SKRIPSI

PERANCANGAN SISTEM *MACHINE LEARNING* UNTUK KLASIFIKASI KENDARAAN PADA PERSIMPANGAN *TRAFFIC LIGHT*

Diajukan Sebagai Salah Satu Persyaratan Untuk Memperoleh Gelar Sarjana (S1)

Jurusan Teknik Elektro Fakultas Teknik Universitas Tidar

Disusun Oleh:

Abid Juliant Indraswara

1810501059

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS TIDAR

2022

HALAMAN JUDUL

PERANCANGAN SISTEM *MACHINE LEARNING* UNTUK KLASIFIKASI KENDARAAN PADA PERSIMPANGAN *TRAFFIC LIGHT*

Diajukan Sebagai Salah Satu Persyaratan Untuk Memperoleh Gelar Sarjana (S1)

Jurusan Teknik Elektro Fakultas Teknik Universitas Tidar

Disusun Oleh:
Abid Juliant Indraswara
1810501059

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS TIDAR

2022

HALAMAN PENGESAHAN PEMBIMBING
SKRIPSI

Skripsi dengan judul:

PERANCANGAN SISTEM *MACHINE LEARNING* UNTUK KLASIFIKASI KENDARAAN PADA PERSIMPANGAN *TRAFFIC LIGHT*

Disusun Oleh:

Nama : Abid Juliant Indraswara

NPM : 1810501059

Telah diperiksa dan disetujui oleh:

Pembimbing I

Bagus Fatkhurrozi, S.T., M.T., IPM.
NIP. 197910122005011001

Tanggal: 25-7-2022

Pembimbing II

Hery Teguh Setiawan, S.T., M.Eng.
NIP. 198701092019031005

Tanggal: 25/07/2022

Mengetahui,

Dekan Fakultas Teknik Universitas Tidar

Dr. Ir. Sapto Nisworo, M.T., IPU.
NIP. 195909281991031001

HALAMAN PENGESAHAN PENGUJI
SKRIPSI

Skripsi dengan judul:

PERANCANGAN SISTEM *MACHINE LEARNING* UNTUK KLASIFIKASI KENDARAAN PADA PERSIMPANGAN *TRAFFIC LIGHT*

Disusun Oleh:

Nama : Abid Juliant Indraswara

NPM : 1810501059

Telah berhasil dipertahankan di depan Dewan Penguji dalam ujian pendadaran skripsi dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik Jurusan Teknik Elektro Fakultas Teknik Universitas Tidar.

Magelang, Juli 2022

Di hadapan penguji:

1. **Bagus Fatkhurrozi, S.T., M.T., IPM.** : .....
   NIP. 197910122005011001

2. **Hery Teguh Setiawan, S.T., M.Eng.** : .....
   NIP. 198701092019031005

3. **Ir. Deria Pravitasari, S.T., M.Eng., IPM.** : .....
   NIP. 198108092021212006

Mengetahui,

Dekan Fakultas Teknik Universitas Tidar

Dr. Ir. Sapto Nisworo, M.T., IPU.
NIP. 195909281991031001

HALAMAN PERNYATAAN

Saya yang bertanda tangan dibawah ini:

Nama : Abid Juliant Indraswara

NPM : 1810501059

Program Studi : S1 Teknik Elektro

Alamat : Botton Gg Waluyo No 25 RT 01/RW 06, Kelurahan Magelang, Kecamatan Magelang Tengah, Kota Magelang, Provinsi Jawa tengah

Menyatakan dengan sesungguhnya bahwa skripsi dengan judul:

PERANCANGAN SISTEM *MACHINE LEARNING* UNTUK KLASIFIKASI KENDARAAN PADA PERSIMPANGAN *TRAFFIC LIGHT*

Benar-benar merupakan hasil karya sendiri, kecuali dalam pengutipan substansi disebutkan sumbernya dan belum pernah diajukan pada institusi manapun, serta bukan karya jiplakan atau plagiat. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi. Demikian pernyataan ini saya buat dengan sebenarnya. Tanpa ada tekanan dan paksaaan dari pihak manapun serta bersedia mendapatkan sanksi akademik jika ternyata dikemudian hari pernyataan ini tidak benar.

