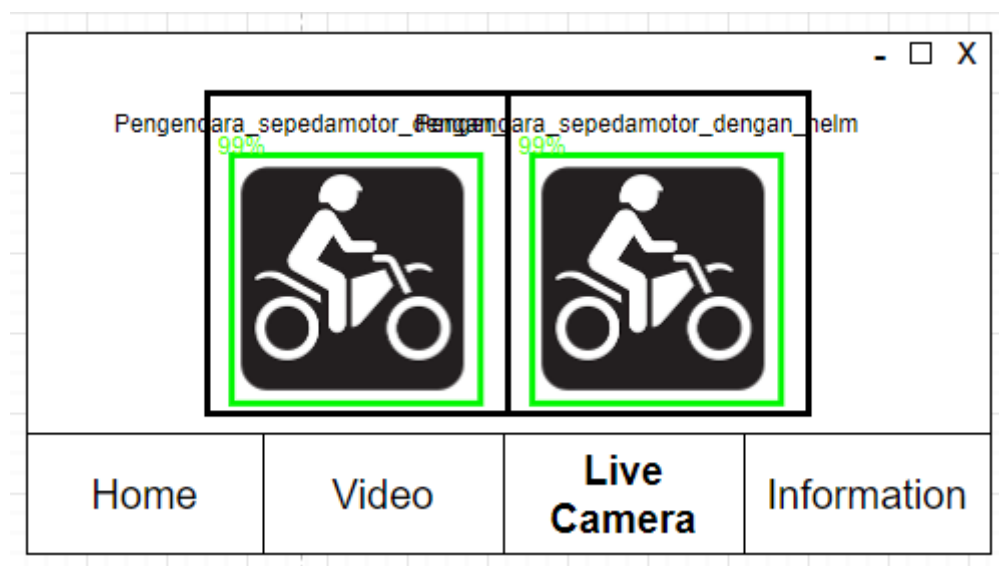
Gambar 3.9 Tampilan Antarmuka Sistem (*Home*)

Tampilan *Home* ini adalah beranda dari aplikasi. Terdapat informasi mengenai penulis. Selanjutnya tampilan video dapat dilihat pada gambar 3.10.



Gambar 3.10 Tampilan Video

Dalam tampilan ini aplikasi memproses data berupa video untuk dideteksi. Selanjutnya tampilan *live camera* dapat dilihat pada gambar 3.11.



Gambar 3.11 Tampilan *Live Camera*

Tab ini memproses citra yang dilakukan secara *live*. Di akhir ada tampilan *information* yang dapat dilihat pada gambar 3.12.



Gambar 3.12 Tampilan *Information*

Pada gambar 3.12. terdapat penjelasan informasi dari aplikasi ini sendiri.

BAB 4

IMPLEMENTASI DAN HASIL PENELITIAN

4.1. Spesifikasi Kebutuhan Aplikasi

Pengujian dilakukan dengan menggunakan *hardware* & perangkat lunak sebagai berikut:

4.1.1. Kebutuhan Perangkat Keras

Perangkat keras yang akan digunakan adalah :

1. Laptop dengan spesifikasi :
 - a. Merk : Asus
 - b. Prosesor : Intel(R) Core (TM) i5-9300H CPU @ 2.40GHz (8 CPUs), ~2.4GHz
 - c. Memori : 8GB DDR5 on board
 - d. *Storage* : 1 TB HDD
128 GB SSD M2 NVME

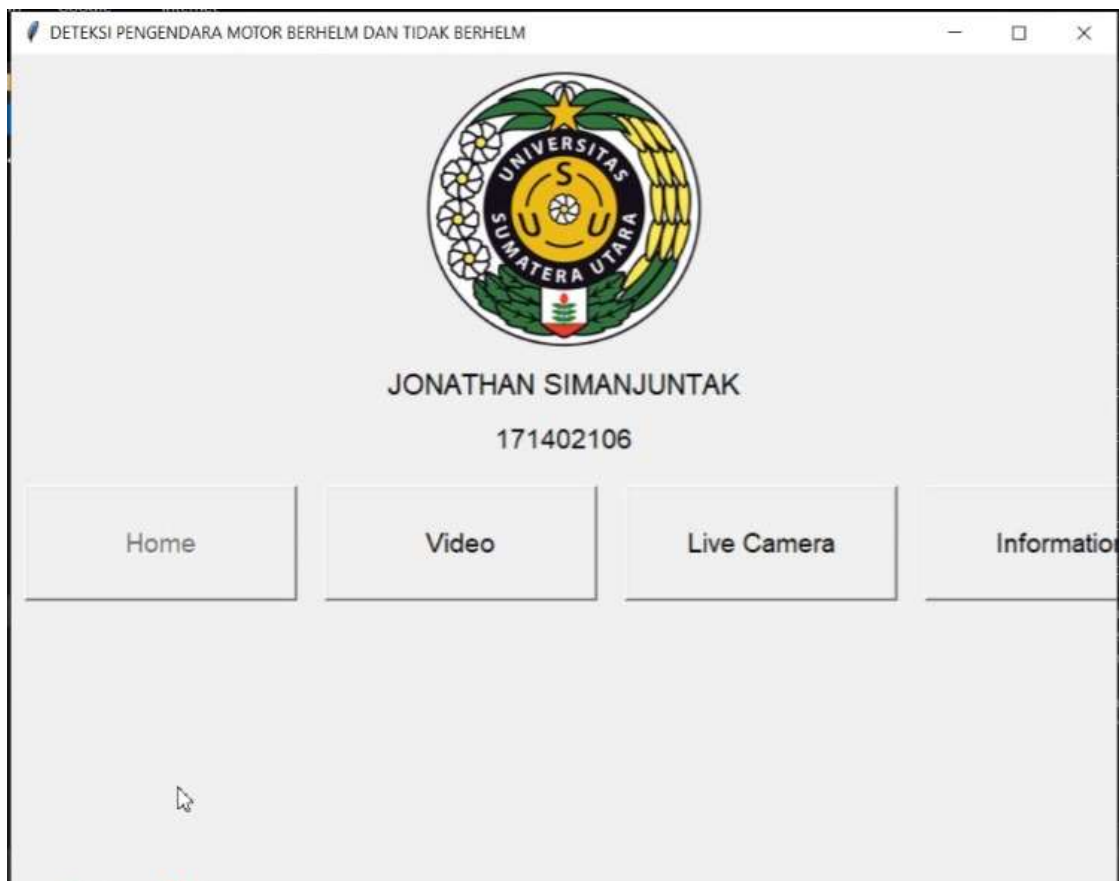
4.1.2. Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan adalah :

