SISTEM PENDETEKSI SEPEDA MOTOR PELANGGAR MARKA JALAN MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORKS* (CNNs)

SKRIPSI

Diajukan untuk Memenuhi Bagian dari Syarat Memperoleh Gelar Sarjana Komputer Pada Departemen Pendidikan Ilmu Komputer Program Studi Ilmu Komputer

oleh

Ragil Nurhawanti

NIM 1504300

PROGRAM STUDI ILMU KOMPUTER

DEPARTEMEN PENDIDIKAN ILMU KOMPUTER

FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PENDIDIKAN INDONESIA

2018

PERNYATAAN

Saya menyatakan bahwa skripsi yang berjudul "Sistem Pendeteksi Sepeda Motor Pelanggar Marka Jalan Menggunakan Metode *Convolutional Neural Networks* (CNNs)" ini sepenuhnya karya sendiri. Tidak ada plagiat dari orang lain di dalamnya dan saya tidak melakukan penyalinan atau pengutipan dengan cara-cara yang tidak sesuai etika keilmuwan yang berlaku dalam masyarakat keilmuwan. Atas pernyataan ini, saya siap menanggung sanksi yang dijatuhkan kepada saya apabila ditemukan adanya pelanggaran terhadap etika keilmuwan di karya ini atau ada klaim dari pihak lain terhadap keaslian karya saya ini.

Bandung, Mei 2019

Pembuat pernyataan,

Ragil Nurhawanti

NIM 1504300

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena berkat rahmat dan hidayah-Nya, penelitian yang berjudul "Sistem Pendeteksi Sepeda Motor Pelanggar Marka Jalan Menggunakan Metode *Convolutional Neural Networks* (CNNs)" dapat diselesaikan. Skripsi ini disusun untuk memenuhi salah satu syarat dalam menempuh gelar Sarjana Komputer di Program Studi Ilmu Komputer, Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam, Universitas Pendidikan Indonesia. Penulis sadari bahwa skripsi yang disusun dengan segala usaha dari awal sampai selesai ini masih jauh dari kata sempurna. Oleh karena itu, penulis mengharapkan berbagai kritik dan saran dari para pecinta ilmu pengetahuan yang bersifat positif supaya skripsi ini dapat dibangun dengan lebih baik. Penulis juga berharap skripsi ini dapat memberikan manfaat, baik untuk penulis sendiri, umumnya bagi para pengembang teknologi dan pecinta ilmu pengetahuan.

Bandung, Mei 2019

Penulis,

Ragil Nurhawanti

UCAPAN TERIMA KASIH

Skripsi ini dapat diselesaikan tidak lepas dari dukungan, dorongan, serta bantuan dari berbagai pihak. Sehingga pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada beberapa pihak yang telah berperan serta dalam penyelesaian laporan ini, diantaraya adalah:

1. Orang tua yang selalu mendo'akan penulis sehingga penulisan skripsi dapat terselesaikan.
2. Prof. Munir, M.T. selaku Ketua Departemn Pendidikan Ilmu Komputer Universitas Pendidikan Indonesia.
3. Bapak Eddy Prasetyo Nugroho, M.T. selaku Ketua Program Studi Ilmu Komputer Universitas Pendidikan Indonesia.
4. Bapak Lala Septem Riza, M.T selaku pembimbing akademik yang telah memberikan arahan dalam perkuliahan.
5. Bapak Prof. Dr. H. Wawan Setiawan, M.Kom selaku dosen pembimbing I yang telah membimbing dalam penyelesaian skripsi dan penyusunan skripsi.
6. Bapak Yaya Wihardi, M.Kom selaku dosen pembimbing II yang telah membimbing dalam penyelesaian skripsi dan penyusunan skripsi.
7. Sahabat dan teman-teman serta semua pihak yang tidak dapat penulis sebutkan satu-persatu yang telah memberikan dukungan selama proses pengerjaan skripsi.

Penulis menyadari bahwa penyusunan skripsi ini masih terdapat banyak kekurangan dan keterbatasan yang perlu disempurnakan, oleh karena itu penulis sangat mengharapkan saran maupun kritik yang membangun agar tidak terjadi kesalahan yang sama di kemudian hari dan dapat meningkatkan kualitas ke tahap yang lebih baik. Akhir kata, semoga skripsi ini dapat bermanfaat dan memberikan wawasan yang luas kepada para pembaca.

Bandung, Mei 2019

Penulis

Abstrak

Banyaknya pelanggaran lalu lintas yang terjadi saat ini baik yang dilakukan oleh pengendara mobil atau sepeda motor merupakan cerminan dari kepribadian suatu bangsa. Pelanggaran terjadi karena kurangnya kedisiplinan pengendara dalam mematuhi rambu lalu lintas dan marka jalan yang ada. Pelanggaran lalu lintas menjadi masalah yang serius karena dari pelanggaran ini dapat berpotensi menjadi kecelakaan lalu lintas. Tidak mampunya petugas untuk mengawasi pengendara selama 24 jam membuat pelanggaran terus saja terjadi. Oleh karena itu untuk membantu mengatasi masalah tersebut dibuatlah sistem pendeteksi sepeda motor pelanggar marka jalan menggunakan metode *Convolutional Neural Networks* (CNNs). Pelanggaran yang dideteksi merupakan sepeda motor yang berhenti melewati *stop line* pada saat lampu merah menyala. Data yang digunakan merupakan video dari rekaman CCTV milik Dinas Perhubungan Kota Bandung yang diambil selama 10 hari agar video yang diperoleh bervariasi. Data yang diperoleh merupakan video dengan format .mp4 sehingga harus dilakukan transformasi citra terlebih dahulu agar bisa diproses ke tahap selanjutnya. Sistem menggunakan metode *deep learning* yaitu CNNs dengan 5 *layer* yang terdiri dari 2 *layer* konvolusi dan 3 layer *fully connected*. Untuk memudahkan proses klasifikasi, data melakukan praproses terlebih dahulu yang terdiri dari pelabelan data, *scalling*, konversi citra, dan normalisasi. Hasil dari percobaan dengan menggunakan 3000 data yang terdiri dari 2 kelas data yaitu motor dan non motor menunjukkan akurasi sebesar 95,94%. Hasil akurasi tersebut menunjukkan bahwa sistem sudah cukup baik dalam mendeteksi sepeda motor pelanggar marka jalan.

Kata Kunci: Deteksi Sepeda Motor, Deteksi Pelanggar, *Convolutional Neural Networks, Deep Learning*

Abstract

The large amount traffic of that happens now a day, whether by motorist or car driver is reflection of the personality of a nation. This violation happen due to lack of discipline of drivers in obeying traffic sign and road marking that exist. Traffic violations is a serious problem because it can occur traffic accident. The officers cannot supervise drivers for 24 hours, therefore the author makes motorcycle detection system that violated road markings using Convolutional Neural Networks (CNNs). The detected violation system is when a motorcycle stops passing through the stop line when the red light is activated. The data that used in this research is a video that author got from Departement of Transportation Bandung wich was taken for 10 days, so the videos that obtained have a lot of variety. The obtained data is .mp4 video so we must transformed the video to image first. The system uses the deep learning method, that is CNNs with 5 layers consisting of 2 convolution layers and 3 layers fully connected. Before make classification, we must make the data praprocesses. The data praprocesses consist of data labeling, scalling, image conversion, and normalization. The results of the experiment using 3000 data consisting of 2 class data that is motorcycle and non motorcycle showed accuracy 95.94%. The results of this accuracy indicate that the system is quite good at detecting motorbike off road markers.

Key Words: Vehicle detection, Violator Detection, Convolutional Neural Networks, Deep Learning

Daftar Isi

PERNYATAAN i
KATA PENGANTAR ii
UCAPAN TERIMA KASIH iii
Abstrak iv
Abstract v
Daftar Isi vi
Daftar Tabel viii
Daftar Gambar ix
BAB I 11
PENDAHULUAN 11
1.1. Latar belakang penelitian 11
1.2. Rumusan masalah penelitian 14
1.3. Tujuan penelitian 14
1.4. Manfaat penelitian 14
1.5. Batasan masalah penelitian 14
1.6. Sistematika penulisan 15
BAB II 17
KAJIAN PUSTAKA 17
2.1. Jenis-jenis marka jalan 17
2.2. Perkembangan *machine learning* dalam mengatasi permasalahan kehidupan sehari-hari 19
2.3. Klasifikasi *deep learning* 20
2.4. *Convolutional Neural Networks* (CNNs) 22
2.5. Arsitektur *Convolutional Neural Networks* (CNNs) 25
2.6. *Deep Learning Open-Source Frameworks* 29
2.7. Image types 31
2.8. Image enhancement methods 33
2.9. *Vehicle detection system* 39
BAB III 41
METODOLOGI PENELITIAN 41
1.1. Desain Penelitian 41

1.2. Perangkat Keras dan Perangkat Lunak 47
1.3. Data Penelitian 47
1.3.1. Data *input* 47
1.3.2. Data *output* 47
BAB IV 49
HASIL PENELITIAN DAN PEMBAHASAN 49
1.1. Pengumpulan data 49
1.2. Sistem pendeteksi sepeda motor pelanggar marka jalan 50
1.2.1. Pengambilan dan transformasi citra 50
1.2.2. Menentukan *stop line* 51
1.2.3. Pra proses 52
1.2.4. *Training* dan *testing dengan CNNs* 56
1.2.5. Mendeteksi sepeda motor 59
1.2.6. Mendeteksi sepeda motor pelanggar marka jalan 62
1.3. Pengembangan perangkat lunak 62
1.3.1. Analisis sistem 62
1.3.2. Deskripsi sistem 63
1.3.3. Batasan perangkat lunak 64
1.3.4. Implementasi *coding* 64
1.4. Pengujian 65
1.4.1. Skenario pengujian 65
1.4.2. Hasil pengujian 66
1.4.3. Pembahasan hasil penelitian 72
BAB V 77
PENUTUP 77
5.1. Kesimpulan 77
5.2. Saran 78
DAFTAR PUSTAKA 79

Daftar Tabel

Tabel 1.1. Data statistik pelanggaran di persimpangan Kota Bandung 12
Tabel 4.1. Representasi kelas data 53
Tabel 4.2. Hasil pengujian dari delapan skenario pengujian 67
Tabel 4.3. Rata-rata akurasi dengan menggunakan 5-*fold* 68
Tabel 4.4. Rata-rata akurasi dengan variasi data *training* dan *testing* 68
Tabel 4.5. *Confusion matrix* pada setiap skenario pengujian 70

Daftar Gambar

Gambar 2.1. Marka membujur (Kementrian Perhubungan, 2014) 17
Gambar 2.2. Marka melintang (Kementrian Perhubungan, 2014)..... 18
Gambar 2.3. Marka kotak kuning (Kementrian Perhubungan, 2014)..... 18
Gambar 2.4. Proses konvolusi (Claesson & Hansson, 2017)..... 23
Gambar 2.5. Proses *Rectified Linear Units Layer* (Claesson & Hansson, 2017).. 24
Gambar 2.6. Proses *max pooling* (Claesson & Hansson, 2017)..... 25
Gambar 2.7. Arsitektur LeNet-5 256
Gambar 2.8. Arsitektur AlexNet (Liliana et al., 2017) 256
Gambar 2.9. Arsitektur VGG16 (Liliana et al., 2017)..... 257
Gambar 2.10. Building block GoogleNet (Liliana et al., 2017)..... 28
Gambar 2.11. *Residual connection in ResNet* (Liliana et al., 2017)..... 29
Gambar 2.12. Tipe citra berdasarkan kompsisi warna (Kumar & Verma, 2010) . 32
Gambar 2.13. Hasil proses *negative image* (Vandra & Kulkarni, 2012)..... 35
Gambar 2.14. Operasi *scalling*..... 35
Gambar 2.15. Hasil proses *tresholding* (Maini & Aggarwal, 2010)..... 36
Gambar 2.16. Citra hasil proses dilasi dan erosi (Goodman & Rhodes, 2009) 37
Gambar 2.17. Citra hasil proses *logarithmic* (Maini & Aggarwal, 2010) 37
Gambar 2.18. Citra hasil proses *histogram equalization* (Yao et al., 2017)..... 378
Gambar 2.19. Citra hasil *fuzzy enhaced* (Thakur & Mishra, 2015) 378
Gambar 3.1. Desain Penelitian..... 41
Gambar 3.2. Alur sistem 42
Gambar 3.3. Arsitektur CNNs..... 44
Gambar 3.4. *Waterfall* model (Bassil, 2012) 45
Gambar 4.1. Contoh dari video rekaman CCTV..... 49
Gambar 4.2. Alur proses sistem pendeteksi sepeda motor pelanggar marka jalan 50
Gambar 4.3. Contoh data citra motor..... 51
Gambar 4.4. Contoh data citra non motor..... 51
Gambar 4.5. Hasil penambahan *stop line*..... 52
Gambar 4.6. Alur pra proses 53
Gambar 4.7. Contoh hasil proses *scalling*..... 54

Gambar 4.8. Contoh hasil proses konversi citra ke *grayscale* 54
Gambar 4.9. Gambar perubahan citra setelah proses *histogram equalization*..... 55
Gambar 4.10. Representasi nilai matriks dari citra 56
Gambar 4.11. Representasi nilai matriks hasil normalisasi 56
Gambar 4.12. Arsitektur CNNs pada sistem pendeteksi pelanggar marka jalan .. 57
Gambar 4.13. Representasi citra 58
Gambar 4.14. Matriks *kernel* 3x3 58
Gambar 4.15. Representasi citra hasil konvolusi 58
Gambar 4.16. Representasi matriks hasil proses ReLu 59
Gambar 4.17. Hasil proses *subsampling* 59
Gambar 4.18. Alur proses mendeteksi sepeda motor..... 60
Gambar 4.19. Proses perubahan yang terjadi pada tahap *denoising* 61
Gambar 4.20. Contoh hasil sistem saat mendeteksi sepeda motor 61
Gambar 4.21. Contoh sistem saat mendeteksi pelanggar marka jalan..... 62
Gambar 4.22. Contoh sistem saat berhasil mendeteksi pelanggar marka jalan 72
Gambar 4.23. Kesalahan sistem yang disebabkan hanya sebagian roda depannya saja yang melewati *stop line* 73
Gambar 4.24. Kesalahan sistem yang disebabkan tertutup oleh kendaraan lain .. 74
Gambar 4.25. Grafik akurasi berdasarkan pembagian data menggunakan 5 *fold*. 75
Gambar 4.26. Grafik akurasi berdasarkan jumlah data *training* 76

BAB I
PENDAHULUAN

1.1. Latar belakang penelitian

Salah satu permasalahan yang paling menantang dan paling rumit dalam manajemen kota khususnya pada negara yang masih berkembang adalah masalah lalu lintas (Shamsher, Mohamamd, & Abdullah, 2013). Kemacetan dan kecelakaan lalu lintas merupakan permasalahan serius yang sudah menjadi permasalahan *global*. Hal ini menarik perhatian dari berbagai kalangan mulai dari ahli tata kota, pemerintah, teknisi hingga para peneliti untuk menemukan solusi dari kemacetan (Lindley, 1987). Keselamatan berkendara dipengaruhi oleh 3 faktor yaitu kualitas kendaraan, infrastruktur dan kondisi pengendara (Castellà & Pérez, 2004). Selain itu kebiasaan buruk pengendara seperti melanggar lalu lintas juga menjadi faktor penting yang mempengaruhi terjadinya kecelakaan. James Elander (Elander, West, & French, 1993) menyebutkan bahwa hampir dari 90% permasalahan yang ada di jalan raya disebabkan oleh kesalahan pengendaranya.

Sama halnya dengan di Indonesia khususnya di kota Bandung pelanggaran lalu lintas sudah menjadi hal yang biasa ditemukan. Berdasarkan data yang diperoleh dari Dinas Perhubungan Kota Bandung selama lima bulan terakhir (tabel 1.1) yaitu dari Bulan Mei hingga September tahun 2018 terjadi lebih dari 11 ribu pelanggaran lalulintas dengan mayoritas pelanggar merupakan pengguna sepeda motor. Pelanggaran yang sering terjadi diantaranya kendaraan yang berhenti melebihi *stop line* atau berhenti di area *zebra cross*, pengendara sepeda motor yang tidak menggunakan helm, mobil yang berhenti di ruang henti khusus motor, serta kendaraan yang kelebihan muatan. Persimpangan dengan jumlah pelanggaran terbanyak selama lima bulan terakhir terjadi di persimpangan jalan Caringin.

Jika melihat peraturan yang ada para pelanggar marka jalan berkewajiban membayar denda paling banyak Rp 500.000,00 atau pidana kurungan paling lama selama 2 bulan sesuai dengan yang tertulis pada Undang-Undang Nomor 22 tahun 2009 tentang Lalu Lintas dan Angkutan Jalan Pasal 279 “Setiap orang yang mengemudikan kendaraan bermotor di jalan yang melanggar aturan perintah atau larangan yang dinyatakan dengan rambu lalu lintas atau marka jalan dipidana dengan pidana kurungan paling lama 2 (dua) bulan atau denda paling banyak Rp

500.000,00 (lima ratus ribu rupiah). Dengan adanya peraturan tersebut seharusnya sudah cukup untuk membuat pengendara tidak melakukan pelanggaran. Pada kenyataannya hukum tersebut belum dapat diaplikasikan secara maksimal karena keterbatasan petugas yang tidak mampu bekerja selama 24 jam di seluruh jalan raya membuat para pengendara tidak dapat terus diawasi. Keterbatasan tersebut yang membuat banyak dari para pelanggar yang tidak dihukum sesuai dengan Undang-Undang yang ada sehingga membuat kebiasaan buruk pengendara yaitu melanggar rambu lalu lintas terus dilakukan.

Tabel 1.1. Data statistik pelanggaran di persimpangan Kota Bandung

Periode Pencatatan (2018)	**Kategori Pelanggaran Yang Sering Terjadi**				
	Berhenti Melebihi *Stop Line*	**Tidak Menggunakan Helm**	**Mobil Berhenti di RHK**	**Kelebihan Penumpang**	**Lainnya**
Mei	785	64	226	4	74
Juni	2541	152	621	9	116
Juli	834	44	225	4	17
Agustus	2584	310	547	11	48
September	2178	272	163	38	106

Beberapa tahun terakhir ini lahir berbagai teknologi untuk membantu manusia dalam berbagai aspek kehidupan. Pada bidang transportasi sendiri berkembang sebuah sistem *intelligent traffic surveillance* yang bekerja dengan memantau gambar pada CCTV untuk memperoleh informasi dan memantau lalu lintas (Zhang, Zhou, & Pan, 2013). Analisis video CCTV dewasa ini berkembang sangat cepat dalam bidang *computer vision* (Buch, Velastin, & Orwell, 2011). Hal tersebut disebabkan oleh peningkatan infrastruktur salah satunya pemasangan CCTV yang hampir diseluruh sudut kota. Selain itu kemajuan teknik analitik dan pemrosesan data video memungkinkan lahirnya aplikasi sistem pengawasan berbasis *computer vision* untuk mengekstrasi informasi dari video.

Sistem pendeteksi kendaraan sudah banyak diteliti sebelumnya dengan menggunakan metode yang berbeda beda seperti yang dilakukan oleh Kafai dengan

menggunakan *Hybrid Dynamic Bayesian Networks* (Kafai & Bhanu, 1983) yang mengklasifikasikan kendaraan kedalam 3 jenis dan menghasilkan akurasi sebesar 97,63%. Selain itu (Mussa, Kwigizile, & Selekwa, 2006) melakukan klasifikasi kendaraan menggunakan *Probabilistic Neural Networks* (PBNs) yang nilai *error* nya hanya sebesar 6,2%, serta Bo Li yang mengklasifikasikan 3D objek menggunakan 3D *Fully Convolutional Network* (Li, 2017) dengan rata-rata presisi sebesar 84,1%.

Berdasarkan hasil kajian dari penelitian sebelumnya menunjukkan kasus pendeteksian kendaraan dengan menggunakan pendekatan *deep learning* dianggap paling efektif. Hal itu disebabkan karena pada pendekatan *deep learning* proses ekstraksi fitur tidak dilakukan oleh *human engineers* seperti yang dilakukan oleh *Suport Vector Machine* (SVM) dengan menggunakan *Histogram of Gradient* (HOG), melainkan dengan melakukan proses *learning* data menggunakan *general purpose learning procedure* (Lecun, Bengio, & Hinton, 2015). Salah satu metode dalam *deep learning* adalah *Convolutional Neural Networks* (CNNs).

CNNs merupakan salah satu metode yang banyak digunakan untuk proses klasifikasi citra karena metode ini memang didesain untuk memproses *multiple array* seperti *signal*, citra dan video. Keunggulan metode ini adalah mudah untuk dibangun karena memiliki arsitektur yang sederhana yaitu terdiri dari *convolution layer* dan *fully connected layer* dan mudah untuk digunakan karena untuk meningkatkan akurasi dapat dilakukan dengan mengubah kedalaman *layer*. Metode ini memiliki tingkat ketelitian yang tinggi, hal ini dimungkinkan karena informasi pada CNNs tidak diprogram, melainkan dilatih dengan menggunakan proses konvolusi dan *pooling* berdasarkan informasi yang diterima (Chellapilla, Puri, & Simard, 2006).