Magelang, Juli 2022

Abid Juliant Indraswara

NPM. 1810501059

MOTTO

"*Utamakan Sholawat*"

PRAKATA

Puji syukur penulis panjatkan kehadirat Allah SWT atas segala limpahan, karunia, serta hidayah-Nya, sehingga penulis dapat menyelesaikan tugas akhir dengan judul "Perancangan Sistem *Machine Learning* Untuk Klasifikasi Kendaraan Pada Persimpangan *Traffic Light*".

Penulis dalam menyusun laporan penelitian ini untuk memenuhi persayaratan dalam menyelesaikan pendidikan S-1 pada jurusan Teknik Elektro Universitas Tidar. Laporan penelitian yang disusun masih memiliki beberapa kekurangan dengan harapan penyempurnaan pada pembaca akan mempunyai nilai dan manfaat yang berarti pada realisasi penelitian selanjutnya dalam lingkungan maupun diluar lingkungan Universitas Tidar Magelang.

Penulis berharap semoga laporan tugas akhir yang diajukan dapat bermanfaat bagi almamater tercinta, terutama rekan-rekan mahasiswa Teknik Elektro Universitas Tidar, serta bangsa dan negara. Penulis menyadari masih banyak hal yang perlu diperbaiki, karena masih terdapat permasalahan yang belum terjawab dan dibahas pada laporan tugas akhir ini.

Proses penyusunan laporan tugas akhir penulis memerlukan bimbingan, dukungan serta saran dari berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada :

1. Prof. Dr. Ir. Mukh Arifin, M.Sc. selaku Rektor Universitas Tidar;
2. Dr. Ir. Sapto Nisworo, M.T., IPU. selaku Dekan Fakultas Teknik Universitas Tidar;
3. Ir. Deria Pravitasari, S.T., M.Eng., IPM selaku Ketua Jurusan Teknik Elektro Fakultas Teknik Universitas Tidar;
4. Bapak Bagus Fatkhurrozi, S.T., M.T., IPM. selaku Dosen Pembimbing I;
5. Bapak Hery Teguh Setiawan, S.T., M.Eng. selaku Dosen Pembimbing II;
6. Bapak Ir. Johan Pamungkas, M.T. Terima kasih atas bimbingannya dari memberikan amanah melalui penelitian pada semester empat hingga arahan untuk mengajukan topik skripsi yang sudah saya susun.

7. Seluruh dosen dan karyawan Jurusan Teknik Elektro Fakultas Teknik Universitas Tidar;
8. Kedua orang tua penulis yang tercinta Bapak Jarot dan Ibu Iin yang selalu memberi doa, semangat, dan motivasi.
9. Adik saya Ahmad Abdillah Indragiri yang memberikan kesempatan, dukungan dan semangat berkuliah dari awal semester hingga saya dapat menyelesaikan skripsi.
10. Sevty Ayu Saputry yang memberikan motivasi untuk bergabung dengan penelitian dosen sehingga melatarbelakangi topik skripsi yang saya susun;
11. Agus Wahyu Widodo yang meminjamkan saya laptop selama semester akhir ini sehingga dapat tersusun skripsi dengan sangat baik;
12. Muhammad Nur Fauzi yang menyempatkan waktu membantu melakukan survei pendahuluan skripsi pada saat saya sedang melakukan magang;
13. Bapak Hery Teguh Setiawan S.T., M.Eng. dan keluarganya terima kasih telah berbagi cerita, pengalaman, dan banyak ilmu;
14. Teman-teman Teknik Elektro Angkatan 2018 Universitas Tidar terkhusus teman-teman anggota SPECTRA, teman-teman *core team* GDSC Univeritas Tidar, dan teman-teman Bangkit 2022 Universitas Tidar;
15. Pihak-pihak lain yang telah membantu penulis yang tidak dapat penulis sebutkan satu persatu.