1. Sistem Operasi : Windows 10 Pro 64-bit
2. Bahasa Program : Python 3.9.2
3. *Library* : Tensorflow 2.9.0
Opencv 4.7.0.72
Tflite 2.10
Pandas 1.4.2
4. *Text Editor* : Visual Studio Code
Google Colab

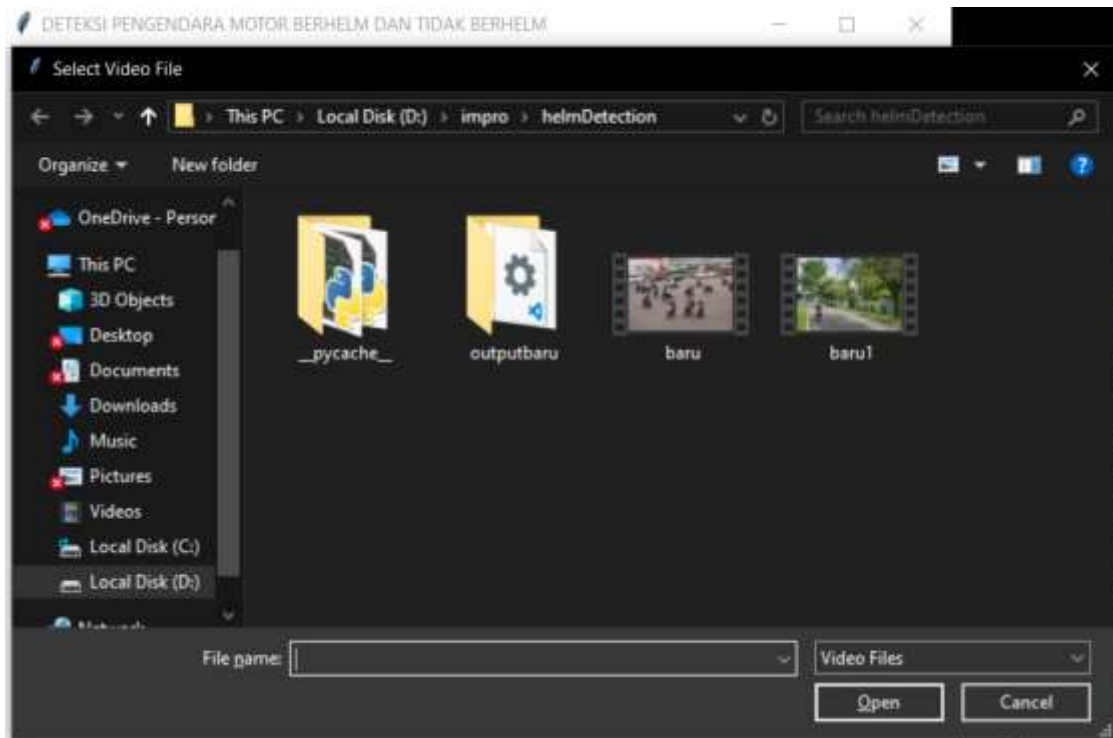
4.2. Implementasi Sistem

Implementasi sistem dimulai dari program dijalankan pada terminal anaconda prompt, dengan memanggil program. Sebuah jendela akan muncul lalu video berjalan. Selanjutnya program akan menampilkan *bounding box* dan menampilkan nilai akurasi serta prediksi dari suatu objek yang sedang berjalan.



