



TUGAS AKHIR - EC184801

**DETEKSI PEJALAN KAKI PADA ZEBRACROSS
UNTUK PERINGATAN DINI PENGENDARA
MOBIL MENGGUNAKAN MASK R-CNN**

**Agung Wicaksono
NRP 0721 17 4000 0002**

**Dosen Pembimbing
Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Dr. Eko Mulyanto Yuniarno, S.T., M.T.**

**DEPARTEMEN TEKNIK KOMPUTER
Fakultas Teknologi ELEKTRO DAN INFORMATIKA CERDAS
Institut Teknologi Sepuluh Nopember
Surabaya 2021**



TUGAS AKHIR - EC184801

**DETEKSI PEJALAN KAKI PADA ZEBRACROSS
UNTUK PERINGATAN DINI PENGENDARA
MOBIL MENGGUNAKAN MASK R-CNN**

**Agung Wicaksono
NRP 0721 17 4000 0002**

**Dosen Pembimbing
Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Dr. Eko Mulyanto Yuniarno, S.T., M.T.**

**DEPARTEMEN TEKNIK KOMPUTER
Fakultas Teknologi ELEKTRO DAN INFORMATIKA CERDAS
Institut Teknologi Sepuluh Nopember
Surabaya 2021**

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir sada dengan judul "**Deteksi Pejalan Kaki pada Zebracross untuk Peringatan Dini Pengendara Mobil menggunakan Mask R-CNN**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 14 Juni 2021

Agung Wicaksono
0721 17 4000 0002

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DETEKSI PEJALAN KAKI PADA ZEBRACROSS UNTUK PERINGATAN DINI PENGENDARA MOBIL MENGGUNAKAN MASK R-CNN

Tugas Akhir ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Teknik di Institut Teknologi Sepuluh Nopember Surabaya

Oleh: Agung Wicaksono (NRP. 0721 17 4000 0002)

Tanggal Ujian : Juli 2021
Periode Wisuda : September 2021

Disetujui Oleh:

Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng. (Pembimbing I)
NIP: 19580916 198601 1 001

Dr. Eko Mulyanto Yuniarso, S.T., M.T. (Pembimbing II)
NIP: 19680601 199512 1 009

Mengetahui,
Kepala Departemen Teknik Komputer FTEIC - ITS

Dr. Supeno Mardi Susiki Nugroho, ST., MT.
NIP. 19700313 199512 1 001

[Halaman ini sengaja dikosongkan]

ABSTRAK

Nama Mahasiswa : Agung Wicaksono
Judul Tugas Akhir : Deteksi Pejalan Kaki pada *Zebracross* untuk Peringatan Dini Pengendara Mobil menggunakan *Mask R-CNN*
Pembimbing : 1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
 2. Dr. Eko Mulyanto Yuniarno, S.T., M.T.

Dewasa ini, fitur keselamatan pada kendaraan roda empat atau mobil sudah sangat berkembang pesat. Hal tersebut terbukti dengan banyaknya produsen mobil yang menerapkan teknologi seat belt, air bag, adaptive cruise control, electronic stability control, autonomous emergency braking, blind spot monitoring dan lain sebagainya. Namun, fitur yang sudah disebutkan diatas dinilai masih kurang ramah bagi pejalan kaki. Terbukti menurut data dari WHO, terdapat 270.000 pejalan kaki meninggal dunia setiap tahun atau sekitar 22% dari seluruh korban meninggal akibat kecelakan di jalan. Berawal dari permasalahan tersebut, penulis akan melakukan penelitian mengenai pendekripsi pejalan kaki pada zebracross untuk peringatan dini pengendara mobil sebagai topik tugas akhir. Pada tugas akhir ini, terdapat 2 objek yang akan dideteksi yaitu pejalan kaki dan zebracross dengan menggunakan metode Mask R-CNN. Hasil yang diharapkan dari tugas akhir kali ini adalah terdapat model yang memiliki akurasi yang tinggi dari dataset yang tersedia yaitu Caltech Pedestrian Dataset.

Kata Kunci: Pejalan Kaki, *Zebracross*, Mask R-CNN, Pengolahan Citra.

[Halaman ini sengaja dikosongkan]

ABSTRACT

*Name : Agung Wicaksono
Title : Pedestrian Detection on Zebracross for Car Driver Early Warning using Mask R-CNN
Advisors : 1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
2. Dr. Eko Mulyanto Yuniaro, S.T., M.T.*

Today, the safety features on four-wheeled vehicles or cars have developed very rapidly. This is evidenced by the number of car manufacturers that apply seat belt technology, airbags, adaptive cruise control, electronic stability control, autonomous emergency braking, blind spot monitoring and so on. However, the features mentioned above are still considered less friendly for pedestrians. According to WHO data, there are 270,000 pedestrians who die every year or about 22% of all victims die due to road accidents. Starting from these problems, the author will conduct research on the detection of pedestrians at zebracross for early warning car drivers as the topic of the final project. In this final project, there are 2 objects to be detected, namely pedestrians and zebracross using the Mask R-CNN method. The expected result of this final project is that there is a model that has high accuracy from the available datasets, namely the Caltech Pedestrian Dataset.

Keywords: Pedestrian, Zebracross, Mask R-CNN, Image Processing

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT atas segala limpahan berkah, rahmat, serta ridho-Nya, penulis dapat menyelesaikan penelitian ini dengan judul **Deteksi Pejalan Kaki pada Zebra-cross untuk Peringatan Dini Pengendara Mobil menggunakan Mask R-CNN**.

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Departemen Teknik Komputer ITS, sera digunakan sebagai persyaratan menyelesaikan pendidikan Sarjana. Penelitian ini dapat diselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terimakasih kepada:

1. Keluarga, Ibu, Bapak dan Saudara tercinta yang telah memberikan dorongan baik secara spiritual dan material dalam penyelesaian buku penelitian ini.

Surabaya, Juni 2021

Agung Wicaksono

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

ABSTRAK	i
ABSTRACT	iii
KATA PENGANTAR	v
DAFTAR ISI	vii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiii
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Permasalahan	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Sistematika Penulisan	3
2 TINJAUAN PUSTAKA	5
2.1 Dasar Teori	5
2.1.1 <i>Artificial Intelligence</i>	5
2.1.2 <i>Machine Learning</i>	5
2.1.3 <i>Deep Learning</i>	7
2.1.4 <i>Convolutional Neural Network</i>	8
2.1.5 <i>Convolutional Layer</i>	9

2.1.6	<i>Pooling Layer</i>	13
2.1.7	Fungsi Aktivasi (<i>Non-Linearity</i>)	15
2.1.8	Fully Connected Layer	17
2.1.9	<i>Loss Function</i>	17
2.2	Penelitian Terkait	18
2.2.1	Real-Time Pedestrian Detection With Deep Network Cascades	18
3	DESAIN DAN IMPLEMENTASI	21
3.1	Deskripsi Sistem	21
3.2	Pengumpulan <i>Dataset</i> Gambar	22
3.3	Pemisahan Data	23
3.4	<i>Pre-Processing</i>	24
3.5	Membangun Model Mask R-CNN	27
3.6	<i>Training Data</i>	28
3.7	<i>Validating Data</i>	29
3.8	<i>Testing Data</i>	30
4	PENGUJIAN DAN ANALISIS	33
4.1	Pengujian Jenis <i>Backbone</i>	34
4.1.1	Resnet 50	34
4.1.2	Resnet 101	34
4.1.3	MobileNet V1	35
4.1.4	MobileNet V2	35
4.2	Pengujian Jenis Teknik Validasi	35
4.2.1	Pengujian dengan Pemisahan Data Validasi	35
4.2.2	Pengujian dengan <i>K Fold Cross Validation</i>	35
5	PENUTUP	37

5.1	Kesimpulan	37
5.2	Saran	37
DAFTAR PUSTAKA		39
BIOGRAFI PENULIS		41

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

2.1	Gambaran <i>Supervised Learning</i> [1]	6
2.2	Gambaran <i>Unsupervised Learning</i> [2]	7
2.3	Gambaran <i>Reinforcement Learning</i> [3]	7
2.4	Gambaran Konsep Arsitektur CNN [4]	8
2.5	Contoh Gambar RGB dan Grayscale	10
2.6	Gambaran Operasi Konvolusi pada <i>Convolution Layer</i>	11
2.7	Gambaran Operasi Konvolusi pada <i>Convolution Layer</i> menggunakan <i>Zero Padding</i>	12
2.8	Gambaran Proses <i>Max Pooling</i>	14
2.9	Grafik Fungsi Aktivasi Sigmoid [5]	15
2.10	Grafik Fungsi Aktivasi Tanh [5]	16
3.1	Blok Diagram Metodologi	21
3.2	Contoh Gambar dari Caltech Pedestrian Database .	22
3.3	Contoh Pembuatan <i>dataset</i> dari <i>Screenshot Youtube</i>	23
3.4	Visualisasi Pembagian Data	23
3.5	Diagram Alir <i>Pre Processing</i>	25
3.6	Contoh <i>Image Resizing</i>	26
3.7	Contoh <i>Image Augmentation</i>	27
3.8	Blok Diagram Alur Mask R-CNN	28
3.9	Visualisasi <i>K Fold Cross Validation</i>	30
4.1	Gambaran Arsitekture ResNet50	34
4.2	Gambaran Hasil Deteksi ResNet50	34
4.3	Gambaran Arsitekture ResNet101	35

4.4 Gambaran Hasil Deteksi ResNet101	35
--	----

DAFTAR TABEL

4.1	Spesifikasi <i>hardware Google Colaboratory</i>	33
4.2	Spesifikasi <i>hardware Komputer yang Digunakan</i>	33

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

Penelitian ini di latar belakangi oleh berbagai kondisi yang menjadi acuan. Selain itu juga terdapat beberapa permasalahan yang akan dijawab sebagai luaran dari penelitian.