Penulis mengharapkan saran dan kritik yang membangun dari segenap pembaca untuk perbaikan penelitian ini. Akhirnya penulis berharap semoga laporan penelitian ini dapat memberikan manfaat bagi semua pihak.

Magelang, 2022

Penulis

Abid Juliant Indraswara

NPM. 1810501059

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

INTISARI

Penelitian ini berfokus pada perancangan sistem *machine learning* untuk melakukan klasifikasi kendaraan pada persimpangan lampu lalu lintas. Sistem machine learning yang dibuat pada penelitian menghasilkan model *training machine learning* yang kemudian diintegrasikan ke dalam aplikasi sehingga dapat melakukan klasifikasi dan penghitung kendaraan kelas roda dua serta roda empat baik secara *real time* maupun rekaman video pada CCTV lampu lalu lintas. Dataset gambar yang dikumpulkan disesuaikan dengan lingkungan yaitu lalu lintas. Semakin banyak dataset maka akan lebih baik lagi performa dari model machine learning yang dibuat. Algoritma YOLO yang merupakan pengembangan algoritma *Convolutional Neural Network* digunakan karena memberikan performa sangat baik serta training model machine learning yang tidak memakan banyak waktu. Aplikasi yang dibuat memiliki fitur dalam menghitung nilai kendaraan yang melaju dari atas ke bawah dan dari bawah ke atas. Pengujian dilakukan per menit dengan pengambilan data selama 10 kali. Aplikasi yang sudah diintegrasikan model machine learning diuji dengan total objek yang diprediksi yaitu sebanyak 425 data gambar uji dengan akurasi yang dihasilkan 82% dengan MSE (*mean absolute error*) sebesar dan metrik evaluasi objek deteksi MAP (*mean average precision*) yaitu 97.51%. UI (*User Interface*) dari aplikasi sistem menggunaka *library* python yang dapa digunakan dengan baik dan mudah mengakses melalui dekstop.

Kata kunci : ***Sistem Machine Learning Real Time, Convolutional Layer, Paramater Konfigurasi Model Machine Learning, Visual Studi Code***

ABSTRACT

This research focuses on designing a machine learning system to carry out vehicles at traffic light intersections. The machine learning system created in this research produces a machine learning training model which is then integrated into the application so that it can classify and calculate two-wheeled and four-wheeled class vehicles both in real time and video recordings on CCTV traffic lights. The image dataset collected is adapted to the environment, namely traffic. The more datasets, the better the performance of the machine learning models created. The YOLO algorithm, which is the development of the Convolutional Neural Network algorithm, is used because it provides excellent performance and a machine learning training model that does not take much time. The application made has a feature in calculating the value of vehicles traveling from top to bottom and from bottom to top. The test is carried out per minute by taking data for 10 times. Applications that have integrated machine learning models were tested with a total predicted object of 425 test image data with an accuracy of 82% with an MSE (mean absolute error) of and evaluation of object detection MAP (mean average precision) of 97.51%. The UI (User Interface) of the system application uses a python library that can be used properly and is easy to access via the desktop.

Keywords: Real Time Machine Learning System, Convolutional Layer, Machine Learning Model Configuration Parameter, Visual Studio Code