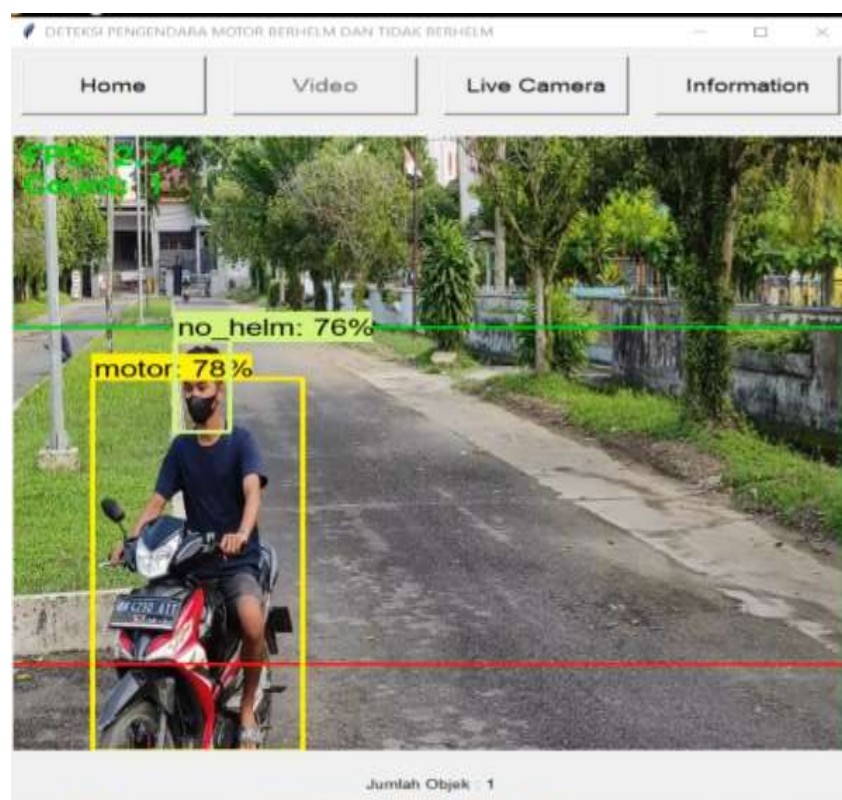
Gambar 4.1 Tampilan program (*Home*)

Pada gambar 4.1 terdapat tampilan *home*. Tampilan ini berjalan ketika program pertama kali dijalankan. Jika menekan tombol video, akan muncul pop up untuk memilih file video. Hal ini dapat dilihat pada gambar 4.2.



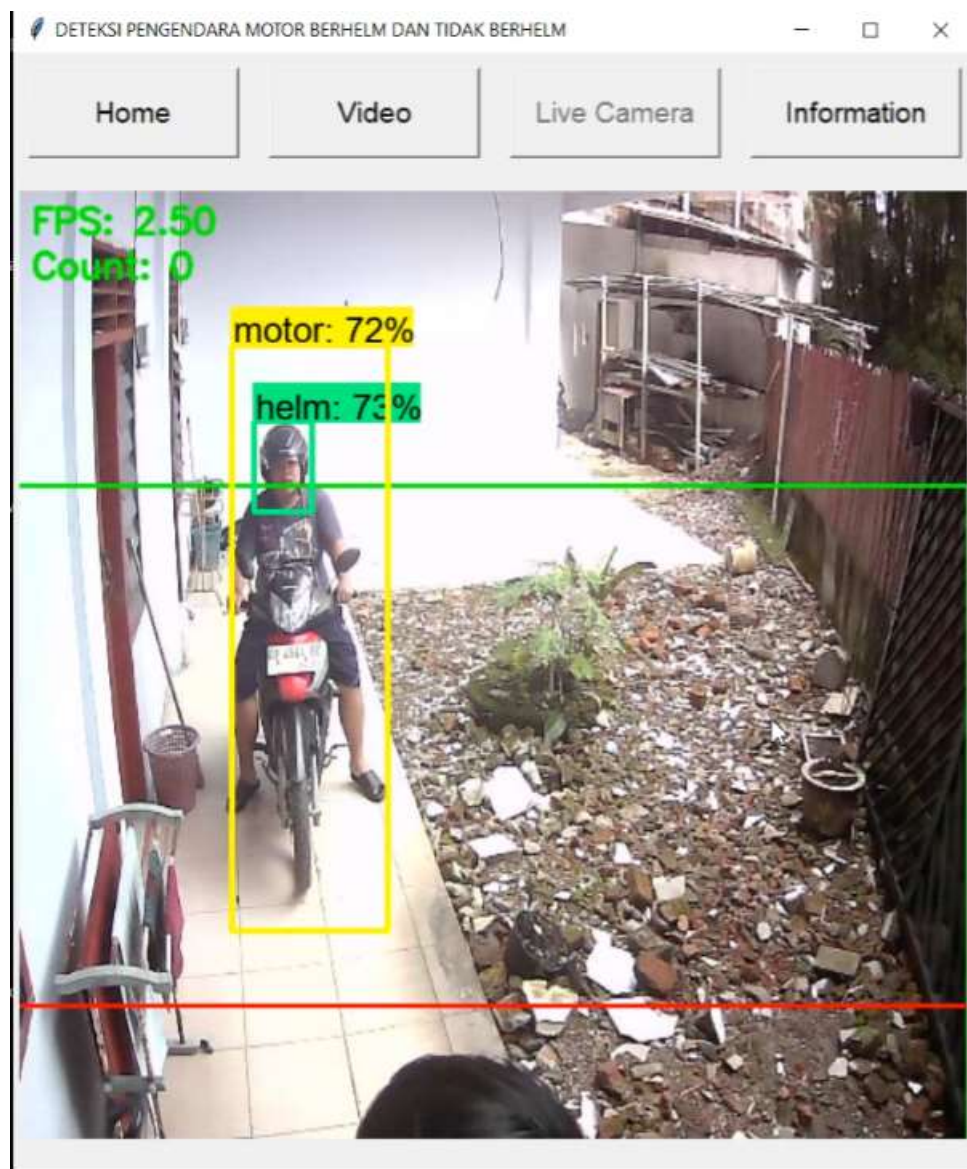
Gambar 4.3 Tampilan program (*Video*)