Pada penerapannya CNNs sudah terbukti menghasilkan performa yang baik seperti yang telah ditunjukkan pada penelitian yang dilakukan oleh (Stallkamp, Schlipsing, Salmen, & Igel, 2012) yang melakukan penelitian mengenai *traffic sign recognition* dengan menggunakan 50000 data set yang terbagi kedalam 43 kelas data dan menghasilkan akurasi sebesar 99,46%. Selain itu CNNs digunakan pada penelitian yang dilakukan oleh Bautista (Bautista, Dy, & Manalac, 2016) dan menghasilkan nilai akurasi sebesar 94,72%.

Sistem pendeteksi kendaraan yang sudah banyak dilakukan hanya mendeteksi jenis kendaraan saja tidak sampai mendeteksi kendaraan yang melanggar marka jalan. Maka dari itu dalam penelitian ini penulis mengimplementasikan metode *Convolutional Neural Networks* pada sistem pendeteksi sepeda motor pelanggar marka jalan yang mampu beroperasi selama 24 jam dengan tujuan untuk membantu petugas dalam pengawasan lalu lintas. Sistem ini berbasis *desktop* yang akan memeriksa video CCTV pada seluruh persimpangan jalan. Sehingga saat ada pelanggar yang terdeteksi oleh sistem, petugas bisa menindaklanjuti pelanggar tersebut agar tidak mengulangi kesalahannya lagi dan membuat kedisiplinan pengendara meningkat serta angka pelanggaran lalu lintas bisa berkurang.

1.2. Rumusan masalah penelitian

1. Bagaimana implementasi metode *Convolutional Neural Networks* (CNNs) untuk mendeteksi sepeda motor di jalan raya?
2. Bagaimana mendeteksi sepeda motor yang melanggar marka jalan?

1.3. Tujuan penelitian

1. Mengimplementasikan metode Convolutional Neural Networks (CNNs) untuk mendeteksi sepeda motor di jalan raya.
2. Mengetahui cara mendeteksi sepeda motor yang melanggar marka jalan.

1.4. Manfaat penelitian

1. Dapat memberikan informasi kepada petugas lalu lintas mengenai sepeda motor yang melanggar marka jalan.
2. Dapat membantu pemerintah dalam meningkatkan kedisiplinan pengendara sepeda motor.
3. Dapat membantu menurunkan angka pelanggaran yang terjadi.

1.5. Batasan masalah penelitian

1. Jenis pelanggaran marka jalan yang di deteksi yaitu pelanggar yang melewati *stop line*.

2. Data yang digunakan merupakan data yang diperoleh dari rekaman CCTV Dinas Perhubungan Kota bandung dengan resolusi 1280 x 720 *pixel*.

1.6. Sistematika penulisan

Sistematika penulisan yang akan disampaikan pada penelitian ini, yaitu:

BAB I PENDAHULUAN

Bab I merupakan pendahuluan dari skripsi ini yang terdiri dari sub bab latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika penulisan. Pada sub bab latar belakang menyampaikan alasan penulis mengangkat judul "Sistem Pendeteksi Sepeda Motor Pelanggar Marka Jalan Menggunakan Metode *Convolutional Neural Networks*" untuk diteliti sebagai skripsi. Di rumusan masalah disebutkan hal-hal apa saja yang menjadi permasalahan dalam penelitian ini. Pada sub bab tujuan menyebutkan tujuan dari dilakukannya penelitian ini. Manfaat penelitian dijelaskan untuk memberitahukan manfaat yang didapat setelah melakukan penelitian ini. Selain itu batasan masalah berisikan batasan-batasan dalam penelitian, dan sistematika penulisan menggambarkan isi dari penelitian ini.

BAB II KAJIAN PUSTAKA

Pada kajian pustaka akan diuraikan materi-materi yang berhubungan dengan penelitian. Materi ini yang mendasari penulis dalam melakukan penelitian. Materi yang disampaikan meliputi hakikat lalu lintas, jenis-jenis marka jalan, tata tertib lalu lintas, perkembangan *machine learning* dalam mengatasi permasalahan sehari-hari, klasifikasi *deep learning*, *Convolutional Neural Networks* (CNNs), arsitektur CNNs, *open source frameworks* yang digunakan untuk *deep learning*, serta penelitian terdahulu mengenai *vehicle detection system*.

BAB III METODOLOGI PENELITIAN

Bab ini berisi tahapan penelitian yang digambarkan pada desain penelitian, alat dan bahan yang digunakan dalam penelitian. Pada bab ini juga menjelaskan mengenai data penelitian baik itu data *input* maupun data *output*.

BAB IV PEMBAHASAN

Bab pembahasan menjelaskan bagaimana penelitian dilakukan sesuai tahapan yang telah dibuat pada desain penelitian. Setiap alur yang ada pada desain penelitian dijelaskan secara berurutan. Pada bab ini juga dijelaskan skenario pengujian yang dilakukan beserta hasil pengujiannya.

BAB V KESIMPULAN

Bab kesimpulan berisi tentang rangkuman dari hasil penelitian yang telah dilakukan. Selain kesimpulan, pada bab 5 disampaikan saran untuk penelitian selanjutnya.

LAMPIRAN

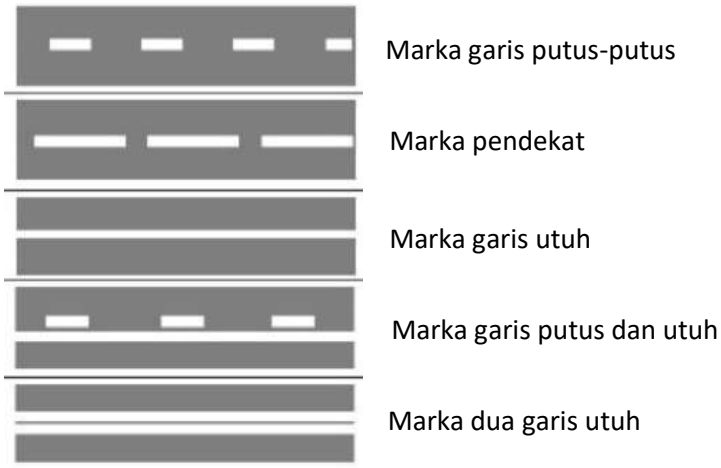
BAB II
KAJIAN PUSTAKA

2.1. Jenis-jenis marka jalan

Menurut pasal 1 Undang-Undang nomor 22 tahun 2009 tentang lalu lintas dan angkutan jalan di *point* 18 dijelaskan bahwa marka jalan adalah suatu tanda yang berada di permukaan jalan atau di atas permukaan jalan yang meliputi peralatan atau tanda yang membentuk garis membujur, garis melintang, garis serong, serta lambang yang berfungsi untuk mengarahkan arus lalu lintas dan membatasi daerah kepentingan lalu lintas. Marka jalan berfungsi untuk mengatur lalu lintas, memperingatkan, atau menuntun pengguna jalan dalam berlalu lintas.

Dalam Peraturan Menteri Perhubungan Republik Indonesia nomor PM 34 tahun 2014 tentang marka jalan dijelaskan bahwa marka jalan berupa sebuah tanda yang meliputi:

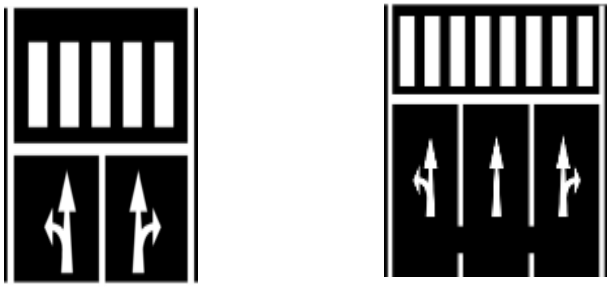
1. Marka membujur



Gambar 2.1. Marka membujur (Kementrian Perhubungan, 2014)

Marka membujur adalah marka jalan yang sejajar dengan sumbu jalan, terdiri atas garis utuh, garis putus-putus, garis ganda yang terdiri dari garis utuh dan garis putus-putus dan garis ganda yang terdiri dari dua garis utuh. Marka membujur berfungsi sebagai larangan bagi kendaraan melintasi garis tersebut dan pembatas atau pembagi jalur (gambar 2.1).

2. Marka melintang



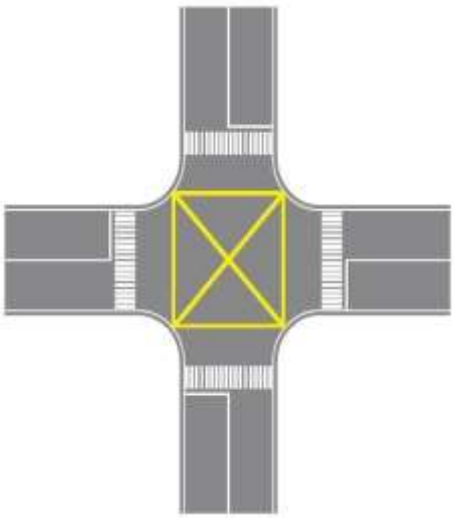
(a) Garis henti pada persimpangan jalan 1 arah

(b) Garis henti pada penyebrangan orang (*zebra cross)*

Gambar 2.2. Marka melintang (Kementrian Perhubungan, 2014)

Marka melintang adalah marka jalan yang tegak lurus terhadap sumbu jalan yang menyatakan batas berhenti kendaraan yang diwajibkan berhenti oleh alat pemberi isyarat lalu lintas, rambu berhenti, tempat penyebrangan, atau *zebra cross* (gambar 2.2).

3. Marka kotak kuning



Gambar 2.3. Marka kotak kuning (Kementrian Perhubungan, 2014)

Marka kotak kuning berbentuk segi empat dengan 2 garis diagonal berpotongan dan berwarna kuning yang berfungsi untuk melarang kendaraan berhenti di suatu area (gambar 2.3).

Bila berdasarkan warnanya marka jalan dikelompokkan menjadi 4, yaitu:

1. Marka jalan berwarna putih menyatakan bahwa pengguna jalan wajib mengikuti perintah atau larangan sesuai dengan bentuknya
2. Marka jalan berwarna kuning menyatakan bahwa pengguna jalan dilarang berhenti pada area tersebut
3. Marka jalan berwarna merah menyatakan keperluan atau tanda khusus
4. Marka jalan dengan warna lainnya yaitu hijau dan coklat menyatakan daerah kepentingan khusus yang harus dilengkapi dengan rambu dan atau petunjuk yang dinyatakan dengan tegas.

2.2. Perkembangan *machine learning* dalam mengatasi permasalahan kehidupan sehari-hari

Di era modern sekarang ini hampir seluruh permasalahan yang ada dapat teratasi oleh teknologi. *Machine learning* hadir sebagai salah satu pendekatan yang mampu mengatasi permasalahan yang ada seperti munculnya sistem rekomendasi pada situs *e-commerce* dimana secara otomatis sebuah situ *e-commerce* mampu memberikan rekomendasi barang yang kemungkinan besar akan kita cari. Namun *machine learning* tidak mampu menyelesaikan semua permasalahan tersebut karena *machine learning* memiliki keterbatasan dalam pengolahan data. (Lecun et al., 2015).

Meniru ketepatan dan keefisienan otak manusia dalam merepresentasikan sebuah informasi menjadi tantangan tersulit yang dialami oleh para peneliti di bidang *artificial intelligence*. Pada dasarnya otak manusia terdiri dari miliaran sel syaraf yang menerima data setiap detiknya. Lebih dari 50 tahun lalu Richard Bellman yang memperkenalkan *dynamic programming* mengatakan bahwa kesulitan utamanya adalah mempelajari kompleksitas pertumbuhan data (Arel, Rose, & Karnowski, 2010).

Deep learning merupakan ranah baru dalam penelitian *machine learning*, dimana dengan lahirnya *deep learning* ini semakin mendekatkan kita menuju ke tujuan dari *artificial intelligent* yaitu bertindak dan berfikir seperti manusia. Dengan metode *deep learning* kita bisa menemukan banyak informasi abstraksi dan

representasi untuk menghasilkan data berupa teks, suara dan gambar yang serupa dengan aslinya (Rere, Fanany, & Arymurthy, 2015).

Deep learning mengacu pada *machine learning* dimana terdapat banyak lapisan tahapan pemrosesan informasi dalam arsitekturnya yang digunakan untuk pembelajaran pada fitur pengklasifikasian pola dan representasi obyek. Diantara banyaknya perbedaan mengenai deskripsi *deep learning*, pada umumnya *deep learning* terdiri dari 2 aspek utama yaitu merupakan sebuah model yang terdiri dari beberapa lapisan atau tahapan pengolahan non linear, dan aspek yang kedua adalah metode *supervised* atau *unsupervised learning* dengan kemampuan representasi tinggi pada lapisan yang abstrak. Tiga alasan pentingnya menggunakan *deep learning* adalah kemampuan pemrosesan *chip* yang meningkat drastis misalnya peningkatan pada GPU, menurunnya waktu pemrosesan yang sangat signifikan, dan kemampuan baru dalam pemrosesan sinyal atau informasi (Deng & Yu, 2014).

2.3. Klasifikasi *deep learning*

Pada umumnya *deep learning* bisa diklasifikasikan kedalam dikriminatif model dan generatif model. Dalam diskriminatif model terdapat beberapa model, yaitu *Deep Neural Networks* (DNNs), R*ecurrent Neural Network* (RNNs), dan C*onvolutional Neural Networks* (CNNs). Sedangkan pada generatif model terdapat *Boltzmann Machine* (RBMs), *Deep Belief Networks* (DBNs), *regularized autoencoders*, and *deep Boltzmann machines* (DBMs) (Rere et al., 2015).

Seperti yang sudah dijelaskan diatas *deep learning* mengacu pada *machine learning* dimana terdapat banyak lapisan tahapan pemrosesan informasi non linear yang bersifat hirarkis bergantung pada arsitekturnya. Sedangkan menurut (Deng, 2013; Deng & Yu, 2014) *deep learning* bisa dibagi menjadi 3 kelas utama, yaitu:

1. *Deep learning* untuk *unsupervised* atau generatif *learning*

   Unsupervised learning mengacu pada ada atau tidaknya informasi spesifik misalnya label target kelas dalam proses pembelajaran. Penggunaan aturan *bayes* dapat mengubah arsitektur ini menjadi diskriminatif. Ada banyak metode yang dapat digunakan pada kategori ini, yaitu:

a. *Deep Belief Network* (DBN)

   Model generatif probabilistik yang terdiri dari beberapa lapisan variabel stokastik yang tersembunyi. Terdiri atas dua lapisan yang memiliki hubungan simetris namun tidak beriringan di antara keduanya. Lapisan bawah menerima koneksi secara *top-down* dari lapisan atas.

b. *Boltzmann Machine* (BM)

   Jaringan yang terhubung secara simetris, seperti *neuron* yang membuat keputusan stokastik tentang jalan mana yang harus diambil hidup atau mati.

c. *Restricted Boltzmann Machine* (RBM)

   Jenis khusus dari *Boltzmann Machine* (BM) yang terdiri dari lapisan unit terlihat dan lapisan unit tersembunyi dengan tidak ada koneksi yang terlihat maupun tersembunyi.

d. *Deep Neural Network* (DNN)

   Merupakan *perceptron multilayer* dengan banyak lapisan tersembunyi, yang bobotnya saling terhubung sepenuhnya dan sering (meskipun tidak selalu) diinisialisasi menggunakan teknik *unsupervised* atau *supervised pretraining*.

e. *Deep autoencoder*

   Merupakan diskriminatif dari *Deep Neural Network* (DNN) yang target produksinya adalah input data itu sendiri bukan label kelasnya, maka termasuk model *unsupervised learning*. Saat dilatih menggunakan *denoising*, *deep autoencoder* juga merupakan model generatif dan bisa menjadi sebuah contoh.

f. *Distributed representation*

   Sebuah representasi internal dari data observasi menjadi sedemikian rupa sehingga data tersebut dapat memberikan informasi dari banyak hal yang tersembunyi. Sebuah faktor mampu belajar dari faktor lain sehingga bisa mengeneralisasikan konfigurasi baru dengan baik. Representasi terdistribusi secara alami terjadi di *neural networks*, dimana konsep ini merepresentasikan sebuah pola aktivitas di

sejumlah unit dan pada waktu dan tempat yang bersamaan sebuah unit mampu berkomunikasi dengan banyak konsep.

2. *Deep learning* untuk *supervised learning* atau *discriminatif learning*

   Pada diskriminatif *learning* pemberian label langsung diberikan pada data yang sudah ada. Ada banyak diskriminatif teknik untuk *supervised learning* dalam memproses *signal* atau informasi, seperti HMMs, *Condisitional Random Fields* (CRFs), *Recurrent Neural Networks* (RNNs), dan *Convolutional Neural Network* (CNNs)

3. *Hybrid deep learning*

   Yang dimaksud *hybrid* pada kategori ketiga ini adalah *deep architecture* yang menggabungkan dua model yaitu model generatif dan model diskriminatif. Pada arsitektur ini komponen generatif digunakan untuk membantu komponen diskriminatif dalam mencapai tujuan dari arsitektur *hybrid* ini. Ada dua tahapan cara bagaimana model generatif mampu membantu model diskriminatif, yaitu:

   a. Optimisasi *viewpoint*, dimana model generatif mampu melatih model diskriminatif sehingga menghasilkan titik inisiasi dengan akurasi tinggi pada parameter non linear (biasa disebut *pre-training*, hal inilah yang menjadi alasan digunakannya *hybrid deep learning*)
   b. Perspektif regulasi, dimana model *unsupervised learning* dapat secara efektif memberikan sebuah model fungsi yang *representable*.

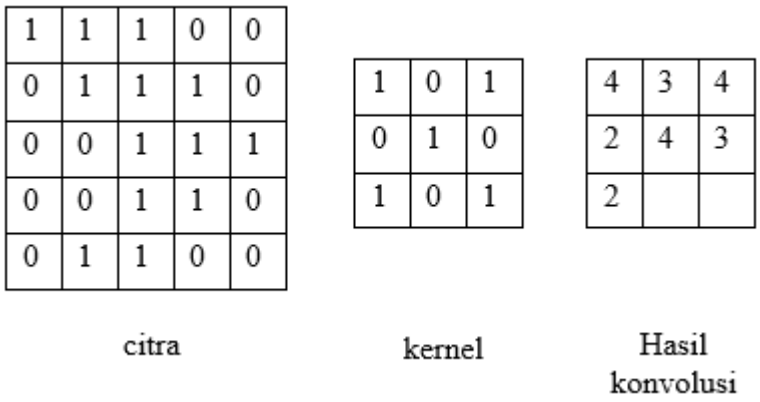
2.4. *Convolutional Neural Networks* (CNNs)

Convolutional Neural Networks (CNNs) lahir dari hasil pengembangan *Multilayer Perceptron* (MLP). CNNs merupakan salah satu model *deep learning* yang didesain untuk mengolah data dua dimensi yang biasa diaplikasikan pada data citra dan suara (Putra, 2016). CNNs diperkenalkan oleh Fukushima dengan nama *neocognitron* dimana jaringan ini mampu belajar sendiri dan mampu mengenali pola berdasarkan kesamaan geometris (Fukushima, 1980). *Neocognitron* mempunyai struktur yang menyerupai model visual jaringan syaraf tiruan milik Hubel dan Wiesel dimana *neocognitron* memiliki *input layer*, dan dua *layer* lain yang disebut "*S-cells*" dan "*C-cells*".

CNNs memiliki beberapa keunggulan diantara arsitektur lainnya, yaitu CNN mudah melakukan *training* dan mudah diimplementasikan karena memiliki parameter yang lebih sedikit dari arsitektur lainnya. CNNs juga memiliki tingkat akurasi yang tinggi dimana tingakat akurasi dapat dikontrol sesuai kedalaman serta lebar dari *layer* yang dibuat (Krizhevsky, Sutskever, & Hinton, 2012). *Convolutional Neural Networks* terdiri dari beberapa *layer*, berikut merupakan *layer* yang umum terdapat pada CNNs (Claesson & Hansson, 2017):

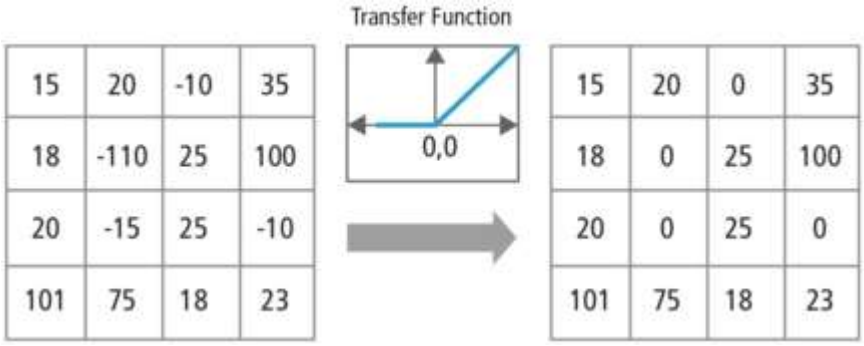
a. *Convolution layer*

Pada *layer* ini dilakukan operasi konvolusi pada *output* dari *layer* sebelumnya. Tujuan dilakukannya konvolusi adalah untuk mengekstraksi fitur dari citra *input*. Operasi konvolusi yaitu mengaplikasikan *kernel* pada citra di setiap *offset* yang memungkinkan. Setiap pergerakan *kernel* dilakukan proses penjumlahan dari perkalian setiap titik pada kernel dengan setiap titik pada citra sehingga menghasilkan sebuah matriks baru (gambar 2.4).