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kendaraan merupakan alat transportasi yang saat ini sudah menjadi kebutuhan utama dalam melakukan kegiatan sehari - hari. Jenis kendaraan yang umum digunakan tergolong menjadi dua yaitu kendaraan roda dua dan kendaraan roda empat dengan kapasitas berbeda - beda. Beberapa tahun terakhir, jumlah volume kendaraan pada lalu lintas di Indonesia meningkat pesat dangan tidak diiringi pembangunan infrastruktur penunjang pengaturan lalu lintas yang memadai. Menurut data dari Badan Pusat Statistik (BPS) jumlah seluruh kendaraan bermotor (sepeda motor, mobil penumpang, mobil bis dan mobil barang) di Indonesia pada tahun 2018 sebesar 126.508.776 juta. Data tersebut meningkat dalam tiga tahun terakhir yaitu pada tahun 2018 hingga 2020 sebesar 9.628.675 juta. Sehingga pada 2020 data dari BPS mencatat total umlah kendaraan sebanyak 136.137.451 juta. Data tersebut diambil melalui pendaftaran registrasi kendaraan yang masuk (BPS, 2018). Jumlah kendaraan yang terus bertambah tiap tahunnya, yang menjadi salah satu sebab kemacetan lalu lintas di jalan raya khususnya pada persimpangan *traffic light*. Hal tersebut terjadi dikarenakan lampu lalu lintas pada beberapa wilayah masih banyak menggunakan pengaturan manual maupun timer otomatis dalam melakukan pengaturan hitungan lampu.

Tindakan pencegahan kepadatan lalu lintas dilakukan oleh pemerintah salah satunya dengan menambahkan kamera CCTV pada lampu lalu lintas yang memiliki potensi kepadatan arus lalu lintas maupun kemacetan. Visualisasi dari kamera CCTV di monitor oleh operator atau pengawas kamera CCTV pada pusat pemantaun yang umumnya berada di dinas perhubungan daerah. Akan tetapi, manusia (operator / pengawas kamera) memiliki kemampuan terbatas untuk memonitor layar – layar pada pusat pemantauan yaitu rata – rata 20 – 30 layar secara bersamaan serta melakukan pendataan status dari ruas jalan yang dipantau (Ma'ali and Hendriyawan A, 2019).

Pemerintah khususnya dinas perhubungan di setiap daerah memiliki kebijakan sendiri dalam melakukan penerapan mekanimse pengontrolan lampu lalu lintas jarak jauh yang sebagian besar menggunakan kamera CCTV atau media kamera digital. Dinas Perhubungan Kota Magelang sebagian besar pengontrolan lampu lalu lintas menggunakan ATCS (*Automatic Control System*) dengan menempatkan sensor untuk deteksi logam untuk kendaraan, namun di beberapa tempat sudah menggunakan kontrol lampu lalu lintas dengan kamera CCTV yang lebih efektif selain itu juga memperoleh data lapangan lalu lintas secara *realtime*. Sistem kontrol lampu lalu lintas Dinas Perhubungan Kota Magelang dirancang setiap setahun sekali dengan melakukan survei kepadatan arus lalu lintas dengan membedakan golongan kendaraan bertipe tertentu yang umumnya golongan kendaraan roda dua dan roda empat sebagai acuan umum kendaraan bermotor. Survei kepadatan arus lalu lintas masih dilakukan secara manual dengan pengambilan data di lapangan menghitung jumlah kendaraan yang melewati ruas jalan yang padat. Survei yang dilakukan manual menggunakan bantuan perangkat penghitung sederhana dimana *surveyor* akan menklik perangkat bila kendaraan lewat, hal tersebut menyebabkan sering terjadinya *human error* karena operator ataupun *surveyor* memiliki keterbatasan dalam penghilatan dan kemampuan berpikir yang umumnya disebabkan faktor kelelahan.

Beberapa daerah di Indonesia sudah melakukan penerapan teknologi kecerdasan buatan yang ditanamkan pada sistem pengaturan lampu lalu lintas dengan melakukan pengolahan citra digital dalam upaya membantu mengatasi kepadatan lalu lintas. Metode pengolahan citra digital digunakan dalam melakukan pengenalan objek kendaraan untuk menentukan kepadatan pada ruas jalan yang terlihat oleh visual kamera CCTV. Metode pengolahan citra digital yang menerapkan kecerdasan buatan menghasilkan akurasi pembacaan objek kendaraan yang lebih baik namun ketika diterapkan dengan perangkat kamera CCTV hasilnya terdapat beberapa kendaraan yang tidak dapat diklasifikasikan (Harahap dkk., 2019).