Selanjutnya pengujian dicoba pada tampilan video pada gambar 4.3.



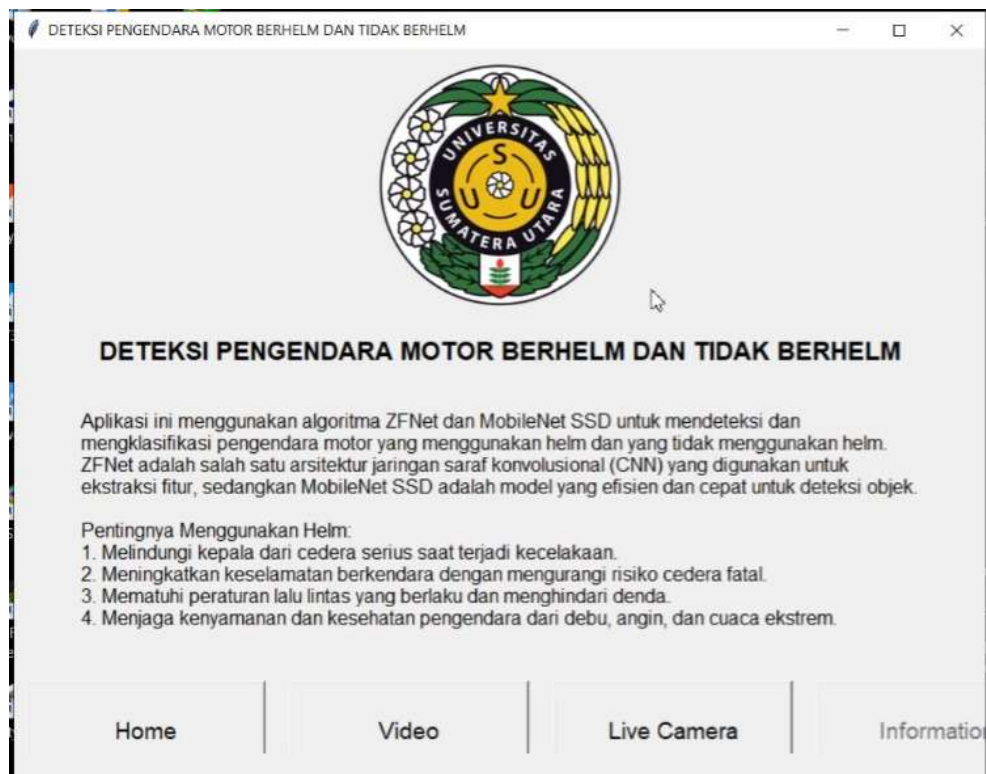
Gambar 4.3 Tampilan program berjalan (*Video*)

Pada gambar 4.3, ketika program dijalankan muncul pop up dari aplikasi berbentuk desktop yang dibuat oleh OpenCV. Pada proses ini, video yang sudah dipilih terus berjalan hingga diberhentikan secara manual. Setiap terdapat objek kelas dalam video, sistem akan menampilkan bounding box berwarna kuning yang meliputi seluruh komponen objek. Selain bounding box, keterangan nama objek berupa helm, motor ditampilkan di atas bounding box dan threshold atau IoU ditampilkan dalam bentuk persen di sebelah keterangan nama objek. Keterangan fps juga diperlihatkan di sini.



Gambar 4.4 Tampilan program (*Live Camera*)

Gambar 4.4 menunjukkan tampilan program ketika menggunakan live camera.



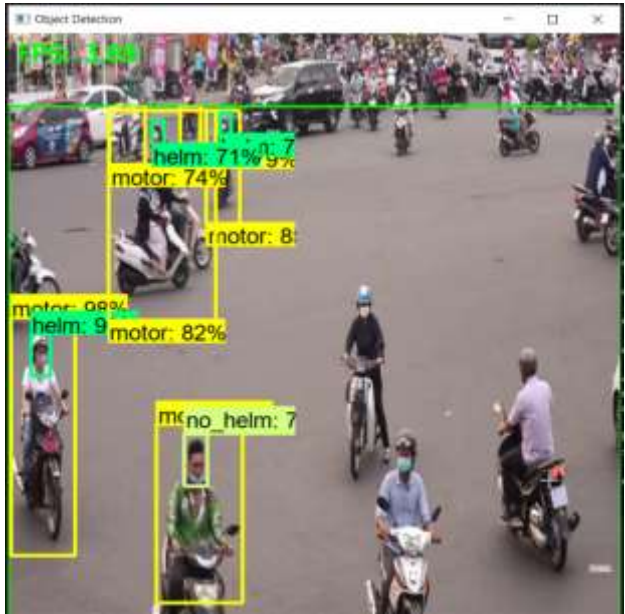
Gambar 4.5 Tampilan program (*Information*)




Gambar 4.5 menunjukkan informasi mengenai aplikasi yang dijalankan.