Gambar 2.4. Proses konvolusi (Claesson & Hansson, 2017)

b. *Rectified Linear Units Layer* (RELU)

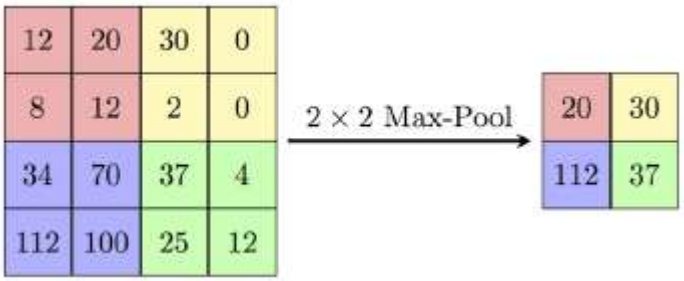


Gambar 2.5. Proses *Rectified Linear Units Layer* (Claesson & Hansson, 2017)

Rectified Linear Unit merupakan fungsi aktivasi yang sering digunakan pada model *deep learning* dibandingkan fungsi aktivasi lain seperti *sigmoid*. RELU berfungsi untuk membuat sebuah *neural network* menjadi non-linear. RELU akan melemparkan nilai 0 jika masukannya bernilai negatif dan akan melemparkan nilai yang sama jika masukkannya bernilai positif atau bisa ditulis *f(x) = max(0,x)*. Pada gambar 2.5 merupakan contoh bagaimana RELU mengubah semua nilai negatif menjadi 0.

c. *Subsampling layer*

Pada *layer* ini dilakukan proses reduksi citra. Proses ini bertujuan untuk mengurangi dimensi dari *feature map* (*downsampling*), sehingga mempercepat komputasi karena parameter yang harus diperbaharui semakin sedikit. Metode *subsampling* yang biasa digunakan yaitu *max pooling*. Metode ini bekerja dengan cara membagi *output* dari lapisan konvolusi menjadi beberapa *grid* kecil kemudian mengambil nilai maksimum dari setiap *grid* sehingga dihasilkan matriks baru (gambar 2.6).



Gambar 2.6. Proses *max pooling* (Claesson & Hansson, 2017)

d. *Fully connected layer*

Fully connected layer adalah *layer* yang digunakan pada MLP. *Layer* ini bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. *Layer* ini biasanya terdapat pada lapisan terakhir.

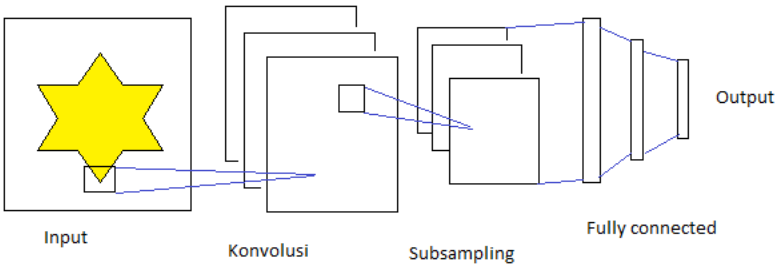
e. *Loss layer*

Loss layer merupakan cara sederhana untuk mengurangi *overfitting*. Selain untuk mengurangi *overfitting*, *layer* ini juga dapat mengurangi jumlah perhitungan yang harus dilakukan sehingga memungkinkan untuk menghasilkan kinerja yang lebih baik.

2.5. Arsitektur *Convolutional Neural Networks* (CNNs)

Ada banyak arsitektur CNN yang telah digunakan dan terbukti memiliki tingkat akurasi yang bagus, berikut beberapa arsitektur yang sering digunakan dalam klasifikasi citra (Liliana, Baji, & Teodoru, 2017):

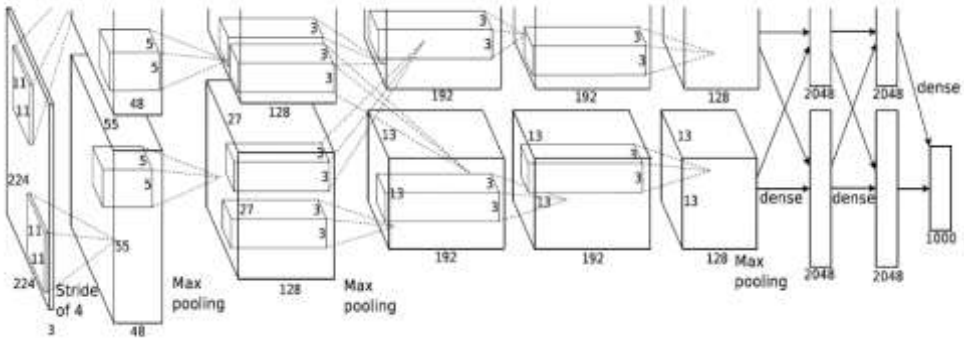
a. LeNet-5



Gambar 2.7. Arsitektur LeNet-5

LeNet-5 merupakan arsitektur CNN yang dikenalkan oleh LeCun pada tahun 1998. LeNet-5 terdiri dari 2 lapisan konvolusi, 2 lapisan *subsampling* dan 2 lapisan *fully connected* (gambar 2.7). Pada lapisan konvolusi menggunakan filter 5x5 sedangkan pada lapisan *subsampling* menggunakan filter 2x2.

b. AlexNet



Gambar 2.8. Arsitektur AlexNet (Liliana et al., 2017)

AlexNet merupakan arsitektur CNN yang dikembangkan oleh Alex Krizhevsky, Ilya Sutskever dan Geoff Hinton pada tahun 2012. Arsitektur AlexNet hampir serupa dengan arsitektur LeNet milik LeCun, hanya saja arsitektur AlexNet lebih dalam, lebih besar dan lapisan konvolusi yang saling terhubung, berbeda dengan arsitektur lainnya yang hanya memiliki satu lapisan konvolusi yang selalu diikuti dengan lapisan *subsampling.*

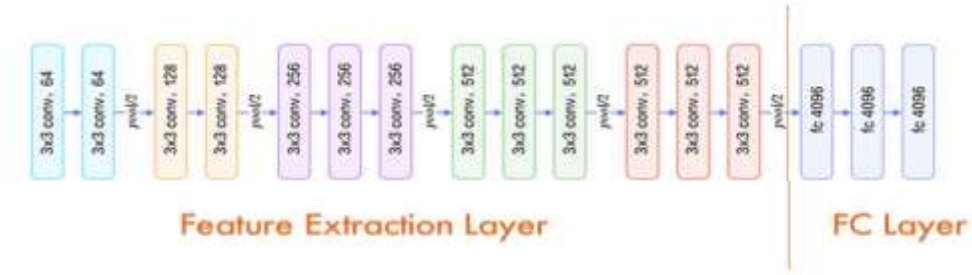
AlexNet memiliki 8 *layer*, yaitu 3 *layer* konvolusi, 2 *layer subsampling* dan 3 *layer fully connected* (gambar 2.8). *Layer* konvolusi memiliki 96 filter dengan ukuran 11x11 dan 4 *stride*. Sedangkan *layer subsampling* menggunakan filter 3x3 dengan 2 *stride*. Pada *layer subsampling* pertama menggunakan ReLU sedangkan *layer subsampling* kedua menggunakan *normalization*.

c. ZFNet

Sesuai dengan namanya ZFNet dipopulerkan oleh Zeiler dan Fergus pada tahun 2013. Arsitektur ZFNet merupakan improvisasi dari arsitektur sebelumnya yaitu AlexNet. Zeiler dan Fergus hanya mengubah filter

pada *layer* konvolusi dari yang awalnya 11x11 *stride* 4 menjadi 7x7 *stride* 2.

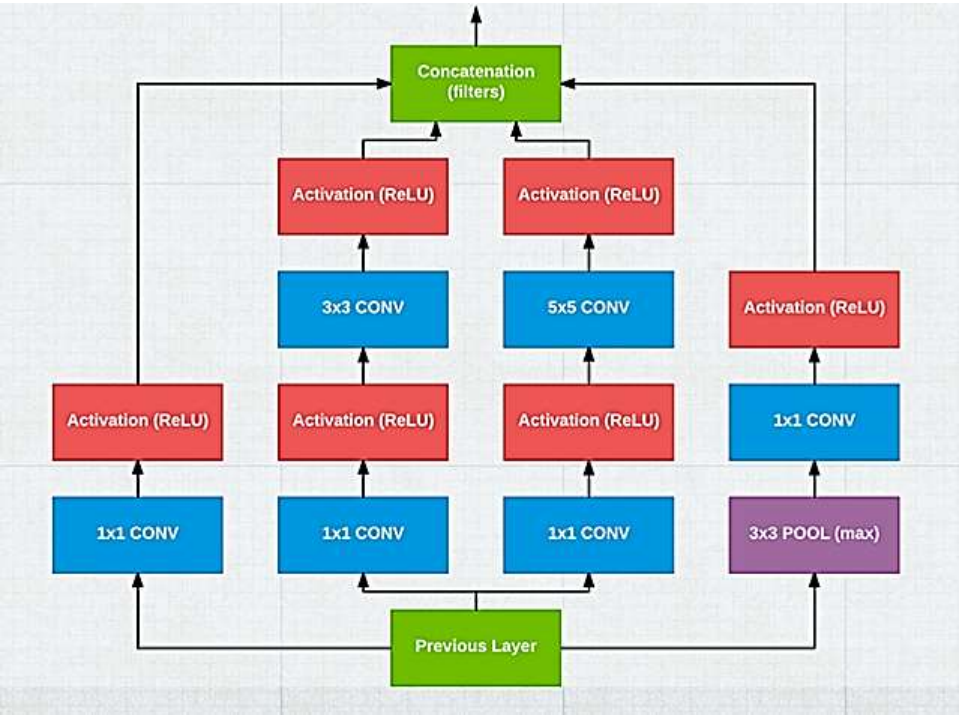
d. VGG *Network*



Gambar 2.9. Arsitektur VGG16 (Liliana et al., 2017)

VGG *Network* pertama kali diperkenalkan oleh Simonyan dan Zisserman pada tahun 2014, menurutnya hanya dengan menumpuk 3 *layer* konvolusi dengan filter 3x3 dan *stride* 1 sama efektifnya dengan *layer* konvolusi dengan filter 7x7. Arsitektur VGG *Network* mendapat peringkat kedua pada ILSVRC'14 dengan menggunakan prosedur *training* sama seperti AlexNet namun tanpa menggunakan *Local Response Normalisation* (LRN) pada *layer* pertama dan juga menggunakan VGG16 atau VGG19 (gambar 2.9) dengan FC7 sebagai fitur untuk mengeneralisasi.

e. GoogLeNet

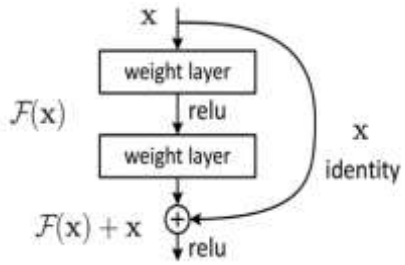


Gambar 2.10. *Building block GoogleNet* (Liliana et al., 2017)

Digunakan pertama kali oleh Szegedy pada tahun 2014, *GoogLeNet* memiliki 22 *layer*. *GoogLeNet* hanya memiliki 3 *full connected layer* dengan 5 juta parameter, dimana parameter tersebut 12 kali lebih kecil dibandingkan AlexNet. *Building block GoogLeNet* (gambar 2.10) sendiri menjadi pemenang pada ILSVRC'14 dengan nilai *error* sebesar 6,7%. *GoogLeNet* terdiri dari paralel filter, yaitu *multiple receptive field size* pada *layer* konvolusi yaitu 1x1, 3x3, 5x5 dan operasi *pooling* dengan *size* 3x3

f. Resnet

Resnet merupakan *deep network* yang sangat dalam karena menggunakan *residual connection* (gambar 2.11). Resnet pertama kali digunakan oleh He pada tahun 2015. Resnet memiliki 152 *layer* dan menjadi juara pada ILSVRC 2015 dengan nilai *error* sebesar 3,57%.



Gambar 2.11. *Residual connection in* ResNet (Liliana et al., 2017)

g. Densenet

Densenet pertamakali diusulkan oleh Huang pada tahun 2016, Huang mengusulkan sebuah arsitektur CNN dimana seluruh *layer* yang ada saling terhubung dengan *layer* lainnya secara *feed-forward* (Nguyen, Fookes, Ross, & Sridharan, 2018). Menurutnya arsitektur densenet memiliki beberapa keuntungan, yaitu mengatasi masalah *vanishing-gradient*, meningkatkan penyebaran fitur, mendorong penggunaan kembali fitur, dan secara substansial mengurangi jumlah parameter.

2.6. *Deep Learning Open-Source Frameworks*

Para peneliti *deep learning* tidak langsung memprogram *neural network* secara langsung, melainkan mereka menggunakan *libraries* perangkat lunak seperti *PyTorch* untuk mempermudah mereka. *Deep learning* merupakan metode yang relatif mudah, oleh karena itu *libraries* dan *tools* yang digunakan sangat cepat berubah. *Python* sejauh ini menjadi bahasa yang paling umum digunakan dalam *neural network*. Berikut *libraries* yang ada saat ini:

a. *Tensorflow*

Tensorflow merupakan salah satu *framework* yang sangat di apresiasi. *Tensorflow* dibuat oleh peneliti di *Google*. Sesuai yang tertulis pada *tensorflow.org*, *Tensorflow* merupakan *open source software library* untuk komputasi menggunakan data *flow graph*. *Tensorflow* termasuk kedalam *low level library*. *Tensorflow* bisa digunakan di *Python* dan C++, selama distribusi komputasinya antara CPU, GPU dan gRPC.

b. *Theano*

Theano merupakan *library* antarmuka dalam *Python* yang telah terintegrasi dengan *Numpy* dan bisa melakukan fungsi komputasi

gradient secara otomatis. Theano termasuk kedalam *low level library* dan tidak bisa digunakan untuk multi GPU atau *horizontal capabilities*.

c. *Caffe2*

Caffe2 dikembangkan oleh *Facebook* dan mengadaptasi *Caffe* dengan dukungan perangkat keras baru selain CPU dan CUDA. *Caffe2* dibangun untuk unggul dalam melakukan komputasi. Perbedaan *Caffe2* dengan *Pytorch* adalah bahwa *Pytorch* sangat bagus untuk penelitian, eksperimen dan percobaan *neural network*, sedangkan *Caffe2* lebih berorientasi untuk mendukung lebih banyak aplikasi industri dengan fokus pada ponsel.

d. *Keras*

Keras termasuk kedalam *high level library*. Berdasarkan *keras.io*, *Keras* mampu bekerja lebih baik dari *Tensorflow*, *Microsoft Cognitive Toolkit*, *Theano* atau MXNet. *Keras* bisa membangun sebuah *neural network* hanya dengan beberapa baris kode saja dan bisa membuat *interface* di *Python*.

e. *Lasagne*

Lasagne adalah *library* yang dibangun dan dikembangkan untuk melatih *neural network* di *Theano*. Tujuan utama *Lasagne* adalah untuk memudahkan komputasi *deep learning* dan membuat *friendly interface* pada *Python*, namun *Lasagne* tidak lebih baik dari *Keras*.

f. *Dsstne*

Dsstne adalah *framework* yang dikembangkan oleh *Amazon*. *Framework* ini bukan untuk kepentingan penelitian melainkan hanya untuk produksi saja dan *framework* ini tidak popular.

g. *Torch*

Torch merupakan *framework* yang cukup terkenal karena *framework* ini digunakan di *Facebook Research* dan di *DeepMind* sebelum diakusisi oleh *Google*. *Torch* mampu membuat grafik sembarang *neural network* dan memparalelkannya dengan CPU dan GPU secara efisien, namun *Torch* memiliki kekurangan karena menggunakan bahasa pemrograman *Lua* sedangkan sebagian besar *deep learning* menggunakan *Python*.

h. *PyTorch*

PyTorch adalan *open source machine learning* untuk *Python* yang lebih baik dari *Torch*. *PyTorch* di *realese* pada Oktober 2016 yang merupakan hasil pengembangan dari tim *Facebook's* dan *Uber*. *PyTorch* memiliki akselerasi GPU yang tinggi dan menyediakan 2 fitur dengan *high level*, yaitu *tensor computation* seperti *Numpy* dengan akselerasi GPU yang tinggi dan *deep neural networks* yang dibangun dengan *tape-based autograph system.*

i. *Microsoft Cognitive Toolkit*

Microsoft Cognitive Toolkit dahulu dikenal dengan CNTK merupakan *framework* hasil pengembangan dari *Microsoft Research*. *Framework* ini menggambarkan *neural network* sebagai serangkaian langkah komputasi dengan menggunakan grafik yang ditulis kedalam bahasa C++.

j. *PaddlePaddle*

PaddlePaddle merupakan *framework* yang dikembangkan oleh *Baidu*. *Framework* ini digunakan untuk *computer vision, natural language understanding* dan *deep learning*.

2.7. Image types

Citra merupakan representasi atau imitasi dari suatu objek. Citra dapat bersifat optik seperti foto, analog seperti sinyal atau bersifat digital yang dapat langsung disimpan pada media penyimpanan. Citra biasanya terdiri dari intensitas warna sebagai fungsi spatial x dan y, maka dari itu citra terdiri dari angka dan dapat di proses secara digital (Utari, 2016).

(a) Binary image *(b) grayscale image* *(c) True color image*

Gambar 2.12. Tipe citra berdasarkan kompsisi warna (Kumar & Verma, 2010)

Berdasarkan komposisi warnanya citra terbagi kedalam 3 tipe (Kumar & Verma, 2010) seperti yang ditunjukkan pada gambar 2.12, yaitu:

a. *Binary image*

   Seperti namanya *binary image* merupakan citra yang didalam *array*nya hanya terdiri dari 0 dan 1. 0 dan 1 ini menunjukkan komposisi warna yang dimiliki oleh citra tipe ini adalah hitam dan putih.

b. *Grayscale image*

   Grayscale image bisa disebut juga *gray level image*, yang terdiri dari kelas *uint8, uint16, int16, single,* dan *double* tergantung pada intensitasnya. Untuk *single* atau *double* nilai *array* terdiri dari *range* 0 hingga 1. Sedangkan untuk *uint8* terdiri dari 0 hingga 255, *uint16* terdiri dari 0 hingga 65535, *int16* terdiri dari -32768 hingga 32767. *Gray level* menunjukkan *interval* dari derajat keabuan. Citra pada umumnya menggunaka metode penyimpanan 8-*bit* yang artinya terdapat 256 *gray levels* dimana setiap *pixel* memiliki nilai dari 0 hingga 255. Nilai 0 menunjukkan warna hitam sedangkan 255 menunjukkan warna putih. Namun untuk perbaikan kualitas gambar atau *edge detection* biasanya menggunakan metode penyimpana 1-*bit*, yang artinya hanya ada 2 *gray levels* saja. Maka dari itu citra biasanya di konversi kedalam *grayscale* untuk meminimalkan kebutuhan memori dalam merepresentasikan citra.

c. *True color image*

Sedangkan pada *true color image* atau biasa disebut RGB *color model*, citra terdiri dari 3 komponen yaitu merah, hijau dan biru. Merah memiliki warna minimum putih dan warna maksimum merah. Hijau memiliki warna minimum putih dan warna maksimum hijau. Sedangkan biru memiliki warna minimum putih dan warna maksimum biru. Komponen warna tersebut biasa disebut *channels* atau *planes.* Intensitas dari setiap *channels* biasanya menggunakan 8-*bit* sehingga memiliki 256 *levels*.

2.8. Image enhancement methods

Citra yang bagus dapat menggambarkan ribuan kata, oleh karena itu *image enhancement* telah menjadi bidang yang banyak diminati. Pengolahan citra merupakan pemrosesan citra menggunakan komputer menjadi citra dengan kualitas yang baik. Pada umumnya operasi pada pengolahan citra dilakukan dengan cara meningkatkan kualitas citra untuk menampilkan beberapa informasi yang terkandung pada citra, mengelompokkan elemen dalam citra, atau penggabungan citra dengan citra lainnya (Roni & Adi, 2000).

Image enhancement merupakan sebuah cara untuk meningkatkan interpretabilitas atau persepsi informasi yang diberikan dalam suatu gambar dan untuk mempersiapkan gambar menjadi *input* yang baik untuk *image processing* selanjutnya. *Image enhancement* memiliki tugas utama yaitu mengubah atribut pada gambar untuk membuatnya lebih cocok dalam melakukan tugas yang diberikan (Thakur & Mishra, 2015).

Tidak ada teori umum dalam *image enhancement* karena ketika suatu citra diproses untuk interpretasi visual maka yang menilai seberapa baik metode *image enhancement* yang digunaan adalah seorang *viewer*. Sehingga evaluasi kualitas gambar adalah proses yang sangan subjektif dan membuat kriteria penilaian citra yang baik tidak bisa dijadikan parameter dalam menentukan kinerja dari suatu algoritma *image enhancement* (Vandra & Kulkarni, 2012).