1.1 Latar Belakang

Mobil merupakan salah satu jenis kendaraan bermotor yang banyak terdapat di Indonesia. Pada tahun 2018 Badan Pusat Statistik mencatat terdapat 16.440.987 mobil penumpang yang berada di Indonesia. Dengan bertambahnya jumlah mobil di Indonesia dari tahun ke tahun, meningkatkan juga jumlah kecelakaan mobil. Fitur keselamatan dan keamanan pada mobil sangat penting bagi para pengendara dan penumpang, sehingga para produsen mobil berusaha meningkatkan teknologi keselamatan dan keamanan pada mobil buatannya. Sebagai contoh beberapa fitur keselamatan dan keamanan yang terdapat pada mobil antara lain, adaptive cruise control, hill strat assist, blind spot monitoring, electronic stability control dan lain sebagainya.

Menurut data dari WHO, terdapat 270.000 pejalan kaki meninggal dunia setiap tahun atau sekitar 22% dari seluruh korban meninggal akibat kecelakan di jalan. Melihat kegiatan para pejalan kaki yang jarang berada di badan jalan, angka tersebut tentu cukup tinggi. Para pejalan kaki hanya menggunakan badan jalan ketika hendak menyebrang jalan lewat zebrafloor. Kelalaian dari pejalan kaki maupun pengendara mobil merupakan faktor utama mengapa angka kematian pejalan kaki cukup tinggi. Salah satu contoh kelalaian pejalan kaki adalah pada saat menyebrang jalan tidak memperhatikan kendaraan yang akan lewat dan atau melihat rambu serta lampu lalu lintas. Di sisi pengendara mobil, kelelahan, kurangnya fokus saat berkendara dan tidak memperhatikan rambu maupun marka dapat berakibat fatal baik kepada pejalan kaki dan

pengendara lain.

Teknologi artificial intelligent sudah banyak disematkan pada mobil pada masa kini, dibuktikan dengan adanya teknologi adaptive cruise control, hill start assist dan lain sebagainya. Artificial intelligent khususnya deep learning tentu dapat digunakan untuk deteksi pejalan kaki di zebracross guna mengurangi jumlah korban akibat kecelakaan. Deteksi pejalan kaki dapat digabungkan dengan buzzer dan atau LED sebagai komponen output untuk mengingatkan kepada pengendara bahwa ada pejalan kaki yang sedang menyebrangi jalan serta mengbalikkan fokus untuk berkendara.

1.2 Permasalahan

Cukup tingginya angka kematian pejalan kaki akibat kecelakaan lalu lintas dan belum adanya deteksi pejalan kaki di zebracross untuk peringatan dini kepada pengendara mobil. Oleh karena itu, diperlukan sebuah sistem yang mampu mendeteksi adanya pejalan kaki yang berada disekitar jalan raya untuk selanjutnya dapat digunakan sebagai peringatan kepada pengendara mobil.

1.3 Tujuan

Berdasarkan rumusan permasalahan di atas, tujuan dari penelitian ini adalah untuk mendeteksi adanya pejalan kaki di zebracross untuk peringatan dini kepada pengendara mobil guna mengurangi angka kematian pejalan kaki akibat kecelakaan lalu lintas

1.4 Batasan Masalah

Batasan masalah yang timbul dari permasalahan Tugas Akhir ini adalah:

1. Menggunakan Mask R-CNN untuk pendekripsi pejalan kaki dan zebracross
2. File Input berupa video dengan format MP4 dengan 30 fps.

1.5 Sistematika Penulisan

Laporan penelitian tugas akhir ini tersusun dalam sistematika dan terstruktur sehingga mudah dipahami dan dipelajari oleh pembaca maupun seseorang yang ingin melanjutkan penelitian ini. Alur sistematika penulisan laporan penelitian ini yaitu :

1. BAB I Pendahuluan

Bab ini berisi uraian tentang latar belakang permasalahan, penegasan dan alasan pemilihan judul, sistematika laporan, tujuan dan metodologi penelitian.

2. BAB II Tinjauan Pustaka

Pada bab ini berisi tentang uraian secara sistematis teori-teori yang berhubungan dengan permasalahan yang dibahas pada penelitian ini. Teori-teori ini digunakan sebagai dasar dalam penelitian, yaitu informasi terkait pejalan kaki, *zebracross*, algoritma *Mask RCNN*, dan teori-teori penunjang lainnya.

3. BAB III Desain dan Implementasi Sistem

Bab ini berisi tentang penjelasan-penjelasan terkait eksperimen yang akan dilakukan dan langkah-langkah pengambilan data jalan raya serta proses deteksi pejalan kaki pada *zebracross*. Guna mendukung hal tersebut, digunakanlah blok diagram atau work flow agar sistem yang akan dibuat dapat terlihat dan mudah dibaca untuk implemtasi pada pelaksanaan tugas akhir.

4. BAB IV Pengujian dan Analisa

Bab ini menjelaskan tentang pengujian eksperimen yang dilakukan terhadap citra jalan raya, proses klasifikasi pejalan kaki dan *zebracross*. Serta terkait tingkat akurasi keberhasilan pengujian yang dilengkapi dengan analisanya.

5. BAB V Penutup

Bab ini merupakan penutup yang berisi kesimpulan yang diambil dari penelitian dan pengujian yang telah dilakukan. Saran dan kritik yang membangun untuk pengembangan lebih lanjut juga dituliskan pada bab ini.

[Halaman ini sengaja dikosongkan]

BAB II

TINJAUAN PUSTAKA

Demi mendukung penelitian ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan refensi. Dengan demikian penelitian ini menjadi lebih terarah.

2.1 Dasar Teori

2.1.1 *Artificial Intelligence*

Artificial Intelligence mengacu pada simulasi kecerdasan manusia dalam mesin yang diprogram untuk berpikir seperti manusia dan meniru tindakan mereka.[6] Istilah ini juga dapat diterapkan pada mesin apa pun yang menunjukkan ciri-ciri yang terkait dengan pikiran manusia seperti pembelajaran dan pemecahan masalah.

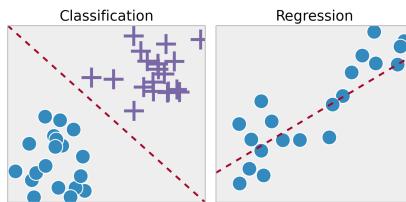
Karakteristik ideal dari *artificial intelligence* adalah kemampuannya untuk merasionalisasi dan mengambil tindakan yang memiliki peluang terbaik untuk mencapai tujuan tertentu. Bagian dari *artificial intelligence* adalah *machine learning*, yang mengacu pada konsep bahwa program komputer dapat secara otomatis belajar dari dan beradaptasi dengan data baru tanpa dibantu oleh manusia. Teknik *deep learning* memungkinkan pembelajaran otomatis ini melalui penyerapan sejumlah besar data tidak terstruktur seperti teks, gambar, atau video.