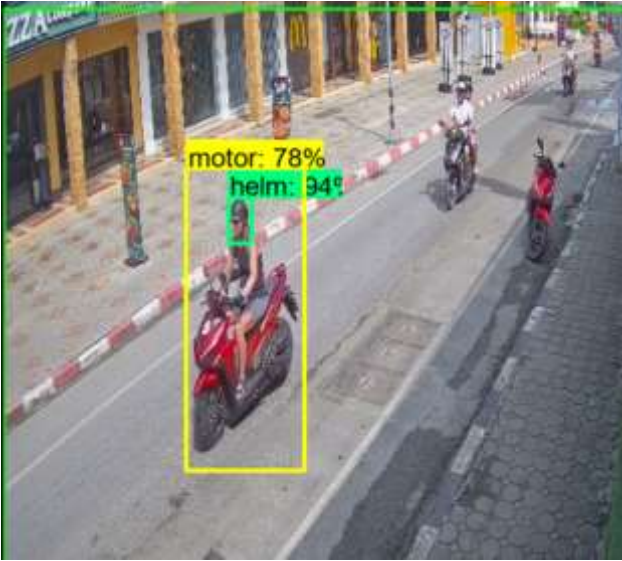
4.3. Hasil Uji Sistem



Uji kinerja model SSD sebagai backbone dan ZFNet sebagai pendeteksi objek akan dilakukan dengan menganalisis confusion matrix, recall, precision, dan juga F1 score dari model yang telah dilatih. Berikut hasil uji dari sistem yang di-deploy dengan threshold ≥ 0.5 .



Tabel 4.1 Hasil Uji Sistem



No	Citra	Actual Output	Desired Output	Keterangan
1		<p>motor: 4 helm: 2 no_helm: 1</p>	<p>motor: 6 helm: 3 no_helm: 2</p>	<p>Pada frame ini terdapat 2 motor yang tidak berhasil terdeteksi dan terdapat 1 motor helm dan 1 motor no_helm yang terdeteksi. Objek yang lain tidak terdeteksi dikarenakan objek terlalu banyak dalam frame.</p>



2		motor: 1 helm: 1	motor: 1 helm: 1	Model berhasil memprediksi i objek motor dan helm
3		motor: 1 helm: 1	motor: 1 helm: 1	Model berhasil memprediksi i objek motor dan helm
4		motor: 2 helm: 1	motor: 2 helm: 2	Model berhasil memprediksi i objek motor dan helm. Tetapi <i>miss</i> pada helm satu lagi karena jarak yang terlalu jauh

5		motor: 2 helm: 1	motor: 1 helm: 1	Model berhasil memprediksi objek motor dan helm. Tetapi sistem mendeteksi motor satu lagi padahal hanya ada 1.
6		motor: 1 helm: 1	motor: 5 helm: 3	Pada frame ini terdapat 4 motor yang tidak berhasil terdeteksi dan terdapat 2 helm yang tidak terdeteksi karena jarak yang terlalu jauh(> 5 m)




7		motor: 2 no helm: 1	motor: 4 helm: 2	<p>Pada frame ini terdapat 4 motor yang tidak berhasil terdeteksi dan terdapat 1 helm yang tidak terdeteksi karena jarak > 5 m. Sistem mendeteksi 1 motor menjadi dua motor (<i>double bounding box</i>)</p>
8		motor: 1 helm: 1	motor: 1 helm: 1	<p>Sistem berhasil mendeteksi kan seluruh objek dalam citra</p>



9		motor: 1 no helm: 1	motor: 1 no helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra
10		motor: 1 helm: 1	motor: 1 helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra



11		motor: 1 no helm: 1	motor: 1 no helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra
12		motor: 1 helm: 1	motor: 1 helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra



13		motor: 1 helm: 1	motor: 1 helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra
14		motor: 1 no helm: 1	motor: 1 no helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra



15		motor: 1 no helm: 1	motor: 1 no helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra
16		motor: 1 helm: 1	motor: 1 helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra



17		motor: 1 no helm: 1	motor: 1 no helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra
18		motor: 1 helm: 1	motor: 1 helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra
19		motor: 1 helm: 1	motor: 1 helm: 1	Sistem berhasil mendeteksi kan seluruh objek dalam citra


20		<p>motor: 1 no helm: 1</p>	<p>motor: 1 no helm: 1</p>	<p>Sistem berhasil mendeteksi kan seluruh objek dalam citra</p>
21		<p>motor: 0 helm: 0 no helm: 0</p>	<p>motor: 3 helm: 5 no helm: 1</p>	<p>Model gagal mendeteksi dikarenakan objek terlalu dekat & terlalu banyak</p>