Ada banyak teknik dalam *image enhancement* yang dapat meningkatkan kualitas gambar tanpa merusaknya. Metode *image enhancement* terbagi menjadi

dua kategori, yaitu s*patial domain methods* dan *frequency domain methods. Spatial domain* merupakan metode *image enhancement* yang menggunakan pendekatan dengan cara memanipulasi secara langsung setiap piksel dalam suatu citra hingga mendapatkan nilai piksel yang diinginkan. Sedangkan pada *frequency domain methods* gambar terlebih dahulu di transformasi ke *frequency domain*, maka pada metode ini *fourier transform* dari citra harus dihitung terlebih dahulu (Maini & Aggarwal, 2010).

Operasi dalam *image enhancement* adalah mengubah *image brightness*, *contrast* atau distribusi dari *gret level. Image enhancement* adalah mentransformasikan gambar f kedalam gambar *g* menggunakan *T*, dimana *T* merupakan *transformation.* Hasil dari transformasi dari gambar *f* dan *g* dinotasikan menjadi *r* dan *s* seperti yang digambarkan pada persamaan 1.

$$s = T(r) \dots (1)$$

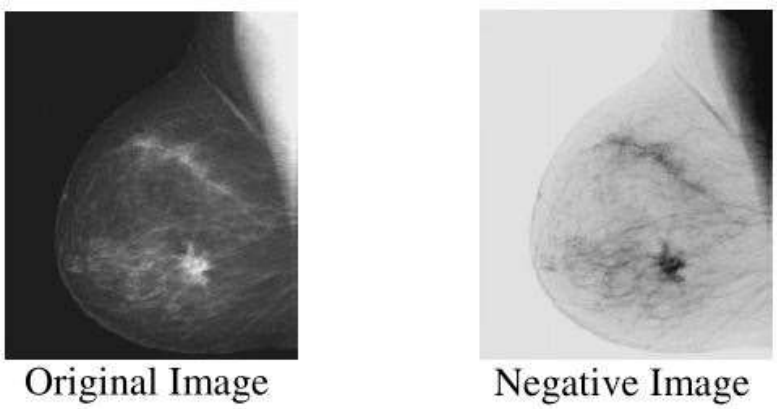
Berikut teknik-teknik yang digunakan dalam *image enhancement*:

a. *Point processing operation*

   Operasi *spatial domain* sederhana akan terjadi jika lingkungan citra tersebut hanya piksel itu sendiri. Pada kasus ini T merupakan fungsi transformasi *gray level* atau *point processing operation*. Fungsi *point processing operation* ditunjukkan pada persamaan 1. Berikut tahapan yang digunakan dalam *point processing operation*:

   1. Membuat *negative image*

      Hal mendasar dalam pemrosesan gambar adalah mengubah gambar menjadi negatif yaitu dengan membalikkan nilai piksel *gray* gambar. Jika suatu gambar memiliki ukuran Rx C dimana R merupakan jumlah baris dan C merepresentasikan kolom, maka nilai negatif dari gambar tersebut adalah N(r, c). gambar 2.13 menunjukkan citra hasil mengubah gambar *gray* menjadi *negative*.

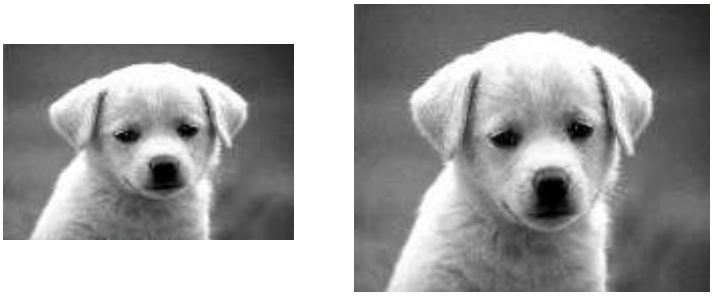


Gambar 2.13. Hasil proses *negative image* (Vandra & Kulkarni, 2012)

2. *Scalling*

$$w = w * x$$
$$h = h * x \qquad \ldots(2)$$

Scalling merupakan operasi geometri yang memberikan efek memperbesar atau memperkecil ukuran citra agar sesuai dengan variabel penskalaan citranya. Ukuran baru hasil *scalling* diperoleh dari perkalian ukuran citra input dengan variabel penskalaan. Jika variabel penskalaan lebih dari 1 maka akan memperbesar citra, namun jika variabel penskalaan lebih kecil dari 1 maka akan memperkecil citra. Misal w merupakan *width* dan h merupakan *heigt* suatu citra, sedangkan x merupakan variabel penskalaan. Operasi penskalaan ditunjukkan pada persamaan 2, jika variabel penskalaan (x) adalah 2 maka citra akan diperbesar menjadi 2 kali (Putu, Made, & Dessy, 2013). Gambar 2.14 menunjukkan citra hasil operasi *scalling.*

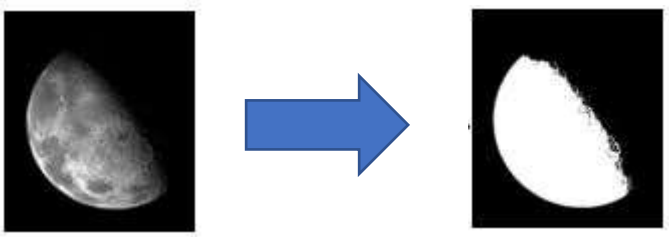


(a) Citra asli (b) Citra hasil *scalling*

Gambar 2.14. Operasi *scalling*

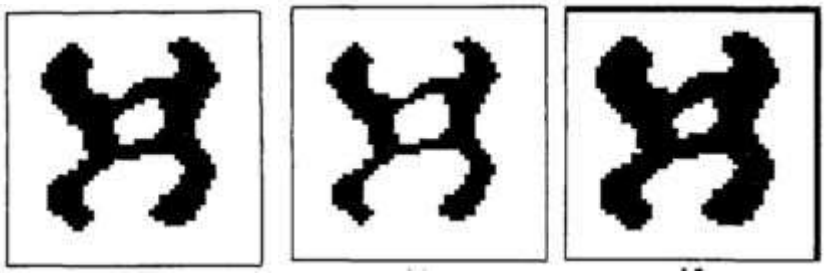
3. *Segmentation*

Pada proses segmentasi ada beberapa cara yang bisa dilakukan yaitu *tresholding, dilated, eroded,* atau *background subtraction.* Proses *tresholding* sangat berguna untuk *image enhancement*, karena pada tahap ini akan dilakukan segmentasi untuk memisahkan objek dengan *background*. Pada proses ini citra diubah kedalam biner sehingga hanya memiliki 2 buah nilai yaitu 0 untuk hitam dan 1 untuk putih. Gambar 2.15 meunjukkan citra hasil proses *tresholding*.



Gambar 2.15. Hasil proses *tresholding* (Maini & Aggarwal, 2010)

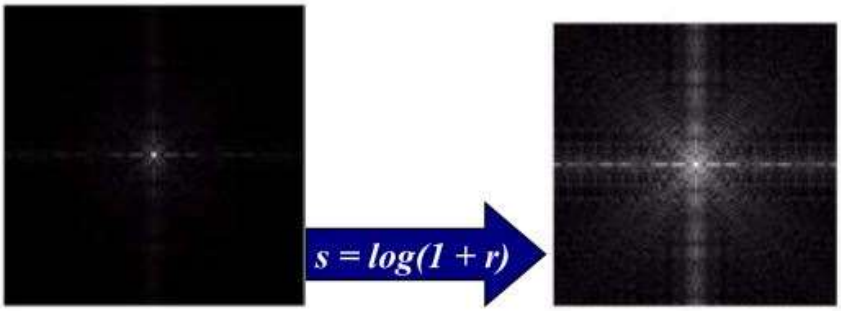
Tresholding bertujuan untuk mengubah citra dari abu-abu menjadi citra biner sehingga dapat diketahui daerah mana yang termasuk obyek dan *background* dari citra secara jelas (Goodman & Rhodes, 2009). Selain *tresholding* cara lain untuk melakukan segmentasi adalah *background subtraction*. *Background subtraction* sama dengan *tresholding* yaitu merupakan salah satu teknik pada bidang pengolahan citra dan *computer vision* yang bertujuan untuk mendeteksi atau mengambil *foreground* dari *background* untuk diproses lebih lanjut (Xie et al., 2016). Proses dilasi bertujuan untuk memperbesar segmen objek dengan menambah lapisan disekeliling objek, sedangkan proses erosi bertujuan untuk memperkecil segmen objek atau mengikis tepi objek (Tariq, Jamal, Shahid, & Malik, 2004). Gambar 2.16 menunjukkan citra hasil proses dilasi dan erosi.



Gambar 2.16. Citra hasil proses dilasi dan erosi (Goodman & Rhodes, 2009)

4. *Logarithmic transformations*

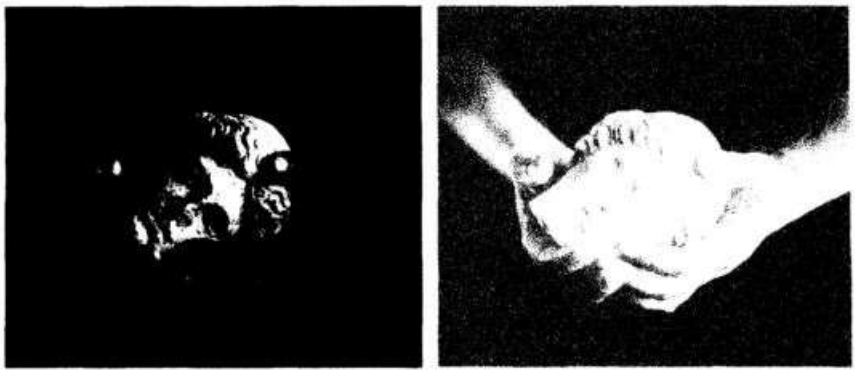
   Logarithmic transformations adalah memetakan *low input gray level* menjadi nilai *output* yang lebih luas. *Inverse* dari *logarithmic* adalah *opposite transformation*. *Logarithmic* sangat berguna ketika *gray scale* memiliki rentang nilai yang sangat besar. Gambar 2.17 menunjukkan citra hasil proses *logartihmic transformations*.



Gambar 2.17. Citra hasil proses *logarithmic* (Maini & Aggarwal, 2010)

b. *Histogram equalization*

   Histogram menggambarkan *statistic probabilistic distribution* dari setiap *gray level* pada citra. Dalam sebuah *histogram* dapat memberikan informasi berupa *gray scale, gray level, density, average luminance,* dan *contrast. Histogram equalization* merupakan metode yang sederhana dan efektif untuk *image enhancement*. Tujuan utama dari *histogram equalization* adalah untuk mengurangi jumlah derajat keabuan citra sehingga kontras gambar dapat ditingkatkan (Yao, Lai, & Wang, 2017). Gambar 2.18 menunjukkan citra hasil proses *histogram equalization*.



Original image *Histogram equalization processing image*

Gambar 2.18. Citra hasil proses *histogram equalization* (Yao et al., 2017)

c. *Fuzzy theory*

Fuzzy image processing application didasarkan pada binarisasi gambar dan peningkatan langsung *gray scale*. Pendekatan *gray scale images enhancement* terdiri dari beberapa langkah yaitu *normalization, local orientation estimation, local frrequency estimation, filtering by designed filters*. Untuk menggunakan pendekatan *fuzzy* citra harus direpresentasikan kedalam bilangan real yang memiliki rentang [0, 1]. Tahap ini biasa disebut dengan normalisasi, dimana rentang *gray scale* akan berkurang. Semua *fuzzy enhancement* dilakukan pada gambar yang ternormalisasi (Thakur & Mishra, 2015). Gambar 2.19 menunjukkan citra hasil proses *fuzzy enhancement*.

Original image *Fuzzy enhaced*

Gambar 2.19. Citra hasil *fuzzy enhaced* (Thakur & Mishra, 2015)

2.9. *Vehicle detection system*

Kemacetan sudah menjadi salah satu permasalahan yang sangat serius. Solusi mudah yang sering ditawarkan adalah dengan menambah jumlah jalan untuk mengurai kemaceten, namun dengan memperluas jalan tidak menyelesaikan kemacetan tapi malah memunculkan masalah baru. Terus meningkatnya angka kemacetan melahirkan sebuah minat baru dalam bidang deteksi kendaraan seperti pemrosesan gambar video. Menurut (Coifman, Beymer, McLauchlan, & Malik, 1998) untuk membuat sebuah sistem pemrosesan gambar video harus memenuhi kriteria sebagai berikut:

1. Sistem mampu melakukan segmentasi dari *background* secara otomatis pada tiap kendaraan yang terdeteksi.
2. Sistem mampu mendeteksi semua objek yang ada. Maksudnya sistem harus mampu membedakan setiap objek yang ada seperti motor, mobil atau pejalan kaki.
3. Sistem mampu bekerja dalam semua kondisi jalan, seperti kemacetan dan perbedaan kecepatan.
4. Sistem mampu bekerja dalam kualitas cahaya yang rendah maupun yang terlalu terang.
5. Sistem mampu beroperasi secara *real-time*

Sedangkan sistem deteksi objek sendiri pada umumnya terdiri dari ekstraksi *frame*, *croping*, *labelling, resize* atau *scalling*, dan klasifikasi. Ekstraksi *frame* dan *croping* merupakan proses untuk pengambilan citra pada setiap *frame* sehingga menghasilkan dataset yang dibutuhkan. Selanjutnya dilakukan proses pelabelan yaitu pembagian dataset berdasarkan kelasnya untuk membedakan objek yang satu dengan objek lainnya. Sebelum melakukan klasifikasi dataset yang memiliki ukuran berbeda diubah sehingga memiliki ukuran yang sama (Limantoro et al., 2017).

Pemantauan lalu lintas menggunakan kamera CCTV membantu petugas lalu lintas dalam memantau dan mengontrol jalan raya di berbagai tempat sekaligus. Sistem ini sangat berguna bila di terapkan di pusat kota, dimana kemacetan merupakan makanan sehari-hari sehingga petugas bisa memantau banyak titik dengan mudah. Ada banyak sistem yang telah dibuat untuk memantau kondisi lalu

lintas dengan kamera CCTV namun sistem tersebut menggunakan kamera dengan resolusi tinggi dan memiliki akurasi yang rendah. (Bautista et al., 2016) mencoba membuat sistem serupa yang mampu bekerja pada video dengan resolusi rendah. Percobaan pertama menggunakan pendekatan dengan gabungan *Support Vector Machine* (SVM) dan *Artificial Neural Network* (ANN) menghasilkan akurasi sebesar 95,7% pada data siang hari dan 88,8% pada data malam hari. Sedangkan saat menggunakan *Convolutional Neural Network* (CNN) menghasilkan akurasi yang sangat tinggi yaitu 94,72% pada data malam hari.

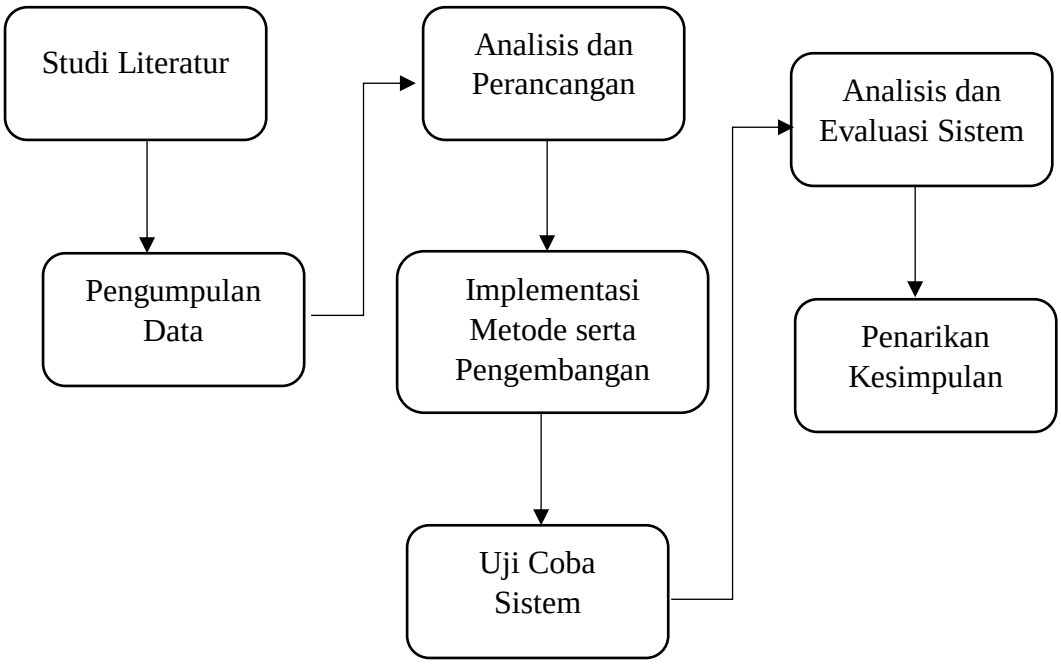
Klasifikasi menggunakan *Convolutional Neural Network* mampu melakukan klasifikasi pada fitur yang kompleks seperti pengendara sepeda motor dan bukan pengendara sepeda motor. Menurut (Limantoro et al., 2017) nilai akurasi validasi CNN dipengaruhi oleh arsitekturnya. Pada penelitiannya ia menggunakan 3 jenis arsitektur yaitu besar, sedang dan kecil menghasilkan akurasi yang berbeda yaitu 96,25%, 95,6% dan 95,6%. Arsitektur kecil terdiri pada lapisan konvolusi dengan *kernel* 5x5 diikuti ReLu dengan filter 3x3. Lapisan kedua merupakan *max pooling* dan lapisan konvolusi yang diikuti ReLu. Pada lapisan ke tiga sampai lima merupakan *full connected*. Pada arsitektur sedang ditambah satu *layer* konvolusi sebelum *full connected*. Sedangkan pada arsitektur besar hampir sama dengan arsitektur sedang namun ada perbedaan parameter pada lapisan *full connected*.

Sedangkan (Dong et al., 2014) memperkenalkan cara baru dalam mendeteksi kendaraan yaitu menggunakan metode *semi-supervised Convolutional Neural Network* (CNN). Zhen Dong menggunakan *Laplacian filter learning* (SLFL) untuk mendapatkan filter dalam jaringan pada data yang tidak berlabel. SLFL merupakan *output* dari *classifier softmax* yang telah dilatih *dengan multi-task learning* pada data berlabel. SLFL akan memberikan probabilitas dari setiap jenis kendaraan yang dimiliki setiap kita memberikan gambar kendaraan tertentu. Berbeda dengan metode CNN yang konvensional, metode SLFL mampu mempelajari fitur untuk klasifikasi secara otomatis. Fitur yang dipelajari cukup diskriminatif untuk bekerja dengan baik. Metode ini menghasilkan akurasi sebesar 95,7% pada data siang hari dan 88,8% pada malam hari dengan *BIT-Vehicle Dataset.*

BAB III
METODOLOGI PENELITIAN

1.1. Desain Penelitian

Desain penelitian digambarkan pada gambar 3.1 yang terdiri dari studi literatur, pengumpulan data, analisis dan perancangan, implementasi metode serta pengembangan, uji coba sistem, analisis dan evaluasi sistem serta penarikan kesimpulan.



Gambar 3.1. Desain Penelitian

a. Studi literatur

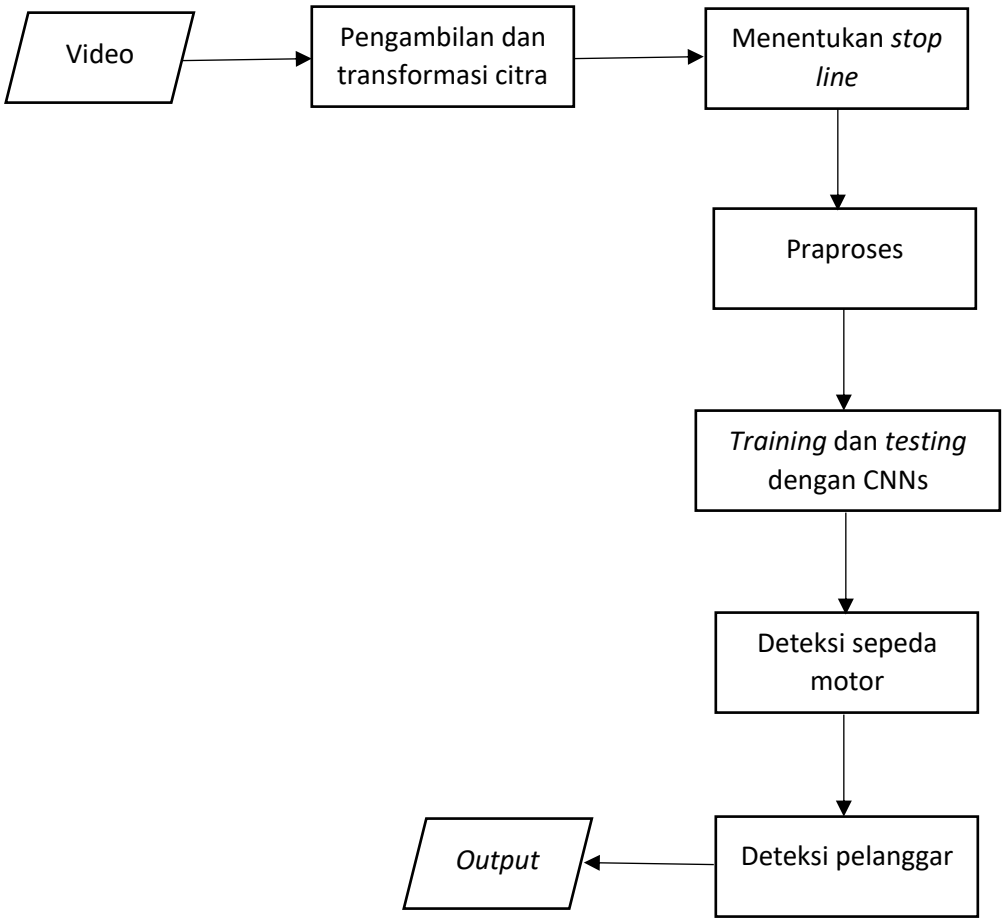
Tahap awal pada penelitian ini adalah studi literatur. Studi literatur dilakukan dengan mencari sumber terkait dengan penelitian yang dilakukan, diantaranya yaitu mengenai hakikat lalu lintas, jenis-jenis marka jalan, tata tertib lalu lintas, perkembangan *machine learning* dalam mengatasi permasalahan sehari-hari, klasifikasi *deep learning*, *Convolutional Neural Networks* (CNNs), arsitektur CNNs, *open source frameworks* yang digunakan untuk *deep learning*, serta penelitian terdahulu mengenai *vehicle detection system*.