2.1.2 *Machine Learning*

Machine Learning adalah studi tentang algoritma komputer yang memberikan sistem kemampuan untuk belajar secara otomatis dan dapat meningkatkan kemampuan dari pengalaman yang sudah didapatkan [7]. Hal ini umumnya dilihat sebagai sub-bidang kecerdasan buatan. Algoritma pembelajaran mesin memungkinkan sistem membuat keputusan secara mandiri tanpa dukungan eksternal. Keputusan semacam itu dibuat dengan menemukan pola dasar

yang berharga dalam data yang kompleks. Berdasarkan pendekatan pembelajaran, jenis data *input* dan *output*, dan jenis masalah yang dipecahkan, ada beberapa kategori utama dari algoritma *machine learning supervised, unsupervised* dan *reinforcement learning*. Ada beberapa pendekatan hibrida dan metode umum lainnya yang menawarkan ekstrapolasi alami dari bentuk masalah pembelajaran mesin. Berikut merupakan penjelasan dari beberapa kategori utama dari algoritma *machine learning*:

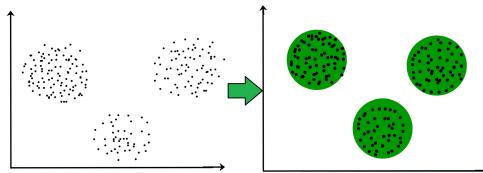
1. *Supervised Learning* diterapkan ketika data dalam bentuk variabel input dan nilai target output. Algoritma akan mempelajari fungsi pemetaan dari *input* ke *output*. Ketersediaan sampel data berlabel dengan skala besar mempunyai nilai yang tinggi dikarenakan masih terdapat kelangkaan *dataset*. Pendekatan ini secara luas dapat dibagi menjadi dua kategori utama yaitu *classification* dan *regression*. Gambar 2.1 menampilkan visualisasi dari *classification* dan *regression* pada *Supervised Learning*



Gambar 2.1: Gambaran *Supervised Learning*[1]

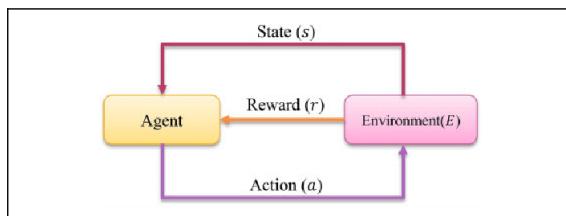
2. *Unsupervised Learning* diterapkan ketika data hanya tersedia dalam bentuk *input* dan tidak ada variabel *output* yang sesuai. Algoritma semacam itu memodelkan pola yang mendasari data untuk mempelajari lebih lanjut tentang karakteristiknya. Salah satu jenis utama dari algoritma *unsupervised* adalah pengelompokan. Dalam teknik ini, kelompok yang melekat dalam data ditemukan dan kemudian digunakan untuk memprediksi *output* untuk *input* yang tidak terlihat. Contoh dari teknik ini adalah untuk memprediksi perilaku pembelian pada

pelanggan. Gambar 2.2 merupakan visualisasi dari algoritma *unsupervised learning*.



Gambar 2.2: Gambaran *Unsupervised Learning*[2]

3. *Reinforcement learning* diterapkan ketika tugas yang ada adalah membuat urutan keputusan menuju *reward* akhir. Selama proses *learning*, *artificial agent* mendapat *reward* atau *penalties* atas tindakan yang dilakukannya. Tujuannya adalah untuk memaksimalkan total *reward* yang didapatkan. Gambar 2.3 merupakan visualisasi dari algoritma *reinforcement learning*.



Gambar 2.3: Gambaran *Reinforcement Learning*[3]

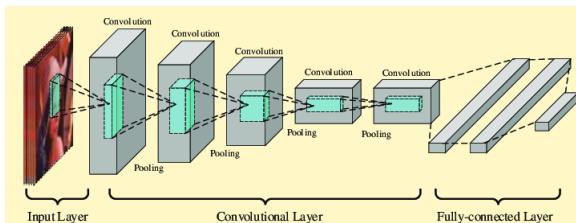
2.1.3 Deep Learning

Deep Learning adalah kelas *machine learning* yang berkinerja jauh lebih baik pada data tidak terstruktur[8]. Teknik *deep learning* mengungguli teknik *machine learning* saat ini. Ini memungkinkan model komputasi untuk mempelajari fitur secara progresif dari data di berbagai level. Popularitas *deep learning* diperkuat karena jumlah data yang tersedia meningkat serta kemajuan perangkat keras yang menyediakan komputer yang kuat.

Arsitektur *deep learning* berkinerja lebih baik daripada jaringan saraf tiruan sederhana, meskipun waktu *learning* dari struktur *deep learning* lebih tinggi dari jaringan saraf tiruan. Namun, waktu *learning* dapat dikurangi dengan menggunakan metode seperti *transfer learning* atau komputasi menggunakan GPU. Salah satu faktor yang menentukan keberhasilan jaringan saraf terletak pada desain arsitektur jaringan yang cermat.

2.1.4 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah jenis khusus dari *multilayer neural network* atau arsitektur *deep learning* yang terinspirasi oleh sistem visual makhluk hidup [9]. CNN sangat cocok untuk berbagai bidang visi komputer dan *natural language processing*. *Convolutional Neural Network* (CNN), juga disebut *ConvNet*, adalah jenis *Artificial Neural Network* (ANN), yang memiliki arsitektur *feed-forward* yang dalam dan memiliki kemampuan generalisasi yang luar biasa dibandingkan dengan jaringan lain dengan lapisan FC (*Fully Connected*), ia dapat mempelajari fitur objek yang sangat abstrak terutama data spasial dan dapat mengidentifikasi-nya dengan lebih efisien. Model CNN yang dalam terdiri dari satu set lapisan pemrosesan yang dapat mempelajari berbagai fitur data *input* (misalnya gambar) dengan beberapa tingkat abstraksi seperti yang ditampilkan pada Gambar 2.4. Lapisan inisiator mempelajari dan mengekstrak fitur tingkat tinggi (dengan abstraksi yang lebih rendah), dan lapisan yang lebih dalam mempelajari dan mengekstrak fitur tingkat rendah (dengan abstraksi yang lebih tinggi).



Gambar 2.4: Gambaran Konsep Arsitektur CNN [4]

Convolutional Neural Network memiliki beberapa keunggulan

dibanding dengan jaringan saraf tiruan lainnya dalam konteks visi komputer, antara lain :

1. Salah satu alasan utama untuk mempertimbangkan CNN dalam kasus tersebut adalah fitur pembagian bobot dari CNN, yang mengurangi jumlah parameter yang dapat dilatih dalam jaringan, yang membantu model untuk menghindari *overfitting* dan juga untuk meningkatkan generalisasi.
2. Pada CNN, lapisan klasifikasi dan lapisan ekstraksi fitur melakukan proses *learning* secara bersama-sama, yang membuat output model lebih terorganisir dan membuat output lebih bergantung pada fitur yang diekstraksi.
3. Implementasi pada jaringan dengan ukuran yang besar akan lebih sulit dilakukan dengan menggunakan jenis jaringan saraf lain daripada menggunakan *Convolutional Neural Network*

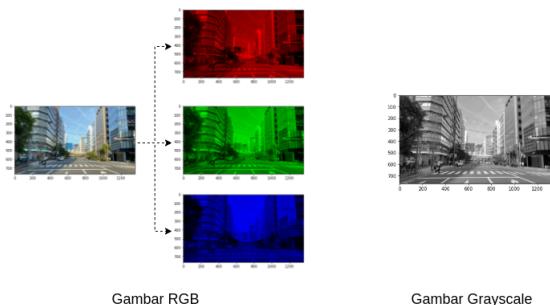
2.1.5 Convolutional Layer

Convolutional layer adalah komponen terpenting dari arsitektur CNN mana pun. Ini berisi satu set kernel convolutional (juga disebut filter), yang dililitkan dengan gambar input (metrik N-dimensi) untuk menghasilkan peta fitur keluaran. Kernel dapat digambarkan sebagai kisi nilai atau angka diskrit, di mana setiap nilai dikenal sebagai bobot kernel. Selama awal proses pelatihan model CNN, semua bobot kernel ditetapkan dengan angka acak (pendekatan yang berbeda juga tersedia untuk inisialisasi bobot). Kemudian, dengan setiap periode *learning*, bobot disetel dan kernel belajar mengekstrak fitur yang memberikan informasi mengenai data.

Pada *Convolutional layer* terdapat istilah kernel yang mempunyai peran penting dalam proses yang dinamakan *convolutional operation*. Kernel dapat digambarkan sebagai kisi nilai atau angka diskrit, di mana setiap nilai dikenal sebagai bobot kernel ini. Selama awal proses pelatihan model CNN, semua bobot kernel ditetapkan dengan angka acak (pendekatan yang berbeda juga tersedia di sana untuk menginisialisasi bobot). Kemudian, dengan setiap periode

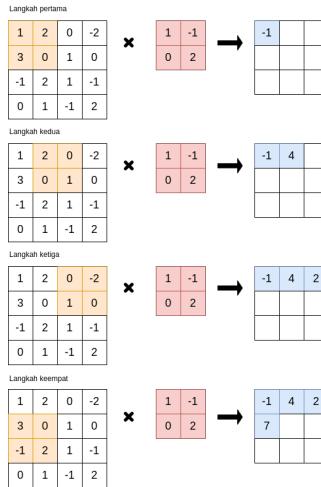
pelatihan, bobot disetel dan kernel belajar mengekstrak fitur yang berarti.