Irawanto, dkk (2022) telah melakukan penelitian yang membuat sistem penggolongan kendaraan pada jalan tol menggunakan metode *Pin Hole*. Penelitian

tersebut menerapkan kemampuan pengolahan citra digital dengan memanfaatkan kamera CCTV kemudian membuat sistem kecerdasan buatan dengan metode Pin Hole. Sistem menghasilkan akurasi yang tinggi namun dengan golongan klasifikasi kendaraan yang cukup banyak, maka diperlukan perangkat yang lebih besar untuk mengelola sistem klasifikasi dan data sampel gambar kendaraan yang lebih banyak Sampel data gambar yang sedikit menyebabkan pembacaan sistem yang tidak sesuai. Sistem yang tidak sesuai ini berhubungan dengan kondisi dimana sistem kecerdasan buatan memiliki akurasi yang tinggi namun ketika sistem ditanamkan pada suatu aplikasi seperti kamera CCTV lampu lalu lintas, sistem tidak dapat membaca atau mengenal objek kendaraan dengan baik dan tepat sehingga menyebabkan kesalahan membaca objek kendaraan.

Budianto, dkk (2018) telah melakukan penelitian dalam melakukan klasifikasi berdasarkan plat nomor kendaraan bermotor dengan membandingkan dua metode berbeda metode *K-Nearest Neighbor* (KNN) dan *Support Vector Machine* (SVM). Hasil yang didapat menunjukkan metode KNN menghasilkan akurasi sebesar 80% dan SVM menghasilkan akuasi sebesar 95%. Penelitian tersebut membahas analisis dari dua metode berbeda dengan akurasi yang cukup tinggi namun penerapannya banyak menghasilkan kesalahan pembacaan objek kendaraan.

Taufiq, dkk (2018) telah melakukan penelitian mengenai klasifikasi berdasar tanda nomor kendaraan bermotor menggunakan *Convolutional Neural Network* menghasilkan akurasi sebesar 99%. Kenyataannya hasil akurasi sistem berbeda tergantung faktor yang mempengaruhi dan parameter yang digunakan seperti resolusi gambar kamera atau video dan sampel gambar yang sedikit ketika membuat sistem dengan CNN.

Harani, dkk (2019) telah melakukan penelitian untuk klasifikasi berdasar nomor plat kendaraan menggunakan *Convolutional Neural Network* (CNN) yang menggunakannya untuk keperluan *image classification*, dengan alasan memiliki akurasi yang relatif tinggi dan menghasilkan pengenalan objek yang signifikan. Namun pada kenyataannya terdapat faktor yang mengurangi pengenalan kendaraan seperti pencahayaan yang berubah mempengaruhi nilai piksel tiap frame,

pengaturan sudut penangkapan gambar yang berperan besar pada pengenalan objek dan pengaturan parameter metode *Convolutional Neural Network* (CNN).

Al Okaishi, dkk (2019) telah melakukan penelitian mengenai sistem survei lalu lintas yang secara *real-time* dapat melakukan klasifikasi kendaraan menggunakan *Convolutional Neural Network* (CNN). Berdasarkan penelitian tersebut terdapat objek kendaraan golongan mobill, motor dan truk yang memberikan akurasi untuk masing-masing kendaraan diatas 85% menggunakan pengolahan citra CNN. Menurut Al Okaishi, dkk, terdapat beberapa faktor terjadi saat sistem tidak dapat menklasifikasikan beberapa objek kendaraan yang tertangkap kamera CCTV diantaranya resolusi kamera CCTV yang kurang baik dan data gambar kendaraan masing-masing golongan yang tidak seimbang dalam membuat model pengolahan citra.