Setelah melakukan studi literatur maka diperoleh suatu langkah dalam menyelesaikan sistem ini. Penjelasan mengenai teori-teori tersebut terdapat pada BAB II.

b. Pengumpulan data

Pada tahap ini dilakukan pengumpulan data-data, alat, dan bahan yang diperlukan dalam penelitian. Hal-hal tersebut didapatkan melalui observasi. Observasi yang dilakukan adalah dengan terjun langsung ke lapangan untuk mendapatkan data penelitian melalui survei ke kantor Dinas Perhubungan Kota Bandung pada bagian *Area Traffic Control System* (ATCS) yang berada di wilayah Balai Kota jalan Wastukencana No. 2 Bandung. Observasi dilakukan untuk memperoleh data berupa video rekaman CCTV serta mendapatkan informasi mengenai proses pengontrolan lalu lintas melalui CCTV yang biasa dilakukan sehari-hari.

c. Analisis dan perancangan



Gambar 3.2. Alur sistem

Langkah analisis dan perancangan adalah langkah untuk menganaliss kebutuhan sistem. Berikut rancangan alur sistem terdapat pada gambar 3.2.

1. Pengambilan dan transformasi citra

   Data *input* yang digunakan berupa video dengan format .mp4. untuk melakukan klasifikasi dibutuhkan data berupa citra, maka dari itu video perlu diubah menjadi bentuk citra agar bisa diproses ke tahap selanjutnya. Pada tahap ini proses pengambilan dan transformasi citra dilakukan secara manual dengan melakukan seleksi pada objek objek tertentu sesuai kebutuhan seperti sepeda motor, pejalan kaki, sepeda dan lainnya.

2. Menentukan *stop line*

   Sistem yang dibuat harus mampu mendeteksi sepeda motor yang melanggar marka jalan. Pelanggaran marka jalan yang akan dideteksi sistem merupakan sepeda motor yang berhenti melebihi *stop line* atau garis pemberhentian. Agar sistem mampu mendeteksi pelanggar maka sistem harus menentukan terlebih dahulu *stop line* pada video *input*. Penentuan *stop line* kendaraan dilakukan secara manual dengan cara menentukan titik kordinat x_1, y_1 dan x_2, y_2. Hal ini dilakukan agar mempermudah proses pendeteksian.

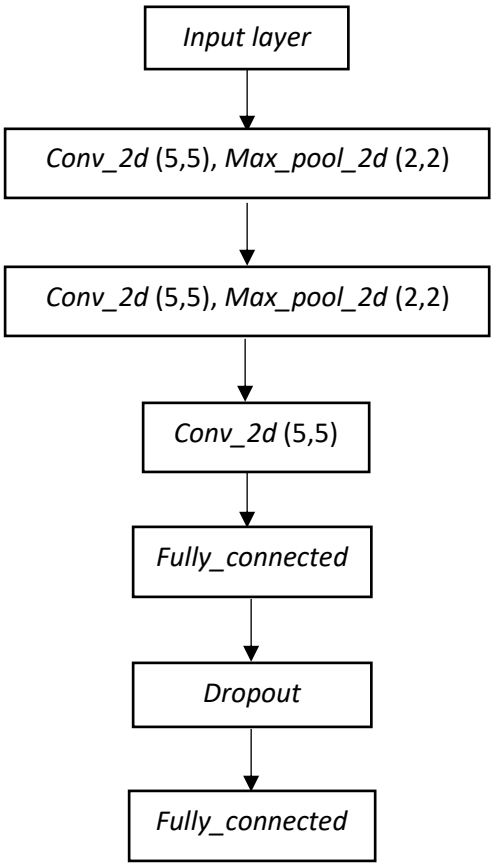
3. Praproses

   Citra yang diperoleh setelah tahap pengambilan dan transformasi citra harus melakukan praproses terlebih dahulu. Praproses data digunakan untuk memperbaiki kualitas citra sehingga objek pada citra terlihat lebih jelas serta memudahkan proses klasifikasi. Pada penelitian yang dilakukan (Putra, 2016) pra proses data terdiri dari proses *scalling* dan konversi citra.

4. *Training* dan *testing* dengan CNNs

   Pada tahap ini dilakukan *training* dan *testing* menggunakan CNNs. Hasil dari tahap praproses berupa citra yang siap untuk di *training*. Setelah melakukan *training* akan memperoleh ciri citra yang digunakan untuk melakukan klasifikasi citra. *Testing* dilakukan untuk mengetahui seberapa baik model CNNs yang dimiliki untuk

klasifikasi. Metode CNNs yang digunakan menggunakan 5 *layer* seperti yang terdapat pada gambar 3.3 yaitu 3 *layer* konvolusi yang diikuti oleh *layer subsampling* dan 2 *layer fully connected* dengan 1000 *epoch*. Pada *layer* konvolusi menggunakan skala 5x5, sedangkan pada *layer subsampling* menggunakan skala 2x2. Arsitektur ini berhasil digunakan oleh Lecun dengan menggunakan dataset MNIST yang memiliki dimensi data citra 28 x 28 piksel dengan akurasi sebesar 99,1% (Lecun, Bottou, Bengio, & Ha, 1998).

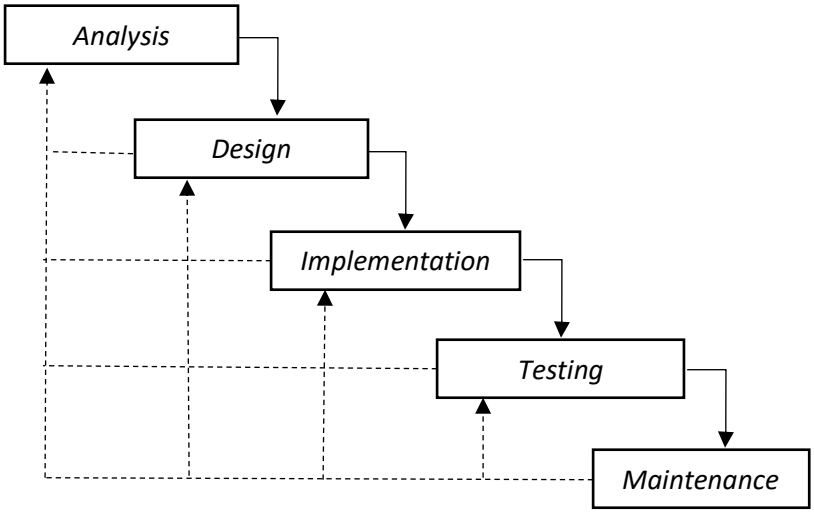


Gambar 3.3. Arsitektur CNNs

d. Implementasi metode serta pengembangan

Langkah selanjutnya adalah mengimplementasikan metode *Convolutional Neural Networks* (CNNs) untuk membuat sistem pendeteksi sepeda motor pelanggar marka jalan. Implementasi dilakukan dengan mengimplementasikan hasil desain ke dalam perangkat lunak dengan

menerjemahkannya kedalam bahasa yang dapat dibaca oleh mesin komputer. Pengembangan aplikasi menggunakan metode *waterfall*. Model *waterfall* merupakan model pengembangan yang bersifat *sequential* dimana tahapan selanjutnya tidak bisa dikerjakan sebelum tahapan sebelumnya selesai (Balaji, 2012). *Waterfall* model digambarkan pada gambar 3.4.



Gambar 3.4. *Waterfall* model (Bassil, 2012)

Berikut tahap-tahap yang dilakukan dalam pengembangan sistem:

1. Analisis kebutuhan

   Analisis kebutuhan merupakan tahapan awal dalam pengembangan perangkat lunak. Pada tahap ini dilakukan pengumpulan data informasi kebutuhan untuk memahami gambaran umum perangkat lunak yang akan dibangun. Pada tahap ini pula ditentukan spesifikasi dari perangkat lunak untuk mencapai tujuan dari dibuatnya perangkat lunak.

2. Desain sistem

   Pada tahap ini perangkat lunak didesain agar dapat berjalan sesuai dengan hasil analisis yang telah dilakukan sebelumnya, diantaranya mendesain arsitektur perangkat lunak serta representasi antarmuka.

3. Implementasi

   Pada tahap ini penulis mengimplementasikan hasil desain ke dalam perangkat lunak dengan diterjemahkan ke bahasa yang dapat dibaca oleh mesin komputer. Pada penelitian ini kode program dibuat

menggunakan bahasa *Python* dengan menggunakan *frameworks Tensorflow*. Pada saat implementasi metode, data video yang diperoleh dibagi menjadi dua kelompok yaitu kelompok data *training* dan data *testing*. Data *training* merupakan data yang digunakan untuk mempelajari pada saat proses klasifikasi. Sedangkan data *testing* digunakan untuk melihat apakah sistem yang dibuat sudah mampu melakukan klasifikasi seperti yang diinginkan.

4. Pengujian sistem

   Pada tahap ini dilakukan proses pengujian setelah kode program selesai dibuat. Hal ini dilakukan untuk menemukan kesalahan-kesalahan pada perangkat lunak agar memastikan perangkat lunak tersebut telah berfungsi sesuai dengan yang diharapkan.

5. Pemeliharaan

   Tahap ini merupakan tahap akhir pada pengembangan perangkat lunak dimana perangkat lunak mengalami perubahan. Perubahan terjadi karena perangkat lunak mengalami kesalahan atau untuk menyesuaikan kembali dengan kebutuhan.

e. Uji coba sistem

   Pada tahap ini dilakukan pengujian terhadap sistem yang telah dibangun untuk mengetahui tingkat akurasi serta kesesuaian antara sistem yang dibuat dengan tujuan penelitian. Tahap pengujian dilakukan dengan membuat beberapa skenario pengujian yang masing-masing skenarionya memiliki parameter berbeda. Hasil dari pengujian kemudian dianalisis di tahap selanjutnya.

f. Analisis dan evaluasi sistem

   Setelah melakukan uji coba sistem, akan menghasilkan hasil percobaan pada tiap skenario pengujiannya. Hasil pengujian yang telah diperoleh akan dianalisi dan dievaluasi pada tahap ini sehingga dapat diketahui sistem yang telah dibuat sesuai atau tidak dengan tujuan yang telah dirumuskan sebelumnya.

g. Penarikan kesimpulan

Tahap terakhir yaitu penarikan kesimpulan, dimana di tahap ini hasil dari analisis sistem yang telah dilakukan uji coba akan dibuat kesimpulannya berdasarkan rumusan masalah yang telah dibuat sebelumnya. Kesimpulan ini bertujuan untuk melihat kinerja sistem yang telah dibuat sudah menjawab rumusan masalah atau belum. Kesimpulan dapat dilihat lebih lanjut pada BAB V.

1.2. Perangkat Keras dan Perangkat Lunak

Dalam membangun sistem ini dibutuhkan perangkat keras berupa komputer dan perangkat lunak. Perangkat keras yang digunakan dalam penelitian ini memiliki spesifikasi sebagai berikut:

a. *Prosesor* AMD A8
b. RAM 8 GB
c. *Mouse* dan *keyboard*
d. *Flashdisk* 64 MB

Adapun perangkat lunak yang digunakan untuk menunjang pembuatan sistem yaitu:

a. Sistem operasi *Microsoft Windows* 10 64 *bit*
b. *Python* 3.6.5
c. *Frameworks Tensorflow*
d. *Opencv library*
e. *Snipping tool*

1.3. Data Penelitian

1.3.1. Data *input*

Data *input* yang digunakan dalam penelitian ini merupakan video CCTV di persimpangan jalan Cibaduyut. Video ini diperoleh dari bagian *Area Traffic Control System* (ATCS) Dinas Perhubungan Kota Bandung. Video rekaman yang diperoleh memiliki format .mp4 sehingga perlu diubah terlebih dahulu menjadi sebuah citra dengan format .jpg.

1.3.2. Data *output*

Output dari sistem pendeteksi sepeda motor pelanggar marka jalan ini adalah kelas data. Kelas data ini berupa hasil klasifikasi sistem dari label pada data

training. Data *output* yang dihasilkan berupa video yang telah mampu melakukan pendeteksian terhadap sepeda motor yang melanggar marka jalan. Jika terdapat sepeda motor yang melanggar sistem akan mendeteksi dengan cara memberi sebuah kotak serta label pelanggar pada kendaraan tersebut.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

1.1. Pengumpulan data

Pengumpulan data dilakukan dengan observasi secara langsung ke kantor Dinas Perhubungan Kota Bandung yang berada di jalan Wastukencana no. 2 Bandung. CCTV yang terdapat pada persimpangan jalan dikelola oleh bagian ATCS (*Area Traffic Control System*). Terdapat banyak persimpangan jalan yang sudah terpasang kamera CCTV dengan kualitas yang berbeda-beda. Persimpangan dengan kualitas CCTV yang paling baik terdapat pada persimpangan jalan Cibaduyut yaitu dengan resolusi 1280 x 720 *pixel*.

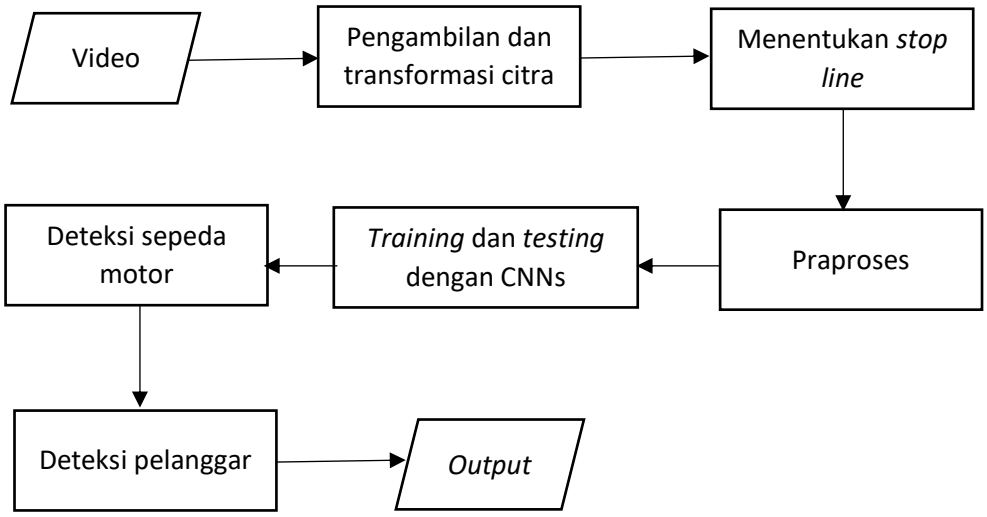
Pihak ATCS tidak dapat menyimpan seluruh data rekaman CCTV karena keterbatasan *server* yang dimiliki. Mereka hanya menyimpan data rekaman selama 5 hari dari setiap persimpangan jalan kemudian data rekaman tersebut akan terhapus secara otomatis. Sehingga membutuhkan waktu yang cukup lama untuk mengumpulkan video rekaman pada hari yang berbeda.

Gambar 4.1. Contoh dari video rekaman CCTV

Video yang digunakan pada penelitian ini diambil mulai dari tanggal 6 hingga 16 Oktober 2018 untuk mendapatkan video yang bervariasi. Sehingga diperoleh 231 video yang diambil dengan durasi 10 menit pada setiap video dengan format .mp4. Video yang digunakan memiliki rentang waktu pengambilan pada pukul 07.00 hingga 17.00 agar mendapatkan video dengan kualitas cahaya yang

bagus. Gambar 4.1 merupakan contoh dari video rekaman CCTV yang diambil pada persimpangan jalan Cibaduyut. Pada gambar tersebut menunjukkan jarak CCTV dengan objek yang cukup jauh serta sudut pandang dari kamera.

1.2. Sistem pendeteksi sepeda motor pelanggar marka jalan



Gambar 4.2. Alur proses sistem pendeteksi sepeda motor pelanggar marka jalan

Pada penelitian ini, sistem pendeteksi pelanggar marka jalan dibangun dengan menggunakan metode *Convolutional Neural Networks*. Metode CNNs digunakan untuk melatih data pada proses pengenalan citra pelanggar. Gambar 4.2 menunjukkan alur proses dari sistem pendeteksi sepeda motor pelanggar marka jalan.

1.2.1. Pengambilan dan transformasi citra

Data *input* hasil rekaman CCTV berupa video dengan format .mp4 maka dari itu data *input* harus diubah menjadi data citra terlebih dahulu agar bisa di proses ke tahap selanjutnya. Proses pengambilan dan transformasi citra dilakukan secara manual dengan menggunakan *software snipping tool*. Video *input* diseleksi pada bagian motor saja dan pada objek bukan motor seperti sepeda, mobil dan pejalan kaki sehingga menghasilkan data citra dengan format .jpg. Pengambilan dan transformasi citra dari 231 video menghasilkan data citra sebanyak 3000 citra,

dengan jumlah citra motor sebanyak 1500 citra dan sebanyak 1500 citra bukan motor. Gambar 4.3 menunjukkan data citra motor, sedangkan gambar 4.4 menunjukkan citra bukan motor. Pada gambar terlihat kualitas dari data citra dimana kondisi pencahayaan dan sudut pandang pada setiap citra berbeda.

Gambar 4.3. Contoh data citra motor

Gambar 4.4. Contoh data citra non motor

1.2.2. Menentukan *stop line*

Sistem yang dibuat berfungsi untuk mendeteksi sepeda motor yang melanggar marka jalan dimana pelanggaran marka jalan yang dideteksi yaitu sepeda motor yang berhenti melebihi *stop line* atau garis henti. *Stop line* untuk sepeda motor berbeda dengan mobil. Pada sepeda motor *stop line* berada sebelum jembatan penyebrangan (*zebracross*) sedangkan pada mobil *stop line* berada sebelum ruang henti khusus sepeda motor. Sistem dapat mendeteksi pelanggar jika sistem mengetahui posisi *stop line*. Oleh karena itu sistem harus menentukan *stop line* pada video terlebih dahulu sebelum mendeteksi pelanggar.

Inisialisasi *stop line* dilakukan secara manual untuk mempermudah sistem yaitu dengan menambahkan garis tepat sebelum jembatan penyebrangan pada video. Sistem menggunakan bahasa pemrograman *python* sehingga untuk pembuatan garis bisa dilakukan dengan perintah *cv2.line*. Pembuatan garis membutuhkan 2 titik, yaitu titik awal dan titik akhir. Pada sistem ini titik awal

diinisialisasikan pada koordinat (630, 430), sedangkan titik akhir diinisialisasikan pada koordinat (900, 410). Gambar 4.5 menunjukkan hasil pembuatan *stop line* pada video rekaman CCTV. *Stop line* diinisialisasikan dengan warna merah agar *stop line* terlihat dengan jelas.

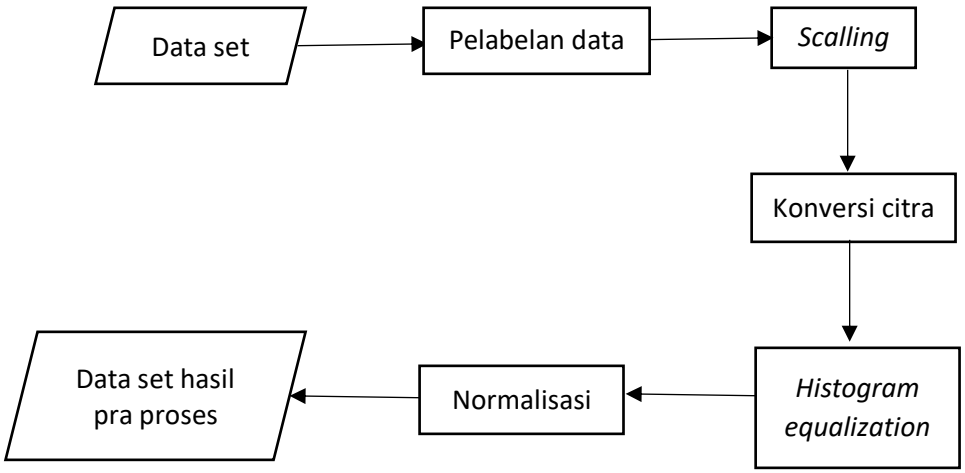
Gambar 4.5. Hasil penambahan *stop line*

1.2.3. Pra proses

Citra yang telah diperoleh pada tahap sebelumnya tidak dapat langsung digunakan untuk *training* dan *testing*. Citra tersebut harus diolah terlebih dahulu agar citra siap untuk diproses oleh sistem sehingga informasi yang dihasilkan dapat digunakan pada tahap selanjutnya. Proses pengolahan citra tersebut ada pada tahap pra proses.