Sebelum kita mempelajari lebih jauh, mari kita pahami dulu format *input* CNN. Berbeda dengan *neural network* klasik lainnya dimana *input*-nya dalam format vektor, di CNN *input*-nya adalah gambar multi *channel*. Misalnya untuk gambar RGB seperti pada Gambar 2.5 mempunyai 3 *channel* dan untuk gambar *grayscale* hanya mempunyai satu *channel*.



Gambar 2.5: Contoh Gambar RGB dan Grayscale

Sekarang, dalam operasi konvolusi, sebagai contoh diambil kernel 2×2 lalu diimplementasikan ke semua gambar 4×4 secara horizontal maupun vertikal dan sepanjang proses berlangsung dilakukan *dot product* antara kernel dan gambar *input* dengan mengalikan nilai yang sesuai dari keduanya dan dijumlahkan semua nilai untuk menghasilkan satu nilai skalar di *feature map output*. Proses ini berlanjut hingga kernel tidak dapat lagi digeser lebih jauh. Untuk memahaminya secara lebih jelas, mari kita lakukan beberapa perhitungan awal yang dilakukan pada setiap langkah secara grafis seperti yang ditunjukkan pada Gambar 2.6, di mana setiap nilai kernel 2×2 (ditunjukkan dengan warna merah) berada dikalikan dengan wilayah berukuran sama (ditunjukkan dengan warna jingga) dalam gambar input 4×4 dan nilai yang dihasilkan dijumlahkan untuk mendapatkan entri yang sesuai (ditunjukkan dengan warna biru) di *feature map output* pada setiap langkah konvolusi.



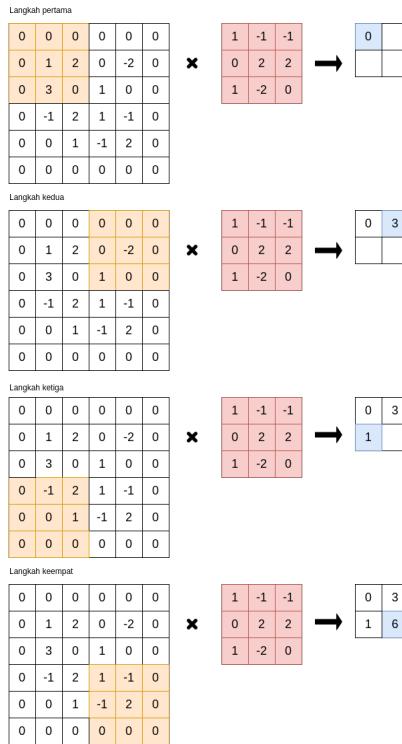
Gambar 2.6: Gambaran Operasi Konvolusi pada *Convolution Layer*

Dalam contoh di atas, diterapkan operasi konvolusi tanpa padding ke gambar masukan dan dengan *stride* (yaitu ukuran langkah yang diambil sepanjang posisi horizontal atau vertikal) sama dengan 1 ke kernel. Tapi bisa digunakan nilai *stride* lain (bukan 1) dalam operasi konvolusi. Hal yang terlihat adalah jika kita meningkatkan langkah operasi konvolusi, itu menghasilkan *feature map* berdimensi lebih rendah. *Padding* penting untuk memberikan informasi ukuran batas dari input gambar jika tidak, tanpa menggunakan *padding* apa pun, fitur sisi tepi akan terhapus terlalu cepat. *Padding* juga digunakan untuk meningkatkan ukuran gambar *input*, akibatnya ukuran *feature map output* juga meningkat. Gambar 2.7 memberikan contoh dengan menunjukkan operasi konvolusi dengan *Zero-padding* dan *3 stride*. Rumus untuk mencari ukuran *feature map* keluaran setelah operasi konvolusi adalah sebagai berikut:

$$h' = \left[\frac{h - f + p}{s} + 1 \right] \quad (2.1)$$

$$w' = \left\lceil \frac{w - f + p}{s} + 1 \right\rceil \quad (2.2)$$

Dimana h' menunjukkan tinggi dari *feature map output*, w' menunjukkan lebar dari *feature map output*, h menunjukkan tinggi dari gambar *input*, w menunjukkan lebar dari gambar *input*, f adalah ukuran filter, p menunjukkan *padding* dari operasi konvolusi dan s menunjukkan *stride* operasi konvolusi.



Gambar 2.7: Gambaran Operasi Konvolusi pada *Convolution Layer* menggunakan *Zero Padding*

Keuntungan utama dari lapisan konvolusi adalah:

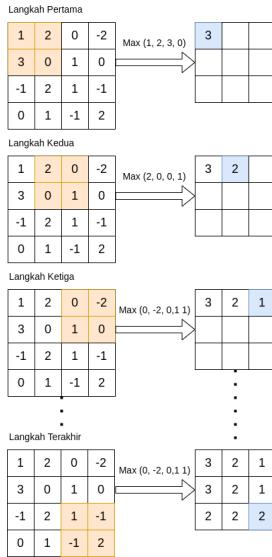
1. ***Sparse Connectivity***: Dalam jaringan saraf yang terhubung penuh, setiap *neuron* dari satu lapisan terhubung dengan setiap *neuron* dari lapisan berikutnya tetapi di CNN sejumlah kecil bobot terdapat di antara dua lapisan. Akibatnya, jumlah koneksi atau bobot yang kita butuhkan kecil, dan jumlah memori untuk menyimpan bobot itu juga kecil, sehingga hemat dalam penggunaan memori. Selain itu, operasi $\text{dot}(\cdot)$ secara komputasi lebih ringan daripada perkalian matriks.
2. ***Weight Sharing***: Di CNN, tidak ada bobot khusus yang ada di antara dua neuron dari lapisan yang berdekatan melainkan semua bobot bekerja dengan setiap piksel dari matriks *input*. Selain mempelajari bobot baru untuk setiap neuron, kita dapat mempelajari satu set bobot untuk semua input dan hal ini secara drastis dapat mengurangi waktu pelatihan.

2.1.6 *Pooling Layer*

Pooling layer digunakan untuk membuat sub-sampel *feature map* (dihasilkan setelah operasi konvolusi), yaitu mengambil *feature map* berukuran lebih besar dan mengecilkannya menjadi *feature map* berukuran lebih kecil. Saat menyusutkan *feature map*, *pooling layer* tetap mempertahankan fitur (atau informasi) yang paling dominan di setiap langkah *pools*. Operasi *pooling* dilakukan dengan menentukan ukuran *pooled region* dan langkah operasi, mirip dengan operasi konvolusi. Ada berbagai jenis teknik *pooling* yang digunakan dalam berbagai *pooling layer* seperti *max pooling*, *min pooling*, *average pooling*, *gated pooling*, *tree pooling*, dan lain-lain. *Max Pooling* adalah teknik *pooling* yang paling populer dan banyak digunakan.

Kelemahan utama dari *pooling layer* adalah terkadang menurunkan kinerja CNN secara keseluruhan. Alasan di balik ini adalah bahwa lapisan penyatuhan membantu CNN untuk menemukan apakah fitur tertentu ada dalam gambar masukan yang diberikan atau tidak tanpa memperhatikan posisi yang tepat dari fitur tersebut.

Gambar 2.8 mengilustrasikan contoh yang menunjukkan beberapa langkah awal serta keluaran akhir dari operasi *max-pooling*, di mana ukuran area *pooling* adalah 2×2 (ditunjukkan dalam warna



Gambar 2.8: Gambaran Proses *Max Pooling*

jingga, dalam *feature map input*) dengan *stride* sama dengan 1 dan yang sesuai dihitung nilai di *feature map output*. Rumus untuk menemukan ukuran *feature map output* setelah operasi *pooling* seperti di bawah ini:

$$h' = \left[\frac{h - f}{s} \right] \quad (2.3)$$

$$w' = \left[\frac{w - f}{s} \right] \quad (2.4)$$

Dimana h' menunjukkan tinggi dari *feature map output*, w' menunjukkan lebar dari *feature map output*, h menunjukkan tinggi dari gambar *input*, w menunjukkan lebar dari gambar *input*, f adalah ukuran area *pooling*, s menunjukkan *stride* operasi *pooling*.