Berdasarkan kajian diatas, untuk mempersiapkan dan menjawab kebutuhan maka diperlukan suatu sistem yang mampu memonitoring kendaraan pada ruas jalan yang secara otomatis menklasifikasikan dan menghitung jumlah kendaraan untuk menggantikan proses manual pencatatan jumlah kendaraan yang digunakan sebagai data perencanaan kapasitas kepadatan arus lalu lintas melalui pengolahan citra digital kamera cctv lampu lalu lintas dalam upaya mencegah kepadatan arus lalu lintas.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan dapat ditarik beberapa rumusan masalah diantaranya pengawasan kamera CCTV lampu lintas dalam melakukan pengelolaan lampu lalu lintas dan pengambilan data jumlah kendaraan oleh surveyor untuk kebutuhan survei arus kepadatan lalu lintas oleh Dinas Perhubungan Kota Magelang masih menggunakan cara manual dimana manusia memiliki keterbatasan terutama dalam hal konsentrasi. Sering terjadinya *human error* dalam pengambilan keputusan dalam menghitung jumlah kendaraan. Sehingga dibutuhkan sistem yang secara optimal mampu menklasifikasikan objek kendaraan untuk golongan roda dua dan roda empat serta mencatat jumlahnya pada ruas jalan tertentu melalui hasil tangkapan video kamera CCTV lampu lalu lintas.

Berdasarkan persoalan yang telah dijelaskan, maka akan dilakukan penelitian tentang perancangan sistem *machine learning* untuk klasifikasi kendaraan pada persimpangan *traffic light*.

1.3 Tujuan Penelitian

Melalui rumusan masalah yang dipaparkan permasalahan yang menjadi topik bahasan pada penelitian ini merujuk pada kemampuan pengawas kamera CCTV lampu lalu lintas dan surveyor dalam menghitung jumlah kendaraan. Sehingga penelitian ini bertujuan yaitu dapat membuat sistem klasifikasi kendaraan pada persimpangan *traffic light* menggunakan sistem kecerdasan buatan yang secara efektif dan efisien mampu menklasifikasikan objek kendaraan pada golongan roda dua dan roda empat. Kemudian mencatat jumlahnya untuk memudahkan operator pengawas kamera CCTV lampu lalu lintas dalam melakukan pemantauan lampu lalu lintas serta pencatatan data jumlah kendaraan yang digunakan dalam melakukan pengumpulan data kepadatam arus lalu lintas dan perencanaan kapasitas beban jalan.

1.4 Manfaat Penelitian

Manfaat yang diharapkan pada penelitian ini yaitu:

a. Penelitian ini diharapkan mampu menghasilkan implementasi sistem pakar untuk klasifikasi kendaraan dengan akurasi yang lebih baik daripada sistem klasifikasi kendaraan yang telah ada berdasarkan penelitian terdahulu.
b. Penelitian ini diharapkan dapat menambah wawasan dan pengetahuan mengenai penerapan layer algoritma kecerdasan buatan yang efektif dan optimal pada sistem klasifikasi kendaraan pada persimpangan *traffic light*.

1.5 Batasan Masalah

Batasan masalah pada penelitian ini yaitu:

a. Sistem klasifikasi pada penelitian ini dibuat menggunakan salah satu algoritma *machine learning* untuk *image processing* yaitu *Convolutional Neural Network*.
b. Kelas kendaraan yang diklasifikasi terdiri dari 2 kelas yaitu kendaraan roda dua dan kendaraan roda empat. Data gambar yang digunakan dalam melakukan

model machine learning menggunakan yang diambil dari *public repository* dan tangkapan gambar melalui video kamera CCTV lampu lalu lintas yang diambil dari Dinas Perhubungan Kota Magelang, dan

c. Pengolahan citra dilakukan dengan melakukan pengolahan pada citra berjenis HSV (*Hue*, *Saturation*, *Value*)

1.6 Sistematika Penulisan

Sistematika penulisan pada penelitian ini terdiri dari 5 bab, seperti yang dapat dilihat pada uraian berikut ini:

BAB I PENDAHULUAN

Bab ini membahas mengenai latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II DASAR TEORI

Bab ini menjelaskan tentang ldanasan teori yang dipakai penulis dalam menyusun tugas akhir ini melalui berbagai referensi yang berkaitan dengan penelitian.