Pra proses yang digunakan terdiri dari *scalling* dan konversi citra seperti yang disebutkan pada bab 3.1, namun untuk mendapatkan informasi yang lebih banyak maka tidak cukup hanya melakukan *scalling* dan konversi citra saja. Maka dari itu pada pra proses ditambah proses untuk memperbaiki kualitas citra atau biasa disebut *image enhancement*. Pada sistem ini *image enhancement* yang dilakukan adalah *histogram equalization* dan normalisasi.

Berikut merupakan alur pra proses yang digunakan pada sistem pendeteksi sepeda motor pelanggar marka jalan digambarkan pada gambar 4.6.



Gambar 4.6. Alur pra proses

1. Proses pelabelan data

Data set terdiri dari 2 kelas yaitu kelas motor dan kelas non motor. Setiap kelas memiliki label yang berbeda begitupun dengan setiap citra pada masing-masing kelas yang memiliki labelnya masing-masing.

Proses pelabelan bertujuan sebagai penanda masing-masing citra sehingga memudahkan sistem pada saat proses *training*. Label yang digunakan dikonversi kedalam bentuk vektor. Tabel 4.1 menunjukkan representasi kelas data dalam bentuk vektor.

Tabel 4.1. representasi kelas data

Kelas data	Representasi kelas (vektor)
Motor	10
Non motor	01

2. Proses *scalling*

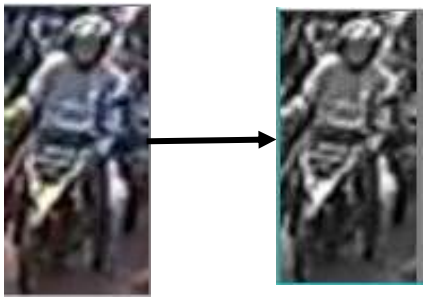
Data set yang ada sebelumnya memiliki dimensi yang berbeda-beda, oleh karena itu data perlu disamakan terlebih dahulu agar bisa dilakukan proses *training*. Proses untuk menyamakan dimensi citra ini dinamakan proses *scalling*. Pada proses ini setiap citra yang awalnya memiliki dimensi berbeda akan diubah menjadi 64x128 piksel. Pada *python* dengan menggunakan *library Opencv* perintah untuk mengubah dimensi bisa dilakukan dengan menggunakan perintah *cv2.resize*.

Gambar 4.7 menunjukkan citra hasil proses *scalling*. Pada gambar terlihat perubahan dimensi citra sebelum dan setelah proses *scalling*.

Gambar 4.7. Contoh hasil proses *scalling*

3. Proses konversi citra

Setelah citra memiliki dimensi yang sama, citra masih harus di proses kembali sebelum siap untuk di *training.* Tahap selanjutnya yaitu proses konversi citra. Data citra yang awalnya memiliki komposisi warna RGB diubah menjadi *grayscale*. Proses konversi citra bertujuan untuk menyederhanakan model citra. Hasil konversi citra ditunjukkan pada gambar 4.8. Pada gambar 27 terlihat perubahan warna citra dari yang awalnya RGB menjadi *grayscale*.

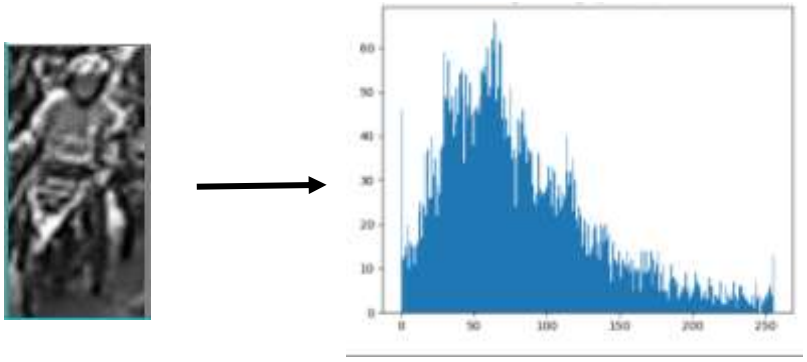


Gambar 4.8. Contoh hasil proses konversi citra ke *grayscale*

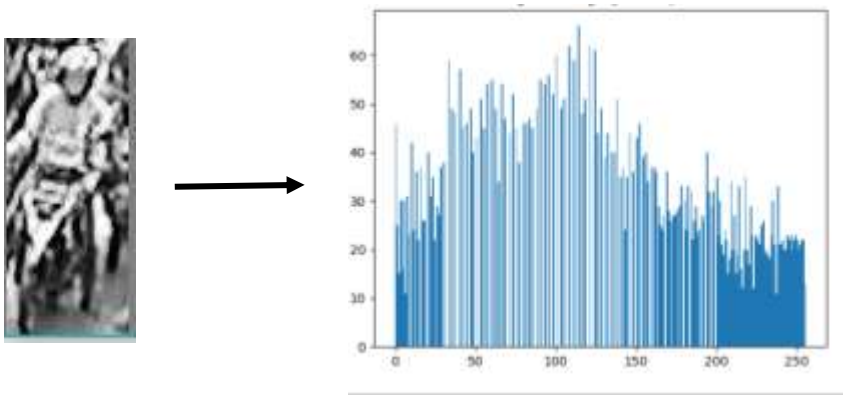
4. Proses *histogram equalization*

Setelah melakukan konversi citra menjadi *grayscale*, citra akan melakukan proses *histogram equalization*. Gambar 4.9 menunjukkan perbedaan citra setelah dan sebelum melakukan proses *histogram equalization*. Setelah melakukan proses *histogram equalization* citra

terlihat lebih terang dan lebih jelas. Sehingga gambar *histogram*nya tidak ada yang curam karena penyebaran warnanya lebih merata.



(a) Citra dan *histogram* sebelum proses *histogram equalization*



(b) Citra dan *histogram* setelah proses *histogram equalization*

Gambar 4.9. Gambar perubahan citra setelah proses *histogram equalization*

5. Proses normalisasi

$$data_{citra} = \frac{data_{citra}}{\max(data_{citra})} \dots (3)$$

Proses terakhir pada alur pra proses adalah proses normalisasi. Citra merupakan representasi nilai matriks seperti yang ditunjukkan pada gambar 4.10. Proses normalisasi dilakukan dengan membagi data citra dengan nilai terbesar dari data citra seperti yang dituliskan pada persamaan 3.

```
[[ 26.  30.  83. ... 228. 177. 114.]
 [ 53.  24.  42. ... 170. 108.  79.]
 [118.  40.  13. ...  60.  50.  79.]
 ...
 [166. 162. 154. ...  98.  81.  57.]
 [162. 156. 148. ... 108.  88.  77.]
 [156. 152. 148. ... 126. 114. 103.]]
```

Gambar 4.10. Representasi nilai matriks dari citra

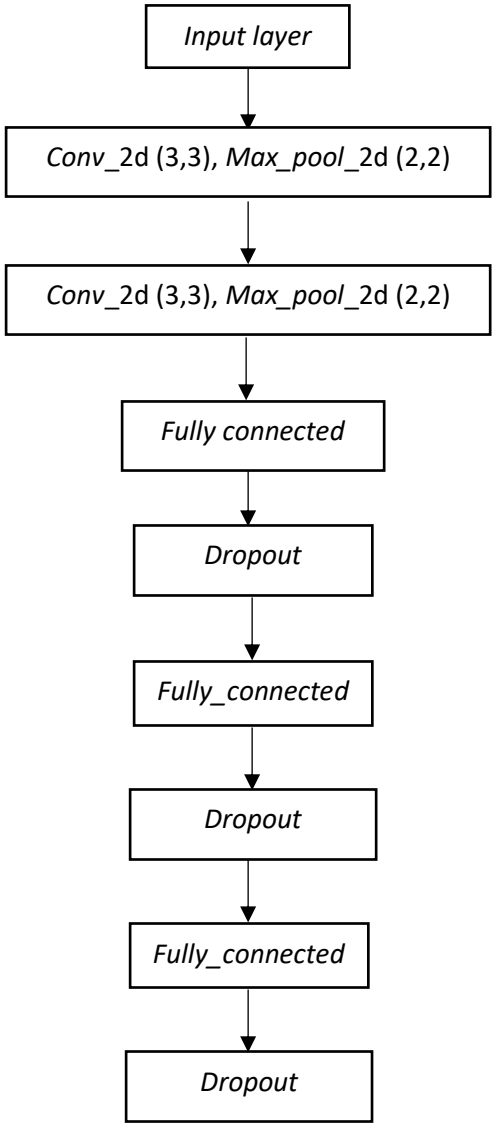
Pada proses ini nilai citra akan diubah menjadi antara 0 dan 1 dengan menggunakan persamaan 1 seperti yang ditunjukkan pada gambar 4.11. Pada gambar terlihat perbedaan nilai matriks sebelum dan sesudah proses normalisasi. Setelah proses normalisasi tidak ada nilai matriks yang melebihi 1.

```
[[-0.17647058  0.24313725 -0.0352941  ...  1.10588253  0.92549011
   0.2509805 ]
 [-0.05882356 -0.12156862 -0.15294119 ...  0.64705878 -0.38431372
  -0.23137254]
 [ 0.18431376 -0.22352943 -0.24313727 ... -0.15294123 -0.39607844
   0.13333334]
 ...
 [ 0.65098041  0.62352943  0.67450982 ...  0.32941183  0.54901963
   0.52549028]
 [ 0.67843139  0.58431375  0.69803923 ...  0.16470578  0.31372559
   0.19999996]
 [ 0.61176473  0.50196081  0.65882355 ...  0.42745104  0.43921575
   0.23529407]]
```

Gambar 4.11. Representasi nilai matriks hasil normalisasi

1.2.4. *Training* dan *testing dengan CNNs*

Pada tahap ini akan dilakukan dua proses yaitu *training* dan *testing*. Pada proses *training* dilakukan pengambilan ciri citra, sedangkan pada proses *testing* yaitu proses pengujian seberapa baik ciri citra yang telah dimiliki dari hasil *training* dalam melakukan klasifikasi. Proses pengambilan ciri citra ini menggunakan metode *Convolutional Neural Networks* (CNNs). Ada beberapa tahapan dalam menggunakan metode CNNs yaitu langkah pertama menentukan jumlah *layer* yang akan digunakan, selanjutnya menentukan ukuran *kernel* pada proses konvolusi. Langkah ketiga yaitu menentukan ukuran filter untuk proses *subsampling*.



Gambar 4.12. Arsitektur CNNs pada sistem pendeteksi pelanggar marka jalan

Sistem pendeteksi sepeda motor pelanggar marka jalan menggunakan arsitektur LeNet-5 dengan 5 *layer*. Saat sistem menggunakan arsitektur seperti pada gambar 3.3 sistem menghasilkan akurasi yang rendah yaitu hanya 67,5% saja. Oleh karena itu dilakukan proses *tuning* parameter untuk menentukan arsitektur CNNs yang terbaik untuk sistem ini. Proses *tuning* parameter dilakukan dengan mengubah arsitektur serta *kernel* yang digunakan. Hasil proses *tuning* parameter mendapatkan arsitektur CNNs seperti pada gambar 4.12. Arsitektur tersebut terdiri dari 2 *layer* konvolusi yang diikuti dengan *layer subsampling* dan 3 layer *fully connected*.

Kernel yang digunakan pada *layer* konvolusi adalah 3x3 dan menggunakan filter 2x2 pada *layer subsampling*. Proses *training* dilakukan sebanyak 1000 *epoch*.

1. Proses konvolusi

```
[[0.10196079 0.11764707 0.3254902  ... 0.8941177  0.69411767 0.44705886]
 [0.20784315 0.09411766 0.16470589 ... 0.6666667  0.42352945 0.30980393]
 [0.46274513 0.15686275 0.0509804  ... 0.23529413 0.19607845 0.30980393]
 ...
 [0.6509804  0.63529414 0.6039216  ... 0.38431376 0.31764707 0.22352943]
 [0.63529414 0.6117647  0.5803922  ... 0.42352945 0.34509805 0.3019608 ]
 [0.6117647  0.59607846 0.5803922  ... 0.49411768 0.44705886 0.4039216 ]]
```

Gambar 4.13. Representasi citra

Proses utama pada *layer* konvolusi adalah proses konvolusi. Pada *layer* ini citra yang berupa representasi nilai matriks akan dihitung menggunakan teknik konvolusi seperti yang sudah dijelaskan pada BAB II dengan nilai *kernel* 3x3.

```
[[ 0. -1.  0.]
 [-1.  5. -1.]
 [ 0. -1.  0.]]
```

Gambar 4.14. Matriks kernel 3x3

Pada gambar 4.13 menunjukkan matriks dari sebuah citra yang akan melakukan konvolusi. Gambar 4.14 merupakan matriks citra hasil konvolusi dengan *kernel* yang terdapat pada gambar 4.15. setelah proses konvolusi akan menghasilkan matriks baru dengan ukuran banyak baris matriks citra dikurangi banyak baris pada matriks *kernel* ditambah satu.

```
[[0.10196079 0.11764707 0.3254902  ... 0.8941177  0.69411767 0.44705886]
 [0.20784315 0.09411766 0.16470589 ... 0.6666667  0.42352945 0.30980393]
 [0.46274513 0.15686275 0.0509804  ... 0.23529413 0.19607845 0.30980393]
 ...
 [0.6509804  0.63529414 0.6039216  ... 0.38431376 0.31764707 0.22352943]
 [0.63529414 0.6117647  0.5803922  ... 0.42352945 0.34509805 0.3019608 ]
 [0.6117647  0.59607846 0.5803922  ... 0.49411768 0.44705886 0.4039216 ]]
```

Gambar 4.15. Representasi citra hasil konvolusi

2. Proses *Rectified Units Layer* (ReLu)

   Fungsi aktivasi yang digunakan pada sistem ini merupakan ReLu. Fungsi aktivasi bertugas untuk membuat sebuah *neural network* menjadi non linear. Suatu fungsi dikatakan non linear jika kemiringannya tidak konstan. Gambar 4.16 merupakan contoh matriks hasil ReLu. Proses ReLu akan melemparkan nilai 0 untuk setiap nilai negatif sehingga setelah proses ReLu tidak ada lagi nilai pada matriks yang bernilai negatif.

```
[[0.         0.24313726 0.         1.1058825  0.9254901  0.2509805 ]
 [0.         0.         0.         0.6470588  0.         0.        ]
 [0.18431376 0.         0.         0.         0.         0.13333334]
 [0.6509804  0.62352943 0.6745098  0.32941183 0.54901963 0.5254903 ]
 [0.6784314  0.58431375 0.69803923 0.16470578 0.3137256  0.19999996]
 [0.6117647  0.5019608  0.65882355 0.42745104 0.43921575 0.23529407]]
```

Gambar 4.16. Representasi matriks hasil proses ReLu

3. Proses *subsampling*

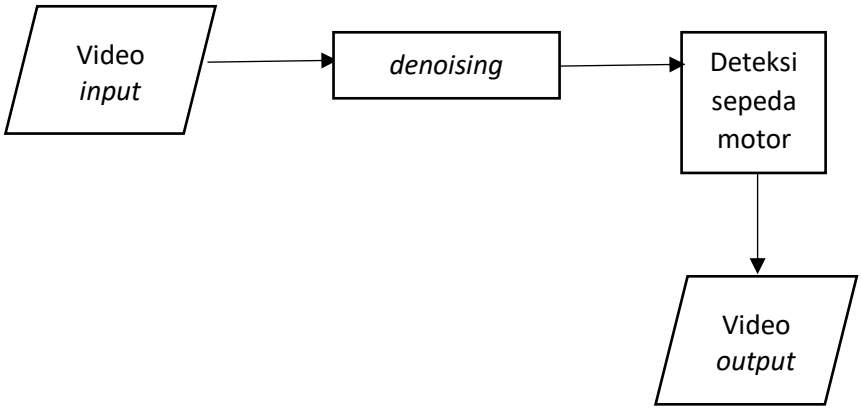
   Subsampling bertujuan untuk mereduksi citra. Pada sistem ini akan dilakukan proses *subsampling* dengan fitur 2x2 menggunakan *max pooling*. Proses penghitungan *subsampling* sudah dijelaskan pada BAB II. Gambar 4.17 menunjukkan hasil dari *subsampling*.

```
[[0.24313726 1.1058825  0.9254901 ]
 [0.6509804  0.6745098  0.54901963]
 [0.6784314  0.69803923 0.43921575]]
```

Gambar 4.17. Hasil proses *subsampling*

1.2.5. Mendeteksi sepeda motor

Setelah melakukan *training* dan *testing* maka sistem telah menyimpan ciri citra dari data set. Langkah selanjutnya adalah menggunakan ciri citra tersebut untuk mendeteksi sepeda motor yang terdapat pada video rekaman CCTV. Alur proses untuk mendeteksi sepeda motor digambarkan pada gambar 4.18.



Gambar 4.18. Alur proses mendeteksi sepeda motor

1. Video *input*

   Video *input* merupakan video rekaman CCTV pada saat lampu merah menyala yang akan di deteksi oleh sistem. Video berdurasi satu menit dengan format .mp4.

2. *Denoising*

   Tahap ini bertujuan untuk menghilangkan *noise* yang ada pada video. Ada beberapa proses yang dilakukan pada tahap ini yaitu *background subtraction*. Setelah melakukan *background subtraction* langkah selanjutnya yaitu proses *tresholding.* Selanjutnya dilakukan proses dilasi dan erosi untuk menghilangkan *noise* pada citra. Proses *denoising* digambarkan pada gambar 4.19.

3. Deteksi sepeda motor

   Dalam video *processing* diawali dengan proses ekstraksi *frame* dari video sehingga pemrosesan video dilakukan pada setiap *frame*. *Frame* yang diperoleh berukuran 1028 x 720 *pixel* maka dari itu dibutuhkan proses untuk membagi *frame* kedalam beberapa bagian atau biasa disebut *Region of Interest* (RoI). RoI yang digunakan disesuaikan dengan ukuran citra pada data set yaitu 64 x 128 piksel. Selanjutnya dari setiap RoI yang ada akan diprediksi dengan cara membandingkan RoI dengan ciri citra yang telah dimiliki pada tahap *training*. Jika nilai prediksi melebihi angka 0,5 maka objek yang ada pada RoI merupakan motor.

(a) Video setelah proses *background subtraction*

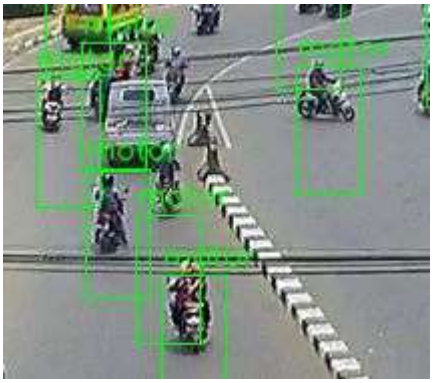
(b) Video setelah proses *tresholding*

(c) Video setelah proses dilasi

(d) Video setelah proses erosi

Gambar 4.19. Proses perubahan yang terjadi pada tahap *denoising*

4. Video *output*



Gambar 4.20. Contoh hasil sistem saat mendeteksi sepeda motor

Gambar 4.20 merupakan contoh hasil sistem pada saat mendeteksi sepeda motor. Setiap RoI yang memiliki hasil prediksi diatas 0,5 akan diberikan kotak berwarna hijau dengan label motor di atasnya.

1.2.6. Mendeteksi sepeda motor pelanggar marka jalan

Setelah sistem mampu mendeteksi sepeda motor, tahap selanjutnya adalah memilih sepeda motor mana saja yang melanggar marka jalan. Pelanggaran marka jalan yang dideteksi merupakan sepeda motor yang berhenti melewati *stop line*. Maka dari itu jika ada sepeda motor yang berhenti melewati *stop line* yang telah diinisialisasikan sebelumnya akan terdeteksi oleh sistem. Gambar 4.21 menampilkan hasil sistem saat mendeteksi pelanggar. Proses pendeteksian pelanggar dilakukan dengan membandingkan koordinat dari RoI yang memiliki nilai prediksi diatas 0,5 dengan koordinat *stop line*. Jika koordinat pada RoI lebih besar daripada koordinat *stop line* maka objek yang ada pada RoI tersebut merupakan pelanggar. Setiap sepeda motor yang terdeteksi sebagai pelanggar akan diberi kotak berwarna hijau dengan label pelanggar diatasnya.

Gambar 4.21. Contoh sistem saat mendeteksi pelanggar marka jalan

1.3. Pengembangan perangkat lunak

Pada sub bab ini akan menjelaskan tentang pengembangan perangkat lunak pada sistem pendeteksi sepeda motor pelanggar marka jalan.