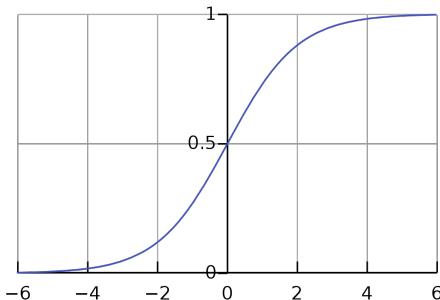
2.1.7 Fungsi Aktivasi (*Non-Linearity*)

Tugas utama dari setiap fungsi aktivasi dalam setiap model berbasis jaringan saraf adalah untuk memetakan *input output*, di mana nilai *input* diperoleh dengan menghitung jumlah terbobot *input neuron* dan selanjutnya menambahkan bias (jika ada bias). Dengan kata lain, fungsi aktivasi memutuskan apakah neuron akan aktif atau tidak untuk *input* yang diberikan dengan menghasilkan *output* yang sesuai. Terdapat beberapa fungsi aktivasi yang digunakan dalam CNN, antara lain :

1. Sigmoid

Fungsi aktivasi sigmoid mengambil bilangan real sebagai inputnya dan mengikat outputnya dalam kisaran [0,1]. Kurva fungsi sigmoid berbentuk 'S' seperti pada Gambar 2.9. Representasi tematik dari sigmoid adalah:

$$f(x)_{sigm} = \frac{1}{1 + e^{-x}} \quad (2.5)$$



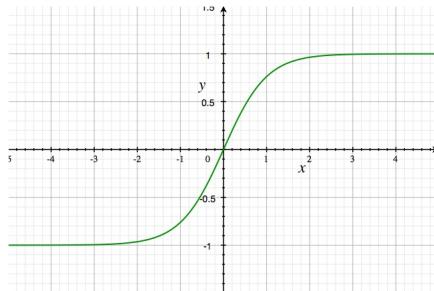
Gambar 2.9: Grafik Fungsi Aktivasi Sigmoid [5]

2. Tanh

Fungsi aktivasi Tanh digunakan untuk mengikat nilai input (bilangan real) dalam kisaran [-1,1] seperti tertampil pada

Gambar 2.10. Representasi matematis dari Tanh adalah:

$$f(x)_{\tanh} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.6)$$



Gambar 2.10: Grafik Fungsi Aktivasi Tanh [5]

3. ReLU

Rectifier Linear Unit (ReLU) adalah fungsi aktivasi yang paling umum digunakan dalam Convolutional Neural Networks. Ini digunakan untuk mengubah semua nilai input menjadi bilangan positif. Keuntungan dari ReLU adalah membutuhkan beban komputasi yang sangat minimal dibandingkan dengan yang lain. Representasi matematis dari ReLU adalah:

$$f(x)_{ReLU} = \max(0, x) \quad (2.7)$$

Namun terkadang ada beberapa masalah besar dalam menggunakan fungsi aktivasi ReLU. Misalnya, jika terdapat gradient yang besar pada saat pencarian *error* dengan menggunakan algoritma *back propagation*. Gradien yang besar ini apabila dilewatkhan melalui fungsi ReLU, dapat menyebabkan bobot akan diperbarui sedemikian rupa sehingga neuron tidak pernah diaktifkan lagi. Masalah ini dikenal sebagai masalah "*Dying ReLU*". Untuk mengatasi masalah ini ada beberapa varian ReLU yang tersedia, diantaranya adalah Leaky ReLU dan Noisy ReLU. Tidak seperti ReLU, fungsi aktivasi Leaky

ReLU tidak mengabaikan input negatif sepenuhnya, melainkan menurunkan skala input negatif tersebut. Leaky ReLU digunakan untuk menyelesaikan masalah *Dying ReLU*. Representasi matematis dari Leaky ReLU adalah

$$f(x)_{\text{LeakyReLU}} = \begin{cases} x, & \text{if } x > 0 \\ mx, & \text{if } x \leq 0 \end{cases} \quad (2.8)$$

Di mana m adalah konstanta, yang disebut faktor kebocoran dan umumnya disetel ke nilai kecil (seperti 0,001). Sedangkan Noisy ReLU digunakan distribusi Gaussian untuk membuat ReLU *noisy*. Representasi matematis dari Noise ReLU adalah

$$f(x)_{\text{NoisyReLU}} = \max(x + Y), \text{ with } Y \sim N(0, \sigma(x)) \quad (2.9)$$

2.1.8 Fully Connected Layer

Biasanya bagian terakhir (atau lapisan) dari setiap arsitektur CNN (digunakan untuk klasifikasi) terdiri dari *fully-connected layers*, di mana setiap neuron di dalam lapisan terhubung dengan setiap neuron dari lapisan sebelumnya. Lapisan terakhir dari lapisan *Fully-Connected* digunakan sebagai lapisan keluaran (*classifier*) dari arsitektur CNN. Lapisan *Fully-Connected* adalah jenis jaringan saraf tiruan *feed-forward* (ANN) dan mengikuti prinsip tradisional *Multi-Layer Perceptron neural network* (MLP). Layer FC mengambil *input* dari *convolutional* atau *pooling layer* akhir, yang berupa sekumpulan metrik (*feature map*) dan matrik tersebut diratakan (*flatten*) untuk membuat vektor dan vektor ini kemudian dimasukkan ke dalam layer FC untuk menghasilkan hasil akhir *output* CNN.

2.1.9 Loss Function

Pada *output layer*, dilakukan perhitungan kesalahan prediksi yang dihasilkan oleh model CNN atas sampel *learning* menggunakan beberapa *Loss Function*. *Error* prediksi ini memberitahu jaringan bagaimana prediksinya dari *output* aktual, dan kemudian *error* ini akan dioptimalkan selama proses *learning* model CNN. *Loss function* menggunakan dua parameter untuk menghitung *error*, para-

meter pertama adalah *output* estimasi dari model CNN (juga disebut prediksi) dan yang kedua adalah *output* aktual (juga dikenal sebagai label). Berikut adalah beberapa contoh *loss function* yang banyak digunakan dalam algoritma *deep learning*:

1. *Cross-Entropy Loss Function*

Cross-entropy loss, juga disebut fungsi *log loss* banyak digunakan untuk mengukur kinerja model CNN, yang outputnya adalah probabilitas $p \in \{0, 1\}$. *Loss Function* ini banyak digunakan sebagai alternatif dari *squared error loss function* dalam masalah klasifikasi multi-kelas. *Cross-entropy loss* menggunakan aktivasi softmax di lapisan *output* untuk menghasilkan *output* dalam distribusi probabilitas, yaitu $p, y \in R^N$, di mana p adalah probabilitas untuk setiap kategori *output* dan y menunjukkan *output* yang diinginkan dan probabilitas setiap kelas *output* dapat diperoleh dengan :

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^N e^{a_k}} \quad (2.10)$$

Di mana N adalah jumlah neuron di lapisan *output* dan e^{a_i} menunjukkan setiap *output* yang tidak dinormalisasi dari lapisan sebelumnya dalam jaringan. Lalu, *cross-entropy loss* dapat didefinisikan sebagai:

$$H(p, y) = - \sum_i y_i \log(p_i) \quad (2.11)$$

dimana $i \in [1, N]$

2.2 Penelitian Terkait

2.2.1 Real-Time Pedestrian Detection With Deep Network Cascades

Penelitian ini dilakukan oleh Anelia Angelova dan kawan-kawan pada tahun 2015 yang menyajikan pendekatan real-time baru untuk deteksi objek yang mengeksplorasi efisiensi *cascade classifiers* dengan akurasi *deep neural network* [10]. *Deep network* telah terbukti

unggul dalam tugas klasifikasi, dan kemampuannya untuk beroperasi pada *raw pixel input* tanpa perlu merancang fitur khusus. Namun, *deep network* terkenal lambat pada waktu inferensi. Dalam *paper* tersebut, Anelia Angelova dan kawan-kawan mengusulkan pendekatan *cascades deep network* dan *fast features*, yang sangat cepat dan sangat akurat. Mereka menerapkannya pada permasalahan pada deteksi pejalan kaki. Algoritma mereka berjalan secara real-time pada 15 frame per detik. Pendekatan yang dihasilkan mencapai tingkat kesalahan rata-rata 26,2% pada *benchmark* deteksi Caltech Pedestrian.

[Halaman ini sengaja dikosongkan]

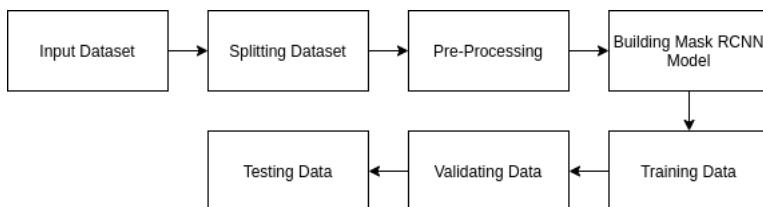
BAB III

DESAIN DAN IMPLEMENTASI

Penelitian ini dilaksanakan sesuai dengan sistem berikut dengan implementasinya. Desain sistem merupakan konsep dari pembuatan dan perancangan infrastruktur dan kemudian diwujudkan dalam bentuk blok-blok alur yang harus dikerjakan. Pada bagian implementasi merupakan pelaksanaan teknis untuk setiap blok pada desain sistem.