22		motor: 0 helm: 0	motor: 1 helm: 2	Model gagal mendeteksi dikarenakan <i>environment</i> yang terlalu gelap
23		motor: 0 helm: 0	motor: 3 helm: 7	Model gagal mendeteksi dikarenakan objek terlalu dekat & terlalu banyak

24		<p>motor: 0 helm: 0 no helm: 0</p>	<p>motor: +- 25 helm: +- 30</p>	<p>Model gagal mendeteksi dikarenakan objek terlalu banyak</p>
25		<p>motor: 0 helm: 0</p>	<p>motor: 0 helm: 0</p>	<p>Model gagal mendeteksi dikarenakan model mendeteksi sepeda sebagai motor</p>

26		motor: 2 helm: 0	motor: 4 helm: 4	Model memprediks i 2 motor dan gagal mendeteksi sisa motor dan helm karena kondisi malam.
27		motor: 0 helm: 0	motor: 1 helm: 1	Model gagal memprediks i karena kondisi malam hari

28		motor: 0 helm: 1	motor: 0 helm: 1	Model mendeteksi helm pada pengguna bersepeda.
29		motor: 1 helm: 0	motor: 2 helm: 0	Model mendeteksi 1 motor dan gagal mendeteksi 1 motor lagi karena kondisi hujan.

30		Motor: 1 Helm: 0	Motor: 3 Helm: 2	Pada percobaan ini,model hanya mendeteksi 1 motor,model gagal mendeteksi motor yang lain dikarenakan kondisi hujan
----	---	---------------------	---------------------	--

Penulis melakukan percobaan sebanyak 100 kali untuk mengetahui apakah model yang dibuat cocok atau tidak. Hasil dari percobaan ini dapat dilihat pada tabel 4.2.

Tabel 4.2 Hasil Seluruh Percobaan

Percobaan Ke-	True Positive	True Negative	False Positive	False Negative
1	3	0	2	0
2	1	1	0	0
3	2	0	0	0
4	2	0	0	0
5	2	0	0	0
6	2	0	0	0
7	2	0	0	0
8	2	0	0	0
9	2	0	0	0
10	2	0	0	0

Percobaan Ke-	True Positive	True Negative	False Positive	False Negative
11	2	0	0	0
12	2	0	0	0
13	2	0	0	0
14	2	0	0	0
15	2	0	0	0
16	2	0	0	0
17	3	0	2	0
18	1	1	0	0
19	3	0	2	0
20	3	0	2	0
21	3	0	2	0
22	3	0	2	0
23	3	0	2	0
24	3	0	2	0
25	1	1	0	0
26	1	1	0	0
27	1	1	0	0
28	1	1	0	0
29	1	1	0	0
30	3	0	2	0
31	3	0	2	0
32	4	0	0	0
33	1	0	2	0
34	1	1	1	1
35	2	0	0	0
36	2	0	0	0
37	2	0	0	0
38	2	0	0	0
39	2	0	0	0
40	2	0	0	0

Percobaan Ke-	True Positive	True Negative	False Positive	False Negative
41	2	0	0	0
42	2	0	0	0
43	2	0	0	0
44	2	0	0	0
45	2	0	0	0
46	2	0	0	0
47	4	1	0	0
48	4	1	0	0
49	4	1	0	0
50	2	0	0	0
51	2	0	0	0
52	2	0	0	0
53	4	1	0	0
54	4	1	0	0
55	2	0	0	0
56	2	0	0	0
57	2	0	0	0
58	2	0	0	0
59	2	0	0	0
60	4	1	0	0
61	1	1	0	0
62	1	1	0	0
63	1	1	0	0
64	3	0	2	0
65	3	0	2	0
66	1	1	0	0
67	1	1	0	0
68	1	1	0	0
69	3	0	2	1
70	1	1	0	1

Percobaan Ke-	True Positive	True Negative	False Positive	False Negative
71	2	0	0	1
72	2	0	0	0
73	2	0	0	1
74	1	1	0	0
75	1	1	0	0
76	1	1	0	0
77	1	1	0	0
78	1	1	0	0
79	1	1	0	0
80	2	0	0	1
81	2	0	0	0
82	1	1	0	1
83	2	0	0	1
84	2	0	0	0
85	2	0	0	0
86	4	1	0	0
87	1	1	0	0
88	1	1	0	0
89	1	1	0	0
90	3	0	2	0
91	1	1	0	0
92	1	1	0	0
93	1	1	0	0
94	1	1	0	0
95	1	1	0	0
96	2	0	0	0
97	2	0	0	0
98	2	0	0	0
99	2	0	0	0
100	2	0	0	0

Percobaan Ke-	True Positive	True Negative	False Positive	False Negative
Total	199	37	31	8

Berdasarkan tabel 4.2 didapat nilai TP, TN, FP, FN dari 100 kali percobaan:

- a. True Positive : 199
- b. True Negative : 37
- c. False Positive : 31
- d. False Negative : 8

Kemudian dilakukan perhitungan untuk mencari nilai *precision*, *recall*, *accuration*, dan, *F1 – Score* agar dapat ditentukan seberapa besar kemampuan dari algoritma MobileNet SSD dan ZFNet. Berikut penghitungan untuk mencari nilai *precision*, *recall*, *accuration*, dan, *F1 – Score*.