1.3.1. Analisis sistem

Pada tahap ini penulis melakukan analisis kebutuhan sistem meliputi arus proses dan data sistem yang akan dibangun. Model sistem pendeteksi sepeda motor pelanggar marka jalan dirancang menggunakan diagram konteks (*context diagram)* serta diagram alir data (*data flow diagram*). Analisis sistem dapat dilihat pada dokumen teknis perangkat lunak.

1.3.1.1. Analisis *input*

Input pada sistem ini berupa video dari rekaman CCTV milik Dinas Perhubungan Kota Bandung sebagai *input*. Video rekaman yang digunakan merupakan video rekaman dengan kualitas terbaik yaitu 1028 x 720 piksel. Video *input* memiliki format .mp4 sebanyak 231 video dengan durasi 10 menit. Video *input* harus diubah kedalam bentuk citra dengan format .jpg agar bisa diolah sebagai data *training* dan *testing*. Video *input* setelah diubah kedalam citra menghasilkan 3000 data citra yang terdiri dari 1500 citra motor dan 1500 citra non motor.

1.3.1.2. Analisis *output*

Output dari sistem pendeteksi sepeda motor pelanggar marka jalan ini adalah kelas data. Kelas data ini berupa hasil klasifikasi sistem dari label pada data *training*. Data *output* yang dihasilkan berupa video yang telah mampu melakukan pendeteksian terhadap sepeda motor yang melanggar marka jalan. Jika terdapat sepeda motor yang melanggar sistem akan mendeteksi dengan cara memberi sebuah kotak serta label pelanggar pada kendaraan tersebut.

1.3.2. Deskripsi sistem

Sistem pendeteksi sepeda motor pelanggar marka jalan merupakan sistem yang berbasis *desktop*. Sistem dibangun dengan bahasa pemrograman *Python* dan menggunakan *library Opencv* untuk pemrosesan citra dan *library Tensorflow* untuk pembuatan model CNNs. Sistem ini memiliki 3 sub sistem utama yaitu sub sistem untuk melakukan pra proses data, sub sistem untuk membuat model dengan menggunakan CNNs dan sub sistem untuk mengklasifikasikan setiap *frame* pada video rekaman CCTV mana yang merupakan sepeda motor pelanggar *stop line* dan mana yang bukan pelanggar.

Sub sistem pra proses data merupakan proses untuk mempersiapkan data set menjadi data set yang siap untuk di *training*. Pada proses ini data set hasil transformasi dan pengambilan citra yang terdiri dari 3000 citra diberi label, disamakan ukurannya, di konversi menjadi *grayscale*, dan kemudian disimpan kedalam *file* berbentuk .npz. Hal ini dilakukan agar setiap sistem dijalankan kita

tinggal memanggil *file* tersebut dan tidak perlu mengulang kembali tahap pra proses.

Pada sub sistem kedua yaitu pembuatan model CNNs. Setelah memiliki data set yang sudah melalui tahap pra proses, kita bisa langsung menggunakannya untuk membuat model CNNs yang terbaik untuk sistem ini. Pembuatan model dilakukan dengan membangun *layer* sesuai dengan arsitektur yang ada, lalu kemudian di *training* pada data set yang dimiliki. Setelah menghasilkan model yang diinginkan model akan disimpan dalam bentuk *file* berekstensi *.tflearn*.

Sub sistem yang terakhir yaitu untuk pendeteksian sepeda motor pelanggar marka jalan. Pada tahap ini dilakukan beberapa proses untuk mempersiapkan video input yaitu *background subtraction*, *tresholding*, dilasi dan erosi untuk memisahkan *foreground* dan *background* serta menghilangkan *noise* pada video. Setelah video siap dilanjutkan dengan mengambil *Region of Interest* (RoI) pada setiap *frame*nya kemudian dibandingkan dengan model yang telah dimiliki. Hasil prediksi berupa angka desimal dimana jika nilainya lebih besar dari 0,5 dan koordinatnya lebih besar dari koordinat *stop line* maka RoI tersebut merupakan pelanggar. Setiap pelanggar akan diberi *bounding box* untuk membedakannya dengan objek lain.

1.3.3. Batasan perangkat lunak

Batasan perangkat lunak merupakan suatu daerah yang membatasi antara satu perangkat lunak dengan perangkat lunak yang lain atau dengan lingkungan luarnya. Batas perangkat lunak ini memungkinkan suatu perangkat lunak dipandang sebagai satu kesatuan dan menunjukkan ruang lingkup (*scope*) dari perangkat lunak tersebut. Batasan perangkat lunak dari sistem pendeteksi sepeda motor pelanggar marka jalan dijelaskan pada dokumen teknis perangkat lunak.

1.3.4. Implementasi *coding*

Setelah melakukan tahap analisis dan desain, tahap selanjutnya adalah implementasi *coding* yaitu mengubah desain tersebut ke dalam bahasa pemrograman sehingga bisa dimengerti oleh komputer. Bahasa pemrograman yang digunakan pada perangkat lunak ini adalah *Python*. Dalam proses implementasi dibutuhkan beberapa *library* yang digunakan untuk membantu proses *coding*

seperti *Tensorflow* untuk pembuatan model CNNs dan *Opencv* untuk pemrosesan citra. Implementasi *coding* dapat dilihat pada dokumen teknis perangkat lunak.

1.4. Pengujian

1.4.1. Skenario pengujian

Dataset terdiri dari 2 kelas data yaitu kelas motor dan kelas non motor. Masing-masing kelas terdiri dari 1500 citra, sehingga keseluruhan dataset ada 3000 citra. *Training* yang dilakukan pada dataset dilakukan dengan jumlah *epoch* 1000. Untuk menghitung rata-rata akurasi sistem menggunakan *cross validation*. *Cross validation* dilakukan dengan menggunakan 5 *fold*. Karena menggunakan 5 *fold* maka dataset dibagi menjadi 5 bagian dengan masing-masing bagian berisi 300 data pada tiap kelasnya.

Sistem pendeteksi sepeda motor pelanggar marka jalan menggunakan metode *Convolutional Neural Networks* harus melakukan proses pengujian. Pengujian dilakukan untuk mengetahui tingkat akurasi dari sistem. Pada penelitian ini ada beberapa skenario pengujian yang akan dilakukan, yaitu sebagai berikut:

1. Pengujian pertama menggunakan *fold* pertama sebagai data *testing* dan *fold* lainnya sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*.
2. Pengujian kedua menggunakan *fold* kedua sebagai data *testing* dan *fold* lainnya sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*.
3. Pengujian ketiga menggunakan *fold* ketiga sebagai data *testing* dan *fold* lainnya sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*.
4. Pengujian keempat menggunakan *fold* keempat sebagai data *testing* dan *fold* lainnya sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*.
5. Pengujian kelima menggunakan *fold* kelima sebagai data *testing* dan *fold* lainnya sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*.

6. Pengujian keenam dilakukan dengan menggunakan 1200 data sebagai data *testing* dan 1800 data sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*.
7. Pengujian ketujuh dilakukan dengan menggunakan 1800 data sebagai data *testing* dan 1200 data sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*.
8. Pengujian kedelapan dilakukan dengan menggunakan 600 data yang diambil dari masing-masing *fold* sebagai data *testing* dan 2400 data sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*.

1.4.2. Hasil pengujian

Hasil pengujian yang telah dilakukan berdasarkan skenario pengujian yang sudah dibuat sebelumnya dapat dilihat pada tabel 4.2. Pengujian pertama menggunakan *fold* pertama sebagai data *testing* dan *fold* lainnya sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*. Dengan menggunakan skenario pertama menghasilkan nilai akurasi sebesar 96,16% dengan nilai kegagalan sebesar 3,84%. Sistem mampu mengklasifikasikan citra dengan benar sebanyak 577 citra dan 23 citra diklasifikasian dengan salah. Pengujian kedua menggunakan *fold* kedua sebagai data *testing* dan *fold* lainnya sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*. Pada pengujian kedua ini citra yang diklasifikasikan dengan benar adalah sebanyak 585 citra dan 15 citra diklasifikasikan dengan salah. Maka dari itu sistem memiliki tingkat akurasi sebesar 97,5% dan nilai kegagalan sebesar 2,5%.

Pengujian ketiga menggunakan *fold* ketiga sebagai data *testing* dan *fold* lainnya sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*. Pada pengujian ketiga sistem memiliki akurasi sebesar 98,16% dan nilai kegagalan sebesar 1,84%. Pengujian keempat menggunakan *fold* keempat sebagai data *testing* dan *fold* lainnya sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*. Pengujian keempat diperoleh akurasi benar mencapai 98,5% sedangkan untuk nilai kesalahannya sebesar 1,5%.

Tabel 4.2. Hasil pengujian dari delapan skenario pengujian

Pengujian ke-	**Data *training***	**Data *testing***	**Jumlah benar**	**Jumlah salah**	**Akurasi (%)**	**Laju *error* (%)**
1.	2400	600	577	23	96.16	3.84
2.	2400	600	585	15	97.5	2.5
3.	2400	600	589	11	98.16	1.84
4.	2400	600	591	9	98.5	1.5%
5.	2400	600	587	13	97.83	2.17
6.	1800	1200	1173	27	97.75	2.25
7.	1200	1800	1741	59	96.72	3.28
8.	2400	600	550	50	91.66	2.34

Pengujian kelima menggunakan *fold* kelima sebagai data *testing* dan *fold* lainnya sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*. Pada pengujian kelima sistem memiliki akurasi sebesar 97,83% dan nilai kegagalan sebesar 2,17%. Pengujian keenam dilakukan dengan menggunakan 1200 data sebagai data *testing* dan 1800 data sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*. Pada pengujian keenam sistem memiliki akurasi sebesar 97,75% dan nilai kegagalan sebesar 2,25%.

Pengujian ketujuh dilakukan dengan menggunakan 1800 data sebagai data *testing* dan 1200 data sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*. Pada pengujian ketujuh sistem memiliki akurasi sebesar 96,72% dan nilai kegagalan sebesar 3,28%. Pengujian kedelapan dilakukan dengan menggunakan 300 data yang diambil dari masing-masing *fold* sebagai data *testing* dan 1200 data sebagai data *training*. Proses *training* dilakukan sebanyak 1000 *epoch*. Pengujian kedelapan menghasilkan akurasi sebesar 97,66% dan nilai kegagalan sebesar 2,34%.

Pada tabel 4.3 menunjukkan rata-rata akurasi dengan menggunakan 5 *fold*. Nilai akurasi tertinggi didapatkan pada skenario pengujian keempat dengan nilai 98.5%. Sedangkan nilai akurasi terendah diperoleh pada skenario pengujian pertama yaitu sebesar 96.16%.

Tabel 4.3. Rata-rata akurasi dengan menggunakan 5-*fold*

No	Pengujian ke-	Akurasi (%)
1.	1	96.16
2.	2	97.5
3.	3	98.16
4.	4	98.5
5.	5	97.83
Rata-rata		97.63

Selain menggunakan metode 5 *fold*, pengujian juga dilakukan dengan menggunakan variasi data *training* dan *testing* yang berbeda. Variasi pertama menggunakan data *training* sebanyak 1800 dan menghasilkan nilai akurasi sebesar 97.75. variasi kedua menggunakan data *training* sebanyak 1200 data dan memperoleh akurasi sebesar 96.72. Variasi ketiga dilakukan dengan menggunakan 2400 data sehingga menghasilkan nilai akurasi tertinggi yaitu 97.75. Sedangkan variasi keempat dilakukan dengan menggunakan 2400 data *training* dimana 2400 data itu diambil dari 5 *fold* yang telah dipisahkan terlebih dahulu. Tabel 4.4 menunjukkan hasil rata-rata akurasi dengan variasi pada data *training* dan data *testing*.

Tabel 4.4. Rata-rata akurasi dengan variasi data *training* dan *testing*

No	Jumlah *training*	Jumlah *testing*	Akurasi (%)
1.	1800	1200	96.72
2.	1200	1800	97.75
3.	2400	600	97.63
4.	2400*	600*	91.66
Rata-rata			95.94

Confusion matrix digunakan untuk merepresentasikan keakuratan dan kegagalan yang dilakukan sistem tiap kelas data. Tabel 4.5 menunjukkan *confusion matrix* pada setiap skenario pengujian. Pada pengujian pertama dengan menggunakan 600 citra pada *fold* pertama sebagai data *training* presisi terbesar ada

pada kelas data motor, sedangkan *recall* terbesar ada pada kelas data non motor. Akurasi terbesar didapat pada kelas non motor yaitu sebesar 98.33%.

Pada pengujian kedua dengan menggunakan 600 citra pada *fold* kedua sebagai data *training* presisi terbesar ada pada kelas data non motor, sedangkan *recall* terbesar ada pada kelas data motor. Akurasi terbesar ada pada kelas motor yaitu sebesar 98.66%. sedangkan pada pengujian ketiga presisi terbesar ada pada kelas data motor, sedangkan *recall* terbesar ada pada kelas data non motor namun kedua kelas tidak memiliki perbedaan nilai yang signifikan. Akurasi pada kelas motor sebesar 98.66% sedangkan pada kelas non motor 98.33%.

Pada pengujian keempat presisi terbesar ada pada kelas data motor, sedangkan *recall* terbesar ada pada kelas data non motor. Akurasi pada kelas data non motor hampir sempurna yaitu 99.33%. Presisi terbesar pada pengujian kelima ada pada kelas data non motor, sedangkan *recall* terbesar ada pada kelas data motor. Akurasi yang didapat pada kelas data motor sebesar 98.66% sedangkan pada kelas data non motor sebesar 97%.

Tabel 4.5. *Confusion matrix* pada setiap skenario pengujian

Klasifikasi		Pengujian ke-	Nilai Prediksi		Presisi	*Recall*	F1 *score*	Akurasi
			Motor	Non Motor				
Nilai Sebenarnya	**Motor**	1	282	18	98.25%	94%	96.08%	94%
	Non Motor		5	295	94.24%	98.33%	96.24%	98.33%
	Motor	2	296	4	96.41%	98.66%	97.52%	98.66%
	Non Motor		11	289	98.63%	96.33%	97.47%	96.33%
	Motor	3	294	6	98.32%	98%	98.16%	98.66%
	Non Motor		5	295	98.01%	98.33%	98.16%	98.33%
	Motor	4	293	7	99.32%	97.66%	98.48%	97.66%
	Non Motor		2	298	97.71%	99.33%	98.51%	99.33%
	Motor	5	296	4	97.04%	98.66%	97.85%	98.66%
	Non Motor		9	291	98.64%	97%	97.81%	97%
	Motor	6	582	18	98.47%	97%	97.73%	97%
	Non Motor		9	591	97.04%	98.5%	97.76%	98.5%
	Motor	7	864	36	96%	96%	96%	96%
	Non Motor		36	877	96%	96.06%	97.06%	97.44%
	Motor	8	281	19	90.06%	93.66%	91.83%	93.66%
	Non Motor		31	269	93.41%	89.66%	91.49%	89.66%

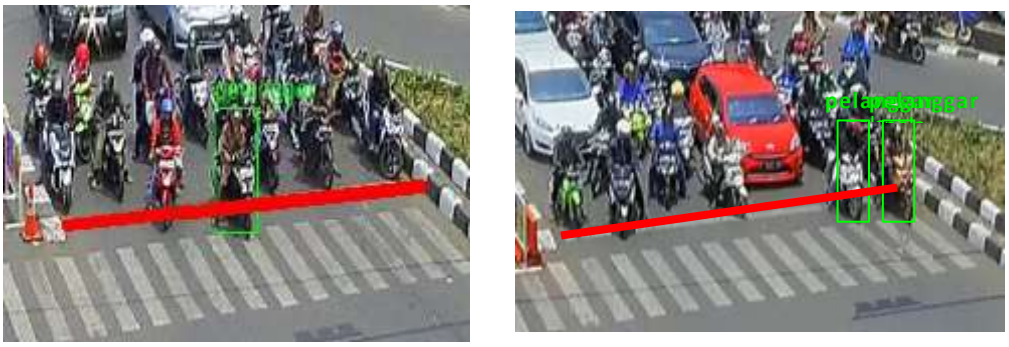
Tabel 4.6. Rata-rata pengujian tiap kelas data dari kedelapan skenario pengujian

Kelas		**Pengujian ke- (%)**								**Rata-rata (%)**
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
Motor	Presisi	98.25	96.41	98.32	99.32	97.04	98.47	94.12	90.06	96.49
	Recall	94	98.66	98	97.66	98.66	97	96.16	93.67	96.72
	F1 *score*	96.08	97.52	98.16	98.48	97.85	97.73	95.13	91.83	96.59
	Akurasi	94	98.66	98.66	97.66	98.66	97	96	93.66	96.78
Non motor	Presisi	94.24	98.63	98.01	97.71	98.64	97.04	98.06	93.41	96.96
	Recall	98.33	96.33	98.33	99.33	97	98.5	97	89.66	96.81
	F1 *score*	96.24	97.47	98.16	98.51	97.81	97.76	97.52	91.49	96.87
	Akurasi	98.33	96.33	98.33	99.33	97	98.5	97.44	89.66	96.86

Presisi terbesar pada pengujian keenam ada pada kelas data motor, sedangkan *recall* terbesar ada pada kelas data non motor. Akurasi terbesar diperoleh pada kelas data non motor yaitu 98.5%. Sedangkan pada pengujian ketujuh presisi terbesar ada pada kelas data non motor, sedangkan *recall* terbesar ada pada kelas data non motor. Akurasi terbesar diperoleh pada kelas data non motor yaitu 97.44% sedangkan kelas data motor memperoleh akurasi sebesar 96%.

Pada pengujian kedelapan presisi terbesar ada pada kelas data non motor, sedangkan *recall* terbesar ada pada kelas data motor. Akurasi pada kelas data motor sebesar 93.66% lebih tinggi daripada akurasi pada kelas data non motor yaitu 89.66%. Tabel 4.6 merupakan tabel rata-rata akurasi dari tiap kelas data. Kelas data non motor sedikit lebih unggul dari kelas data motor. Rata-rata akurasi, presisi, *recall* dan f1 *score* dari masing-masing kelas data tidak menunjukkan perbedaan yang signifikan yaitu masing-masing nilai berada pada angka 96%.

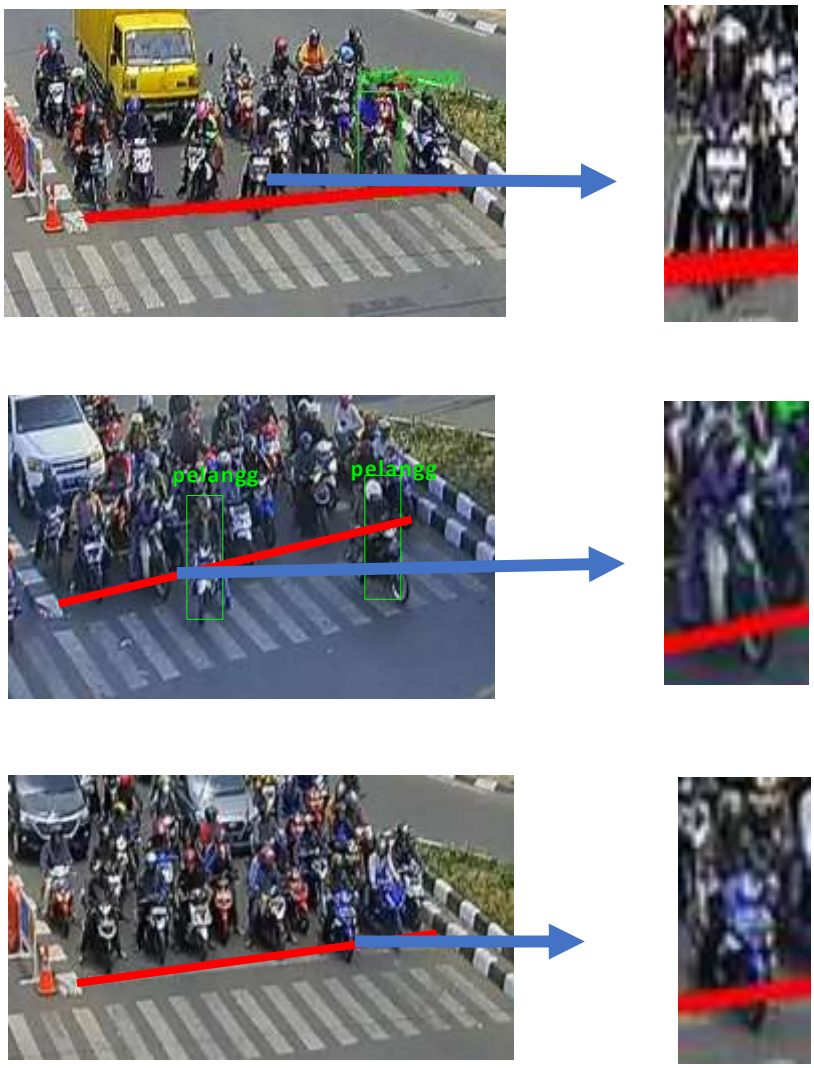
1.4.3. Pembahasan hasil penelitian



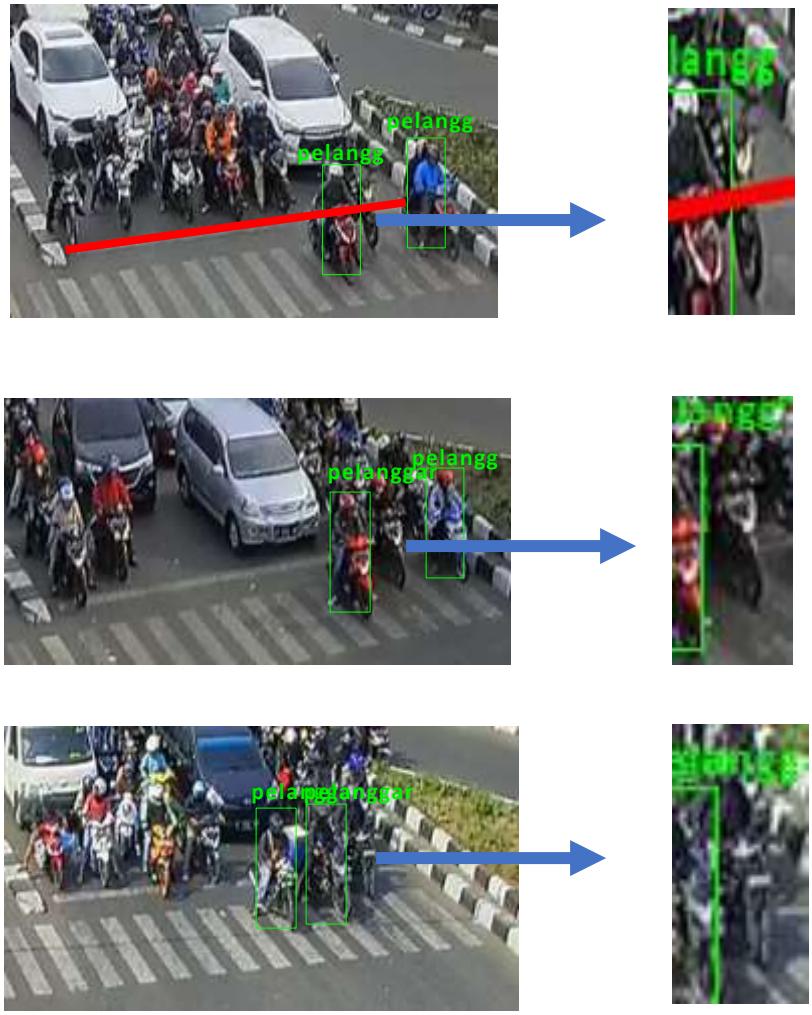
Gambar 4.22. Contoh sistem saat berhasil mendeteksi pelanggar marka jalan

Langkah awal dalam pembuatan sistem pendeteksi sepeda motor pelanggar marka jalan dilakukan dengan mencari arsitektur CNNs terbaik. Proses tersebut dinamakan *tuning* parameter. Proses *tuning* parameter dilakukan dengan mengacu arsitektur yang pernah digunakan sebelumnya yang terdapat pada gambar 3.3. pada saat arsitektur tersebut digunakan untuk sistem ini menghasilkan akurasi yang buruk sehingga arsitektur mengalami perubahan seperti yang dijelaskan pada sub bab 4.2.4. Perbedaan akurasi ini disebabkan karena data yang digunakan pada sistem ini diambil dari CCTV Dinas Perhubungan Kota Bandung dengan resolusi 1280 x 720 piksel.