3.1 Deskripsi Sistem

Sistem pada tugas akhir ini merupakan implementasi dari salah satu disiplin ilmu *Deep Learning* dan pengolahan citra yang berfungsi untuk mendeteksi adanya pejalan kaki yang berada di pinggir jalan, trotoar dan jalur penyebrangan. Selain pejalan kaki, deteksi juga dilakukan pada jalur penyebrangan atau *zebracross* dengan tujuan untuk memberi informasi bahwa disekitar area tersebut terdapat banyak aktivitas pejalan kaki yang menyebrang jalan. Blok diagram metodologi sistem yang digunakan pada penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1: Blok Diagram Metodologi

3.2 Pengumpulan *Dataset* Gambar

Pada tugas akhir ini, *dataset* yang digunakan didapatkan dengan beberapa cara, antara lain:

1. *Caltech Pedestrian Database*, merupakan kumpulan gambar yang diambil dari sudut pandang pengendara mobil di California Amerika Serikat dengan ukuran 640 x 480 pixel. Terdapat sekitar 250.000 gambar dengan 350.000 *bounding boxes* dan sekitar 2.300 pejalan kaki dengan kriteria unik diberi tanda. Namun, pada *dataset* ini hanya pejalan kaki saja yang diberi label, sehingga perlu dilakukan proses pelabelan ulang sesuai kelas yang diinginkan. Tidak semua gambar pada *dataset* ini diambil untuk digunakan, gambar yang mempunyai objek berupa pejalan kaki dan *zebracross* saja yang akan digunakan. Gambar 3.2 merupakan contoh dari gambar yang terdapat pada *Caltech Pedestrian Database*.



Gambar 3.2: Contoh Gambar dari Caltech Pedestrian Database

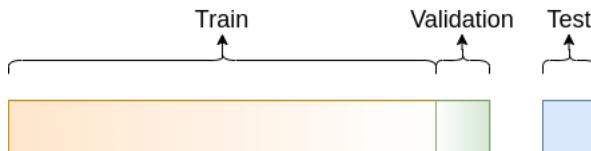
2. Tangkapan layar dari beberapa video *online Youtube*. Pada cara ini, penulis mencari video yang berada pada salah satu *website video streaming* yaitu Youtube dengan persyaratan video diambil dari sudut pandang pengendara mobil yang berkendara pada jalan raya dengan ukuran gambar 1360x768 px. Pada *frame-frame* tertentu dilakukan *screenshot* dan disimpan untuk selanjutnya dilakukan proses pemberian label pada objek-objek yang diinginkan seperti pada Gambar 3.3.
3. Pengambilan gambar secara mandiri menggunakan kamera *smartphone* yang diambil dari sudut pandang pengendara motor dengan ukuran gambar yang diambil sebesar 1280x720 px. Pengambilan gambar dilakukan di jalan-jalan Surabaya. Setelah dilakukan pengambilan gambar, proses selanjutnya adalah pemberian label pada objek-objek yang ingin dideteksi.



Gambar 3.3: Contoh Pembuatan *dataset* dari *Screenshot Youtube*

3.3 Pemisahan Data

Dalam *machine learning* pemisahan data ke beberapa *subset* merupakan suatu hal yang sangat penting. Hal ini dikarenakan setiap *subset* memiliki fungsi masing-masing. Gambar 3.4 merupakan rasio pembagian data ke masing-masing subset.



Gambar 3.4: Visualisasi Pembagian Data

1. *Training Sets*

Training Sets merupakan sampel data yang digunakan untuk melatih model yang sudah kita buat, dalam bidang *Neural Network* bisa disebut juga bobot dan bias. Model yang sudah kita buat mempelajari pola masukan dan keluaran dari data ini.

2. *Validation Sets*

Validation Sets merupakan sampel data yang digunakan untuk mengevaluasi model yang sudah dilatih menggunakan *train-*

ining sets. Selain itu, data ini digunakan untuk memperbarui dan menyempurnakan hyperparameter dari model ke tingkat yang lebih tinggi.

3. *Test Sets*

Test Sets merupakan sampel data yang digunakan untuk meng-evaluasi model akhir setelah melalui proses *training* dan *validation*. Apabila pengujian model pada data ini sudah sesuai dengan yang diinginkan, maka proses *learning* sudah selesai. Namun apabila pengujian tidak sesuai dengan yang diharapkan maka diperlukan pengaturan ulang mulai dari proses *training*.

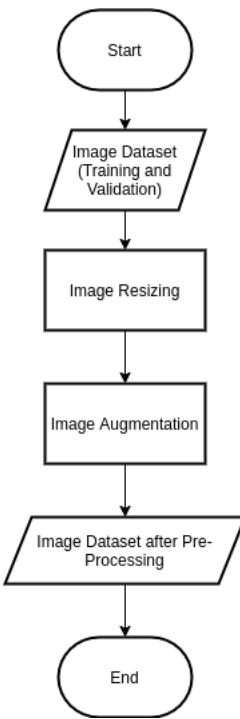
3.4 *Pre-Processing*

Pada tahap ini, gambar-gambar dari *dataset* akan mengalami proses penyesuaian sebelum masuk ke proses *data training*. Setiap gambar yang akan dijadikan bahan pembelajaran model harus memiliki dimensi dan kedalaman yang sama. Tujuan dari *pre processing* adalah perbaikan data gambar dengan menekan distorsi yang tidak diinginkan atau meningkatkan beberapa fitur gambar yang relevan untuk pemrosesan lebih lanjut. Gambar 3.5 merupakan tahapan dari *pre-processing* gambar *dataset* yang dilakukan.

Berikut merupakan penjelasan mengenai tahapan *pre-processing* yang dilakukan pada tugas akhir kali ini:

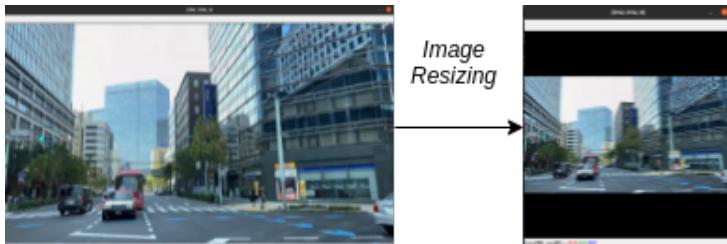
1. *Image resizing*

Langkah awal dari proses *pre-processing* adalah memastikan semua gambar dalam *dataset* kita memiliki ukuran yang sama. Selain itu, sama seperti sebagian besar model dari *neural network* lainnya, metode yang dilakukan penulis juga meng-asumsikan gambar *input* berbentuk persegi. Jadi diperlukan pemeriksaan gambar di awal, apakah gambar sudah berben-tuk persegi atau belum. Berbeda dari metode *image resizing* pada model *neural network* lainnya yang menggunakan teknik *cropping* untuk membuat aspek rasio gambar input menjadi persegi, penulis menggunakan metode yang sudah terdapat pada *Mask R-CNN*.



Gambar 3.5: Diagram Alir *Pre Processing*

Ukuran gambar yang penulis pilih pada tugas akhir kali ini adalah 512x512 pixel. Pemilihan ukuran gambar ini dilakukan untuk mengurangi beban dan waktu saat *training data*. Apabila terdapat gambar pada *dataset* dengan ukuran baik panjang maupun lebar lebih dari 512 pixel. maka gambar akan di *down scaling* sampai ukuran 512 pixel. Sebaliknya, apabila ada gambar pada *dataset* dengan ukuran lebih kecil dari 512 pixel maka akan dilakukan *up scaling* sampai gambar berukuran 512 pixel. Aspek rasio gambar yang sudah melalui proses *scaling* tetap dipertahankan, namun diperlukan penambahan *zero padding* untuk membuat gambar *input* menjadi persegi seperti yang diinginkan.



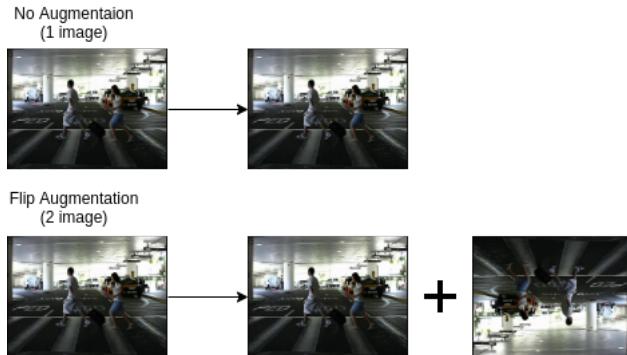
Gambar 3.6: Contoh *Image Resizing*

Gambar 3.6 merupakan salah satu contoh *image resizing* yang dilakukan. Gambar *input* (gambar sebelah kiri) mempunyai ukuran 768x1360 dengan kedalaman 3 atau mempunyai format warna RGB. Setelah mengalami *image resizing* (gambar sebelah kanan) ukuran gambar menjadi 290x512. Namun untuk membuat gambar memiliki aspek rasio 1:1 (berbentuk persegi) maka diperlukan penambahan *zero padding* pada bagian atas gambar sebesar 111 pixel dan pada bagian bawah gambar sebesar 111 pixel. Dengan penambahan *padding* seperti itu membuat gambar *input* berbentuk persegi namun tidak mengurangi informasi gambar.