$$Precision = \frac{TP}{TP + FP} = \frac{199}{199 + 31} = 0.87 \text{ atau } 87\%$$

$$Recall = \frac{TP}{TP + FN} = \frac{199}{199 + 8} = 0.96 \text{ atau } 96\%$$

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} = \frac{199 + 37}{199 + 37 + 31 + 8} = 0.86 \text{ atau } 86\%$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = 0.91 \text{ atau } 91\%$$

Dari hasil perhitungan di atas, didapat bahwa nilai *precision* sebesar 0.87 atau 87%, *recall* sebesar 0.96 atau 96%, akurasi sebesar 0.86 atau 86%, dan F1-Score sebesar 0.91 atau 91%. Sehingga dapat disimpulkan bahwa model ZFNet dan MobileNet SSD terbukti cocok jika ingin digunakan dalam pengdeteksian pemotor dengan helm dan tidak.

BAB 5

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Setelah dilakukan implementasi, didapat kesimpulan bahwa:

1. Penelitian ini menggunakan algoritma ZFNet dan MobileNet SSD. Aplikasi dalam penelitian ini berhasil berjalan dengan baik. Algoritma ZFNet dan MobileNet SSD berhasil berjalan pada kondisi siang hari dengan mengabaikan situasi lingkungan dan dengan jarak kurang dari 5 meter. Model ini memberi nilai *precision* sebesar 0.87 atau 87%, *recall* sebesar 0.96 atau 96%, akurasi sebesar 0.86 atau 86%, dan F1-Score sebesar 0.91 atau 91%.
2. Dengan nilai F1 – Score sebesar 91% menunjukkan bahwa model ZFNet dan MobileNet SSD terbukti cocok jika ingin digunakan dalam pengdeteksian pengendara sepeda motor dengan menggunakan helm atau tidak menggunakan helm.

5.2. Saran

1. Sistem yang di-*deploy* pada laptop memiliki kekurangan dalam kemudahan instalasi dan portabilitas. Penelitian selanjutnya diharapkan dapat menggunakan mini PC atau perangkat ARM sejenis dan menggunakan perangkat keras grafis yang lebih baik seperti Movidius NCS, Google Edge TPU, atau Nvidia Jetson.
2. Memperbanyak jumlah citra dan variasinya, variasi yang dimaksud seperti sudut pengambilan gambar/video, resolusi citra dan waktu pengambilan citra (siang atau malam hari). Hal ini ditujukan agar model yang akan dibangun dapat mengenali lebih banyak fitur dari objek dalam berbagai macam situasi.

DAFTAR PUSTAKA

- Abidin, S. 2021. Deteksi Wajah Menggunakan Metode Haar Cascade Classifier Berbasis Webcam Pada Matlab. Jurnal Teknologi Elekterika e-ISSN 2656-0143 21 No.1, Vo. 15.
- Ardi, M. T. B. Z., Setyawan, G. E. & Utaminingrum, F. 2020. Implementasi Algoritma Hough transform Pada Object Following Menggunakan Ar.Drone Quadcopter. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X Vol. 4, No. 1, Januari 2020, hlm. 134-141.
- Arya, S., Pratap, N., & Bhatia, K., 2015. *Future of Face Recognition: A Review*.
- Cahu S., Banjarnahor J., Irfansyah D., Kumala S. & Banjarnahor J., 2019. *Analisis Pendeteksian Pola Wajah Menggunakan Metode Haar-Like Feature*. Journal of Informatics and Telecommunication Engineering. (Online) Diakses 19 Juli 2019.
- Christyono, Y., Santoso, I. & Zahra, A. A. 2017. Perancangan Dan Pengukuran Kinerja Video Streaming Menggunakan Red5 Pada Mesin Virtual. Jurnal Transmisi, 19, (3), JULI 2017, e-ISSN 2407–6422, 139.
- Christyono, Y., Santoso, I., & Zahra, A. (2017). Perancangan Dan Pengukuran Kinerja Video Streaming Menggunakan Red5 Pada Mesin Virtual. *Jurnal Transmisi*, 19(3).
- Dewi, I. S. & Chandra, A. Y. 2021. Prototipe Sistem Deteksi Suhu Tubuh Dan Masker Wajah Menggunakan Algoritma Local Binary Pattern dan Arduino Nano. Jurnal Sistem Informasi Dan Bisnis Cerdas (SIBC) Vol. 14, No. 2 Agustus 2021.