Setelah menemukan arsitektur terbaik, maka dilakukan proses implementasi sistem untuk mendeteksi pelanggar marka jalan seperti yang dijelaskan sebelumnya pada sub bab 4.2.6. Setelah melakukan implementasi sistem mampu mendeteksi sepeda motor pelanggar marka jalan seperti yang ditunjukkan pada gambar 4.22, walaupun masih ada beberapa kesalahan yang dilakukan oleh sistem dalam melakukan pendeteksian seperti yang terdapat pada gambar 4.23. Jika dilihat pada gambar 4.23 sepeda motor pelanggar marka jalan yang tidak terdeteksi sistem disebabkan oleh bagian dari sepeda motor yang melewati *stop line* hanya sebagian roda depannya saja sehingga sistem tidak mengenalinya sebagai pelanggar. Sedangkan pada gambar 4.24 kesalahan terjadi karena pelanggar tersebut tertutup oleh kendaraan lain sehingga sistem tidak dapat mengenalinya.

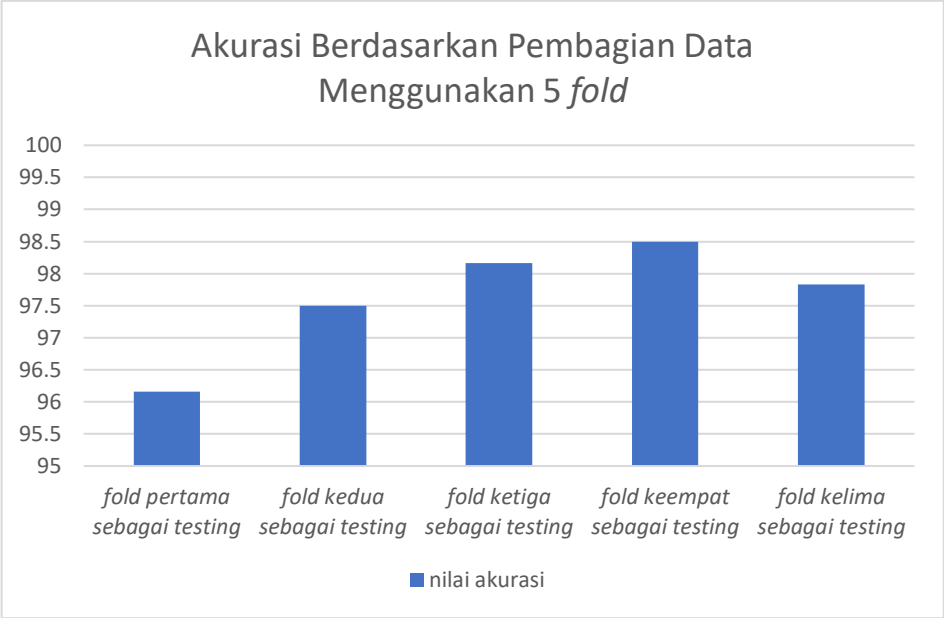


Gambar 4.23. Kesalahan sistem yang disebabkan hanya sebagian roda depannya saja yang melewati *stop line*



Gambar 4.24. Kesalahan sistem yang disebabkan tertutup oleh kendaraan lain

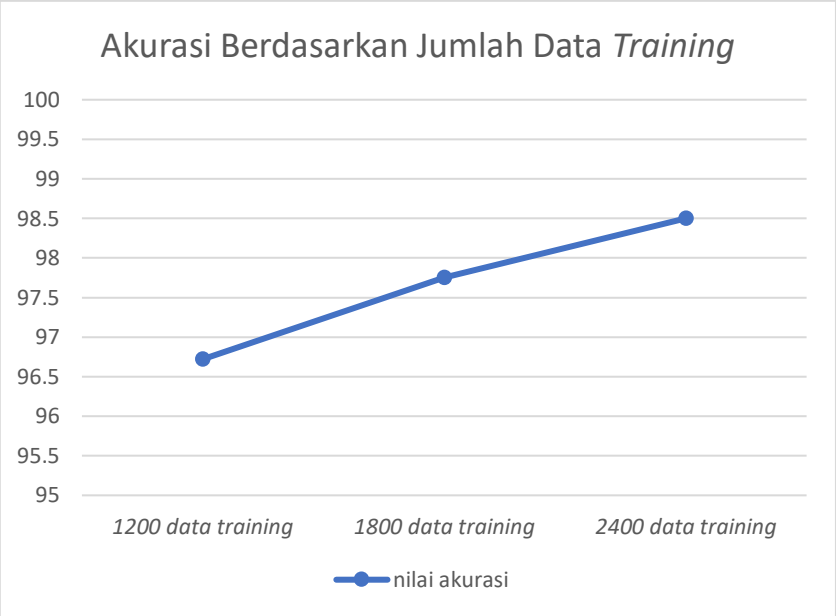
Sistem pendeteksi sepeda motor pelanggar marka jalan sudah menghasilkan rata-rata akurasi yang cukup baik yaitu 95.94%. Perubahan akurasi yang ditunjukkan pada gambar 4.25 terjadi pada saat komposisi data *training* dan data *testing* diubah menggunakan 5 *fold* seperti yang sudah dijelaskan pada sub bab 4.4.1. Akurasi terbaik diperoleh pada skenario pengujian ke 4 yaitu sebesar 98.5% dengan menggunakan 600 data pada *fold* keempat sebagai data *testing* dan *fold* lainnya sebagai data *training*. Perubahan akurasi yang terjadi menunjukkan bahwa variasi yang ada pada data *training* sangat mempengaruhi akurasi pada sistem.



Gambar 4.25. Grafik akurasi berdasarkan pembagian data menggunakan 5 *fold*

Perubahan akurasi juga terjadi pada saat melakukan penambahan dan pengurangan jumlah data seperti yang digambarkan pada gambar 4.26. Terjadi peningkatan akurasi pada saat data *training* ditambah. Hal ini menunjukkan bahwa sistem akan menjadi lebih pintar jika lebih banyak belajar.

Sedangkan bila dilihat dari hasil *confusion matrix* pada tabel 4.6 menunjukkan bahwa rata-rata presisi mencapai 96.49% untuk kelas data motor dan 96.96% untuk kelas data non motor. Sedangkan rata-rata *recall* pada kelas data motor yaitu 96.72% dan 96.81% pada kelas data non motor. Hal tersebut membuktikan bahwa sistem belum mampu mendeteksi pelanggar secara sempurna. Jika kita lihat pada tabel 4.5 sistem melakukan kesalahan dalam klasifikasi objek dimana sebanyak 23 motor diklasifikasikan sebagai non motor dan 36 non motor diklasifikasikan sebagai motor dan pada tabel 10 terlihat laju *error* tertinggi yaitu mencapai 3.28%.



Gambar 4.26. Grafik akurasi berdasarkan jumlah data training

BAB V
PENUTUP

5.1. Kesimpulan

Berikut kesimpulan yang di dapat pada penelitian ini:

1. Metode *Convolutional Neural Networks* (CNNs) berhasil diimplementasikan pada sistem untuk mendeteksi sepeda motor. Sistem pendeteksi sepeda motor pelanggar marka jalan menggunakan metode CNNs menggunakan data yang diambil dari CCTV Dinas Perhubungan Kota Bandung pada persimpangan jalan Cibaduyut karena memiliki resolusi yang paling tinggi yaitu 1280 x 720 *pixel*. Tahapan dalam pembuatan sistem dimulai dengan pengambilan dan transformasi citra yaitu dengan mengambil objek motor dan non motor saja pada video input. Tahap selanjutnya yaitu menginisalisasikan *stop line* dengan menentukan koordinat x dan y secara manual untuk mempermudah sistem. Selanjutnya masuk ke tahap pra proses, dimana pada tahap ini terdiri dari beberapa proses yaitu proses pelabelan data set yang terdiri dari dua kelas data menjadi 01 untuk motor dan 10 untuk non motor, proses *scalling* data citra menjadi 64 x 128 *pixel*, proses konversi citra dari RGB menjadi *grayscale* kemudian dilakukan proses *histogram equalizazion* untuk memperoleh penyebaran derajat keabuan yang sama rata, dan proses normalisasi untuk mengurangi resolusi citra. Setelah melakukan pra proses, tahap selanjutnya yaitu tahap *training* dan *testing* menggunakan CNNs. Pada proses *training* ini arsitektur CNNs yang digunakan terdiri dari 5 *layer* yaitu 2 *layer* konvolusi yang diikuti *layer subsampling* dan 3 *layer fully connected.* Selanjutnya yaitu tahap pendeteksian sepeda motor, pada tahap ini ada beberapa proses yang dilakukan untuk menghilangkan *noise* pada video yaitu dengan proses *background subtraction*, *treshold*, dilasi dan erosi. Sistem dapat mendeteksi sepeda motor yang terekam kamera CCTV dengan cukup baik, walaupun ada beberapa sepeda motor yang tidak terdeteksi oleh sistem. Hasil klasifikasi pada sistem setelah melakukan delapan kali

pengujian menunjukkan rata-rata akurasi yang baik yaitu 95.94% seperti yang ditunjukkan pada gambar 4.24.

2. Setelah sistem dapat mendeteksi sepeda motor yang terekam CCTV, sistem akan memilih sepeda motor mana saja yang melewati *stop line* pada saat lampu merah menyala. Pendeteksian pelanggar dilakukan dengan membandingkan koordinat *stop line* dengan motor yang terdeteksi. Jika koordinat motor lebih besar dari koordinat *stop line* maka sepeda motor tersebut merupakan sepeda motor yang melanggar marka jalan. Sepeda motor yang terdeteksi melanggar marka jalan diberi tanda dengan kotak hijau serta label pelanggar diatasnya agar mudah dikenali untuk membantu petugas dalam meningkatkan kedisiplinan pengendara.

5.2. Saran

Berikut saran penulis untuk penelitian selanjutnya dalam sistem pendeteksi sepeda motor pelanggar marka jalan menggunakan metode *Convolutional Neural Networks* (CNNs):

1. Pada penelitian selanjutnya dapat dilakukan dengan melakukan penelitian untuk mendeteksi *stop line* karena pada penelitian ini *stop line* masih diinisialisasi secara manual. Selain itu juga penelitian selanjutnya bisa dilakukan dengan meneliti proses identifikasi terhadap sepeda motor pelanggar marka jalan seperti memberikan informasi mengenai plat nomer kendaraan atau jenis kendaraan karena pada penelitian ini sistem hanya mendeteksi sepeda motor pelanggar marka jalan saja tidak melakukan identifikasi terhadap pelanggar.
2. Menambahkan jumlah data citra yang digunakan untuk proses *training*. Seperti yang terlihat pada gambar 4.26 dimana terjadinya penurunan akurasi saat data *training* dikurangi, maka dari itu untuk meningkatkan akurasi dapat dilakukan dengan menambahkan jumlah data *training*.
3. Menambahkan jenis pelanggaran lain seperti pengendara sepeda motor yang tidak menggunakan helm, atau mobil yang berhenti di ruang henti khusus sepeda motor pada saat lampu merah menyala dan sebagainya.

DAFTAR PUSTAKA

Arel, I., Rose, D., & Karnowski, T. (2010). Deep Machine learning-A New Frontier in Artificial Intelligence Research. *IEEE Computational Intelligence Magazine*, *5*(4), 13–18.

Balaji, S. (2012). Waterfall vs V-Model vs Agile : A Comparative Study on SDLC. *Waterfall vs V-Model vs Agile: A Comparative Study On SDLC*, *2*(1), 26–30.

Bassil, Y. (2012). A Simulation Model for the Waterfall Software Development Life Cycle. *International Journal of Engineering & Technology*, *2*(5), 2049–3444. https://doi.org/10.15680/ijircce.2015.0305013

Bautista, C., Dy, C., & Manalac, M. (2016). Convolutional Neural Network For Vehicle Detection In Low Resolution Traffic Videos. *2016 IEEE Region*, 277–281.

Buch, N., Velastin, S. A., & Orwell, J. (2011). A Review of Computer Vision Techniques For The Analysis of Trban Traffic. *IEEE Transactions on Intelligent Transportation Systems*, *12*(3), 920–939.

Castellà, J., & Pérez, J. (2004). Sensitivity to Punishment and Sensitivity to Reward and Traffic Violations. *Accident Analysis and Prevention*, *36*(6), 947–952.

Claesson, L., & Hansson, B. (2017). Deep Learning Methods and Applications.

Coifman, B., Beymer, D., McLauchlan, P., & Malik, J. (1998). A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance. *Transportation Research Part C: Emerging Technologies*, *6*(4), 271–288.

Deng, L. (2013). Three Classes of Deep Learning Architectures and Their Applications: A Tutorial Survey. *Research.Microsoft.Com*.

Deng, L., & Yu, D. (2014). Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing*, *7*(3–4), 197–387.

Elander, J., West, R., & French, D. (1993). Behavioral Correlates of Individual Differences in Road-Traffic Crash Risk: An Examination of Methods and Findings. *Psychological Bulletin*, *113*(2), 279–294.

Fukushima, K. (1980). Neocognitron: A self-Organizing Neural Network Model for A Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, *36*(4), 193–202.

Goodman, S. D., & Rhodes, W. T. (2009). Symbolic Substitution Applications to Image Processing. *Applied Optics*, *27*(9), 1708.

Kafai, M., & Bhanu, B. (1983). Dynamic Bayesian Networks for Vehicle Classification in Video. *IEEE Transactions on Industrial Informatics*, *264*(1), 119–128.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, 1–9.

Kumar, T., & Verma, K. (2010). A Theory Based on Conversion of RGB image to Gray image. *International Journal of Computer Applications*, *7*(2), 5–12.

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lecun, Y., Bottou, L., Bengio, Y., & Ha, P. (1998). Gradient-Based Learning Applied to Document Recognition. *IEEE*, (November), 1–46.

Li, B. (2017). 3D Fully Convolutional Network for Vehicle Detection in Point Cloud. *IEEE International Conference on Intelligent Robots and Systems*, *2017–Septe*, 1513–1518.

Liliana, D., Baji, F., & Teodoru, R. (2017). Overview of Deep Learning in Medical Imaging. *Annals Of The University Of Craiova*, *14*(1), 8–18.

Limantoro, S. E., Kristian, Y., Purwanto, D. D., Informasi, T., Tinggi, S., Surabaya, T., … Timur, J. (2017). Deteksi Pengendara Sepeda Motor Menggunakan Deep Convolutional Neural Networks, 2–9.

Lindley, J. A. (1987). Urban Freeway Congestion: Quantification of the Problem and Effectiveness of Potential Solutions. *ITE Journal*, 27–32.

Maini, R., & Aggarwal, H. (2010). A Comprehensive Review of Image Enhancement Techniques, *2*(3), 8–13.

Mussa, R., Kwigizile, V., & Selekwa, M. (2006). Probabilistic Neural Networks Application for Vehicle Classification. *Journal of Transportation Engineering*, *132*(4), 293–302.

Nguyen, K., Fookes, C., Ross, A., & Sridharan, S. (2018). Iris Recognition With Off-the-Shelf CNN Features : A Deep Learning Perspective. *IEEE Access*, *6*, 18848–18855.

Putra, I. W. S. E. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) Pada Caltech 101. *Jurnal Teknik ITS*, *5*(1), 65–69.

Putu, W., Made, W., & Dessy, S. (2013). Pengembangan Aplikasi Pembuatan Pola Motif Batik Dengan Menggunakan Pengolahan Citra Digital, *1*, 1–2. https://doi.org/10.1038/ncomms294

Republik Indonesia. 2014. Peraturan Menteri Perhubungan Republik Indonesia Nomor PM 34 Tahun 2014 Tentang Marka Jalan. Jakarta: Dinas Perhubungan

Repubik Indonesia. 2009. Undang-Undang Republik Indonesia nomor 22 Tahun 2009 Tentang Lalu Lintas dan Angkutan Jalan. Lembaran RI Tahun 2009 No.96. Jakarta: Sekretariat Negara

Rere, L. M. R., Fanany, M. I., & Arymurthy, A. M. (2015). Simulated Annealing Algorithm for Deep Learning. *Procedia Computer Science*, *72*, 137–144.

Roni, A., & Adi, E. (2000). Studi Analisis Pengenalan Pola Tulisan Tangan Angka Arabic (Indian) Menggunakan Metode K- Nearest Neighbors Dan Connected Component Labeling, *50*(2), 102–119.

Shamsher, R., Mohamamd, &, & Abdullah, N. (2013). Traffic Congestion in Bangladesh-Causes and Solutions: A Study of Chittagong Metropolitan City. *Asian Business Review*, *2*(3), 2304–2613.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. *Elsevier*, *32*, 323–332.

Tariq, U., Jamal, H., Shahid, M. Z. J., & Malik, M. U. (2004). Face detection in color images, a robust and fast statistical approach. *Proceedings of INMIC 2004 - 8th International Multitopic Conference*, 73–78. https://doi.org/10.1109/INMIC.2004.1492849

Thakur, A., & Mishra, D. (2015). Fuzzy Contrast Mapping for Image Enhancement. *2nd International Conference on Signal Processing and Integrated Networks, SPIN 2015*, 549–552.

Utari, C. T. (2016). Implementasi Algoritma Run Length Encoding Untuk Perancangan Aplikasi Kompresi Dan Dekompresi File Citra. *Jurnal TIMES*, *V*(2), 24–31.

Vandra, K., & Kulkarni. (2012). Electronics and Communication Spatial Domain

Image Enhancement. *Journal of Information, Knowledge and Research in Electronics and Communication*, *2*(1), 229–241.

Xie, Y., Gu, S., Liu, Y., Zuo, W., Zhang, W., & Zhang, L. (2016). Weighted Schatten p-Norm Minimization for Image Denoising and Background Subtraction. *IEEE Transactions on Image Processing*, *25*(10), 4842–4857. https://doi.org/10.1109/TIP.2016.2599290

Yao, Z., Lai, Z., & Wang, C. (2017). Image Enhancement Based on Equal Area Dualistic Sub-image and Non-parametric Modified Histogram Equalization Method. *Proceedings - 2016 9th International Symposium on Computational Intelligence and Design, ISCID 2016*, *1*(1), 447–450.

Zhang, B., Zhou, Y., & Pan, H. (2013). Vehicle classification with confidence by classified vector quantization. *IEEE Intelligent Transportation Systems Magazine*, *5*(3), 8–20.