2. *Image Augmentation*

Langkah selanjutnya pada *pre-processing* adalah *image augmentation*. Proses augmentasi yang dilakukan pada tugas akhir ini adalah rotasi dan transformasi. Tujuan dari penggunaan *image augmentation* adalah untuk mengekspos *neural network* ke berbagai variasi, agar dapat mengenali fitur yang akan dilakukan pada proses *training*. Hal tersebut akan sangat membantu *neural network* untuk mengenali variasi yang tidak terdapat pada dataset. Seperti yang terdapat pada Gambar 3.7, tanpa ada augmentasi maka *neural network* hanya mengenali satu kondisi saja. Jika memakai augmentasi gambar seperti transformasi, maka setidaknya *neural network* akan dapat mengenali 2 kondisi. Semakin banyak augmentasi yang digunakan semakin banyak pula kondisi yang bisa dikenali oleh *neural network*. Namun semakin banyak kondisi

yang dikenali, semakin lama dan berat proses *training data* yang dilakukan.

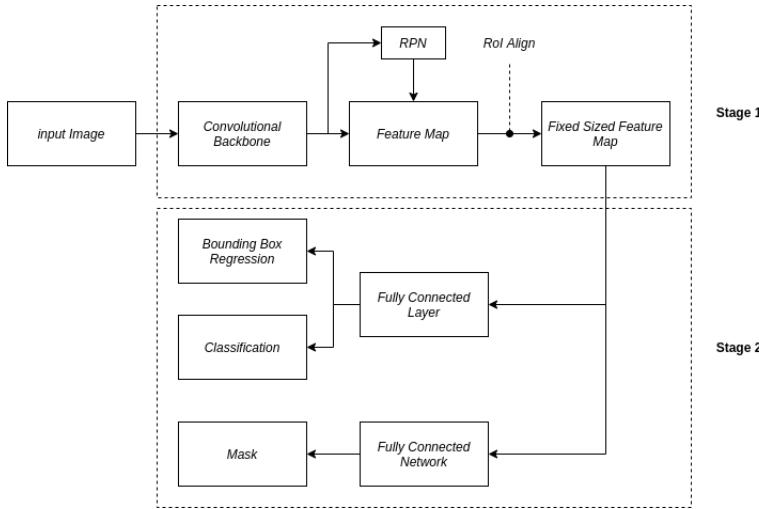


Gambar 3.7: Contoh *Image Augmentation*

3.5 Membangun Model Mask R-CNN

Mask R-CNN merupakan salah satu metode *deep learning* yang dikembangkan dari Faster R-CNN dengan menambahkan satu cabang di tahap akhir untuk menghasilkan *mask* dari objek yang didekksi. Pengembangan tersebut dilakukan untuk memecahkan masalah *instance segmentation* yang terjadi dalam *machine learning* dan pengolahan citra. Dengan kata lain, *mask r-cnn*, dapat memisahkan objek yang berbeda walaupun dalam satu kelas yang sama pada gambar atau video. Selain memberikan hasil berupa *bounding box* dan klasifikasi objek seperti kebanyakan algoritma *object detection* lainnya, *mask r-cnn* juga memberikan *mask* dimana hal ini sangat bermanfaat pada segmentasi objek.

Seperti yang ditampilkan pada Gambar 3.8, pada proses *training* menggunakan *mask r-cnn* dibagi menjadi 2 tahapan. Tahap pertama adalah tahap untuk menghasilkan proposal tentang area di mana mungkin ada objek berdasarkan gambar *input*. Lalu tahap kedua adalah tahap untuk memprediksi kelas objek, memperbaiki *bounding box* dan menghasilkan *mask* di tingkat piksel objek berdasarkan proposal tahap pertama. Kedua tahap terhubung ke struktur



Gambar 3.8: Blok Diagram Alur Mask R-CNN

backbone.

Backbone adalah *deep neural network* yang memiliki struktur seperti FPN (*Feature Pyramid Network*). *Backbone* terdiri dari *bottom-up pathway*, *up-bottom pathway* dan *lateral connection*. *Bottom-up pathway* dapat berupa berbagai jenis *Convolutional Network*, biasanya berupa *ResNet* atau *VGG*, yang mengekstrak fitur dari *raw images*. *Up-bottom pathway* menghasilkan *Feature Map Pyramid* yang ukurannya mirip dengan *bottom-up pathway*. *Lateral connection* adalah operasi konvolusi dan penjumlahan antara dua *pathway* dengan tingkat yang sesuai. FPN mempunyai kinerja yang lebih baik dari ConvNet tunggal lainnya terutama karena FPN dapat mempertahankan fitur semantik yang sangat baik pada berbagai skala resolusi.

3.6 *Training Data*

Proses *training data* dilakukan setelah pembuatan model telah selesai dan *dataset* sudah melalui proses *pre-processing*. Pada

saat pertama kali menjalankan proses *training*, bobot awal diam-bil dari *pre-trained weight* yang sudah tersedia pada *mask r-cnn*. Hal ini bisa dilakukan dengan menerapkan metode *transfer learning*. *Transfer learning* sendiri adalah teknik yang sangat efisien untuk melakukan proses *training* atau *retrain* pada *neural network*. Penggunaan *transfer learning* mempunyai keuntungan diantara lain proses *training* pada data baru memakan waktu yang lebih cepat daripada memulai dari awal serta masalah dapat dipecahkan dengan menggunakan training data yang lebih sedikit daripada membangun model dari awal.

Ada beberapa hal yang perlu diperhatikan dalam melakukan pengaturan saat akan menjalankan proses *training* antara lain :

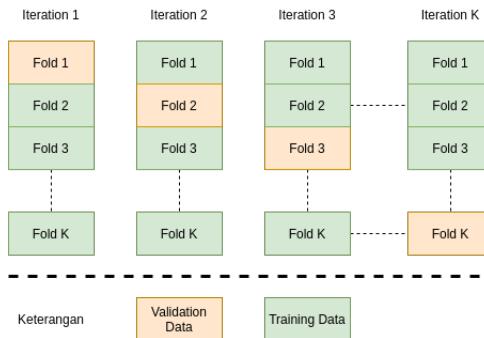
1. *Iteration* adalah banyaknya proses yang dilakukan untuk melakukan *forward* dan *backward pass*. *forward pass* adalah proses dimana *output value* dari *neural network* didapatkan setelah *input value* dari *input-neuron* telah selesai diproses. Sedangkan *backward pass* adalah proses mengkalkulasikan bobot dari *neural network* mulai dari *output neuron* ke *input neuron* untuk mendapatkan *loss* dari setiap *neuron*. Pada *mask r-cnn iteration* bisa disebut juga dengan *step_per_epoch* sesuai dengan yang tercantum pada *file pengaturan mask r-cnn*.
2. *Epoch*, ketika seluruh dataset sudah melalui proses *training* pada *neural network* sampai dikembalikan ke awal untuk sekali putaran. Sebagai contoh apabila kita menggunakan *iteration* sebanyak 10 kali, maka satu *epoch* sebanyak 10 *iteration* dan kelipatannya. Pada tugas akhir kali ini penulis menggunakan *epoch* sebanyak 100.

3.7 *Validating Data*

Setelah proses *training* dilakukan, perlu dilakukan apakah mo-del yang dibuat sudah memiliki tingkat akurasi sesuai yang kita inginkan dengan menggunakan teknik validasi. Pada proses inilah, *dataset* yang telah dipisahkan pada proses sebelumnya akan berper-an. Evaluasi memungkinkan pengujian model terhadap data yang belum pernah dilihat dan digunakan untuk pelatihan dan dimak-

sudkan untuk mewakili bagaimana model dapat menyelesaikan permasalahan tersebut. Tahapan pada validasi akan membantu untuk menemukan parameter terbaik untuk model prediktif dan mencegah dari *overfitting*.

Pada tugas akhir kali ini, digunakan salah satu jenis teknik validasi menggunakan *Cross Validation*. Pada *Cross Validation* data akan dibagi menjadi K lipatan, dimana setiap lipatan akan diambil satu data sebagai *validation data* dan sisanya akan digunakan untuk *training data*. Pemilihan *validation data* dilakukan secara menyilang, dengan ketentuan apabila *training* terjadi pada iterasi ke k maka data yang dipilih untuk validasi adalah data k juga. Gambar 3.9 merupakan visualisasi dari K *Fold Cross Validation*.