- Gangopadhyay I., Chatterjee A. & Das I., 2018. *Face Detection And Recognition Using Haar Classifier And Lbp Histogram*. Journal of Advanced Research in Computer Science.
- Gonzalez, R. and Woods, R. (1992) Digital Image Processing. Addison Wesley, 5, 414-428.
- Gonzalez, R. C. (2009). *Digital image processing*. Pearson education india.
- Gonzalez, R.C. & Woods, R.E. 2003. Digital Image Processing. Second edition, USA: Addison-Wesley Publishing Co, University of Tennessee.
- Hatta, M. & Mahmudi, M. 2019. Penggunaan Algoritma Transformasi Hough Untuk Prototype Aplikasi Mobile Perhitungan Jumlah Pipa Pada Bagian Gudang PT Spindo Tbk. Jurnal Engineering and Sains ISSN 2579-5422 online Volume 3, Nomor 2, Desember 2019, 89-94
- Israni, S., & Jain, S. (2016, March). Edge detection of license plate using Sobel operator. In *2016 international conference on electrical, electronics, and optimization techniques (ICEEOT)* (pp. 3561-3563). IEEE.
- Irianto, K. D., & Ariyanto, G. (2009). Motion Detection Using Opencv With Background Subtraction and Frame Differencing Technique.
- Kaddoun, Said Si, et al. "Convolutional Neural Algorithm for palm vein recognition using ZFNet architecture. " 2021 4th International Conference on Bio Engineering for Smart Technologies (BioSMART). IEEE, 2021.
- Kementerian Perhubungan. (2020). 5 Ribu Pengendara Motor Tewas Tak Gunakan Helm, Masih Mau Nekat?
- Kurniawati, L. (2019). Pemanfaatan Teknologi Video Streaming di LPP TVRI Jawa Barat. *Jurnal Komunikasi*, 10(1), 10-18.

- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep CNN. *Advances in neural information processing systems*, 25.
- Lanes, Y. T. A. (2019). Perancangan dan Implementasi Aplikasi Sistem Pemantauan Helm Pendeteksi Kantuk Berbasis Website. *Jurnal Elektro dan Telekomunikasi Terapan (e-Journal)*, 6(2), 765-775.
- Liantoni, F. (2015). Deteksi tepi citra daun mangga menggunakan algoritma Ant Colony Optimization. In *Seminar Nasional Sains dan Teknologi Terapan III* (Vol. 3, pp. 411-418).
- Ramadhan, M. P. (2023). *Deteksi Objek Robot Sepakbola secara Real Time dengan Mobilenet Single Shot Detector (SSD)* (Doctoral dissertation, Universitas Sumatera Utara).
- Negara, C. U. P & Firdausy, K. 2019. Pengolahan Citra menggunakan Metode Otsu dan Hough Circle Transform untuk Prototipe Alat Sortir Buah Apel. Seminar Nasional Aplikasi Teknologi Informasi (SNATi) Yogyakarta, 03 Agustus 2019.
- Pandriantama, H. & Sanjaya, M 2018. Aplikasi Transformasi Hough Pada Robot Vision Lane Tracking. *Al Jazari Journal of Mechanical Engineering*.
- Panetta, K., Gao, C., & Agaian, S. (2015). Human-visual-system-inspired underwater image quality measures. *IEEE Journal of Oceanic Engineering*, 41(3), 541-551.
- Perhubungan, K., & INDONESIA, R. (2020). Aman Bertransportasi Di Masa Pandemi?: Tantangan, Strategi, dan Kebijakan. *Webinar. Kementerian Perhubungan Republik Indonesia*.
- Pratiwi, Hanissa Anggraini, Margi Cahyanti, and Missa Lamsani. "Implementasi Deep Learning Flower Scanner Menggunakan Metode Convolutional Neural Network." *Sebatik* 25.1 (2021): 124-130.

- R. A. Asmara, “Pengolahan Citra Digital – Teori, Praktek, dan Latihan - Latihan,” Malang: POLINEMA PRESS, 2018.
- S. Abdul Kadir. Teori dan Aplikasi Pengolahan Citra. Yogyakarta: Andi Publisher, 2013.
- Sthevanie, F., Kurniawan, A., & Ramadhani, K. N. (2020). Deteksi Helm pada Video Pengendara Sepeda Motor menggunakan Ekstraksi Ciri Histogram of Oriented Gradients. *Indonesia Journal on Computing (Indo-JC)*, 5(1), 63-72.
- Sugianto, Setyati, E. & Armanto, H. 2019. Deteksi Alat Pelindung Kepala (Helm) Menggunakan Metode Haar Cascade Classifier. Jurnal JOUTICA Volume 4 No.1 2019.
- Syafeeza, A. R., Khalil-Hani, M., Liew, S. S., & Bakhteri, R. (2014). Convolutional neural network for face recognition with pose and illumination variation. *International Journal of Engineering & Technology*, 6(1), 0975 4024. Symposium on Computer Vision and the Internet.
- Utami, F., Suhendri, S., & Mujib, M. A. (2021). Implementasi Algoritma Haar Cascade Pada Aplikasi Pengenalan Wajah Personel. *Journal of Information Technology*, 3(1), 33-38.
- Wibowo, B. C., Nugraha, F. & Utomo, A. P. 2021. Uji Deteksi Objek Bentuk Bola Dengan Menerapkan Metode Circular Hough Transform. Jurnal Informatika UPGRIS Vol. 7, No. 1 JUNI 2021.
- Widodo, B., Armanto, H., & Setyati, E. (2021). Deteksi Pemakaian Helm Proyek Dengan Metode Convolutional Neural Network. *INSYST: Journal of Intelligent System and Computation*, 3(1), 23-29.
- Yogi, M (2016). Aplikasi Kematangan Buah Semangka Berbasis Nilai RGB Menggunakan Metode Thresholding. *JURIKOM (Jurnal Riset Komputer)*, 3(6).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I 13* (pp. 818–833). Springer International Publishing.