Gambar 3.9: Visualisasi K *Fold Cross Validation*

3.8 Testing Data

Testing data merupakan tahap akhir dalam algoritma *machine learning* secara umum. Pada tahap ini model akan diuji untuk didapatkan keakuratan untuk mendeteksi objek. Data uji yang digunakan pada tugas akhir kali ini adalah data yang berbentuk video. Jadi diperlukan *pre-processing* yang berbeda dengan *training data* dan *validation data*. Data video akan dipecah atau dipotong-potong menjadi format gambar dengan ketentuan 30 *frame per second*. Ketika *data test* sudah dikonversi menjadi bentuk gambar, maka proses deteksi bisa dilakukan. Gambar hasil deteksi berupa

gambar asli yang sudah ditambah dengan *bounding box*, *classification*, *mask*. Lalu gambar-gambar tersebut disatukan lagi menjadi format video dengan ketentuan sama seperti saat pemotongan menjadi format gambar, yaitu 30 *frame per second*.

[Halaman ini sengaja dikosongkan]

BAB IV

PENGUJIAN DAN ANALISIS

Pada bab ini dipaparkan hasil pengujian serta analisa dari desain sistem dan implementasi. Pengujian dilakukan guna mengetahui tingkat kesalahan dan menarik kesimpulan dari sistem yang telah dibuat.

Pada proses pengujian digunakan salah satu layanan *Google* yaitu *Google Colaboratory* dengan spesifikasi *hardware* seperti pada Tabel 4.1. Sedangkan untuk spesifikasi *hardware* komputer penulis dapat dilihat pada Tabel 4.2.

Prosesor	Intel Xeon Processor @ 2.3 GHz
Graphic Card	Tesla K80 12 GB GDDR5 VRAM
RAM	16 GB

Tabel 4.1: Spesifikasi *hardware* *Google Colaboratory*

Prosesor	Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz
Graphic Card	Nvidia GeForce GTX 1650 4 GB GDDR6
RAM	8 GB

Tabel 4.2: Spesifikasi *hardware* Komputer yang Digunakan

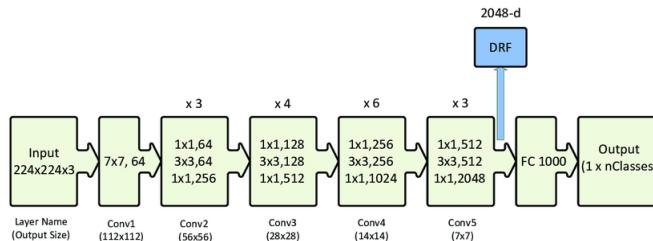
Pengujian dilakukan dengan membagi menjadi beberapa bagian yaitu sebagai berikut

1. Pengujian jenis *backbone*
2. Pengujian jenis teknik validasi

4.1 Pengujian Jenis *Backbone*

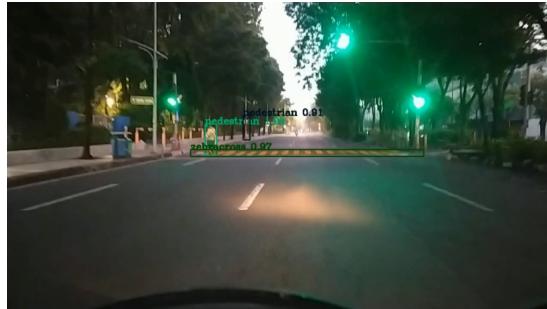
4.1.1 Resnet 50

Resnet-50 merupakan salah satu *Image Classification* dari CNN dengan jumlah layer sebanyak 50. Gambar 4.1 adalah arsitektur dari Resnet50.



Gambar 4.1: Gambaran Arsitekture ResNet50

Gambar 4.2 merupakan hasil deteksi yang dilakukan menggunakan *backbone* resnet-50.

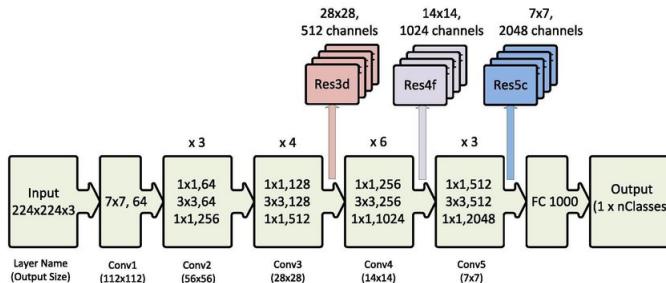


Gambar 4.2: Gambaran Hasil Deteksi ResNet50

4.1.2 Resnet 101

Resnet-101 merupakan salah satu *Image Classification* dari CNN dengan jumlah layer sebanyak 101. Gambar 4.3 adalah arsitektur dari Resnet-101.

Gambar 4.4 merupakan hasil deteksi yang dilakukan menggunakan



Gambar 4.3: Gambaran Arsitekture ResNet101

nakan *backbone* resnet-101.



Gambar 4.4: Gambaran Hasil Deteksi ResNet101

4.1.3 MobileNet V1

4.1.4 MobileNet V2

4.2 Pengujian Jenis Teknik Validasi

4.2.1 Pengujian dengan Pemisahan Data Validasi

4.2.2 Pengujian dengan *K Fold Cross Validation*

[Halaman ini sengaja dikosongkan]

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, penulis dapat menyimpulkan beberapa hal sebagai berikut:

1. Dalam penelitian ini telah diimplementasikan proses pendeteksian pejalan kaki dan zebracross dengan menggunakan Mask R-CNN
2. Backbone Resne101 memiliki hasil akurasi yang lebih baik dibanding dengan backbone lainnya serta penggunaan cross validation membuat loss menjadi lebih kecil

5.2 Saran

Untuk pengembangan lebih lanjut pada penelitian mendatang, maka penulis memiliki saran sebagai berikut:

1. Menambah jumlah sample data pada kelas pengendara motor yang dinilai masih sedikit untuk dataset yang dipilih.
2. Pembuatan dataset pejalan kaki di Indonesia karena perilaku pejalan kaki antar negara memiliki perbedaan yang cukup signifikan.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

- [1] Supervised vs. unsupervised learning, 2018. URL <https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d>. (Dikutip pada halaman xi, 6).
- [2] Clustering in machine learning, 2020. URL <https://www.geeksforgeeks.org/clustering-in-machine-learning>. (Dikutip pada halaman xi, 7).
- [3] Hongbo Gao, Guanya Shi, Guotao Xie, and Bo Cheng. Car-following method based on inverse reinforcement learning for autonomous vehicle decision-making. *International Journal of Advanced Robotic Systems*, 15:172988141881716, 11 2018. doi: 10.1177/1729881418817162. (Dikutip pada halaman xi, 7).
- [4] N. Ferracuti, Claudia Norscini, Emanuele Frontoni, P. Gabellini, Marina Paolanti, and Valerio Placidi. A business application of rtls technology in intelligent retail environment: Defining the shopper's preferred path and its segmentation. *Journal of Retailing and Consumer Services*, 47:184–194, 03 2019. doi: 10.1016/j.jretconser.2018.11.005. (Dikutip pada halaman xi, 8).
- [5] Sigmoid function, 2021. URL https://en.wikipedia.org/wiki/Sigmoid_function. (Dikutip pada halaman xi, 15, 16).
- [6] Artificial intelligence (ai), 2021. URL <https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp>. (Dikutip pada halaman 5).
- [7] Shagan Sah. Machine learning: A review of learning types. 07 2020. doi: 10.20944/preprints202007.0230.v1. (Dikutip pada halaman 5).

- [8] Amitha Mathew, Amudha Arul, and S. Sivakumari. *Deep Learning Techniques: An Overview*, pages 599–608. 01 2021. ISBN 978-981-15-3382-2. doi: 10.1007/978-981-15-3383-9_54. (Dikutip pada halaman 7).
- [9] Anirudha Ghosh, A. Sufian, Farhana Sultana, Amlan Chakrabarti, and Debasish De. *Fundamental Concepts of Convolutional Neural Network*, pages 519–567. 01 2020. ISBN 978-3-030-32643-2. doi: 10.1007/978-3-030-32644-9_36. (Dikutip pada halaman 8).
- [10] Anelia Angelova, Alex Krizhevsky, Vincent Vanhoucke, Abhijit Ogale, and Dave Ferguson. Real-time pedestrian detection with deep network cascades. 2015. (Dikutip pada halaman 18).

BIOGRAFI PENULIS



Elon Reeve Musk, lahir pada Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

[Halaman ini sengaja dikosongkan]