BAB III METODOLOGI PENELITIAN

Bab menguraikan tahapan yang akan dikerjakan untuk menyelesaikan permasalahan yang ada sesuai dengan judul penelitian. Metode yang digunakan disampaikan dalam bentuk diagram alir (*flowchart*) atau bentuk lain.

BAB IV HASIL DAN PEMBAHASAN

Bab ini menguraikan dan menjelaskan mengenai hasil pengujian sistem yang telah dibuat serta menganalisisnya.

BAB V PENUTUP

Bab ini mengemukakan kesimpulan dari penelitian yang dilakukan dan saran untuk penelitian selanjutnya.

DAFTAR PUSTAKA

LAMPIRAN

BAB II
DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian yang akan dilakukan memerlukan tinjauan pustaka sebagai bahan acuan dari hasil penelitian – peneltian sebelumnya yang telah ada, sangat penting di lakukan agar terhindar dari penjiplakan atau duplikasi dari penelitian terdahulu yang bertujuan sebagai bahan untuk kontribusi penelitian bagi penulis agar penelitian mengenai topik ini terus berkembang. Berikur beberapa pemAParan tentang penelitian sebelumnya yang pernah di lakukan sebelumnya mengenai data dan metode yang digunakan.

Setiawan (2020) melakukan implementasi untuk melakukan pengenalan warna kendaraan menggunakan metode *convolutional neural network*. Menggunakan 2000 data gambar yang terbagi dalam 4 kelas yaitu warna puthin, abu-abu, hitam, dan silver. Data gambar dipisahkan menjadi dua bagian yaitu data *training* sebanyak 1600 data gambar dan data *validation* sebanyak 400 data gambar. Gambar dilakukan *preprocessing* awal melakukan *resize* gambar 100x100 piksel kemudian memasukkan input data gambar ke dalam model CNN yang disusun menggunakan 4 layer konvolusi ukuran kernel 3x3, *maxpooling* ukuran kernel 2x2 dan menggunakan *adam optimizer*. Model CNN atau layer konvolusi yang digunakan berukuran 100x100x3 dengan angka 3 yang berarti memiliki 3 channel berupa *Red*, *Green* dan *Blue* (RGB). Jumlah total layer konvolusi yang digunakan adalah enam lapisan dengan ukuran kernel sama serta setiap dua layer konvolusi mempunyai jumlah filter yang sama. Tahap terakhir melakukan proses *flatten* atau proses mentransformasikan *feature mAP* hasil layer *pooling* ke dalam bentuk *vector*. Proses training model menghasilkan akurasi klasifikasi warna kendaraan sebesar 73.00%. Sedangkan proses pengujian model menggunakan input data gambar sejumlah 360 yang masing – masing dari kelas berisi 90 data gambar yang menghasilkan akurasi 76.94%.

Nurolan (2020) melakukan penelitian untuk mengklasifikasikan jenis kendaraan berdasarkan jenis kendaraan roda empat pada gerbang tol yaitu Mobil MVP, Minibus, Bus da Mobil Pickup. Jumlah data gambar yang digunakan adalah

sebanyak 200 gambar yang terbagi menjadi lima kelas dengan empat jenis kelas diatas dan satu kelas data gambar campuran. Data gambar dibagi menjadi lima kelas untuk data training sejumlah 200 data gambar dan data testing sejumlah 25 gambar. Convolutional neural network merupakan metode yang digunakan dalam penlitian ini, dengan menggunakan layer konvolusi, fungsi aktivasi *Relu*, layer *pooling* dan layer *fully connected*. Hasil dari klasifikasi data berupa file berekstensi XML sehingga diubah ke bentuk file berekstensi CSV dimaksudkan agar dapat di Generate TFRecord yaitu perekam proses pelatihan data. Framework yang digunakan adalah Tensorflow yang menghasilkan checkpoint secara otomatis berbentuk graph tensor. Penelitian menggunakan CNN dengan jumlah iterasi sebesar 10.000 proses training database dengan waktu rata – rata 2 – 2.5 detik untuk sistem membaca objek klasifikasi. Terdapat 8 pengujian yang dilakukan menggunakan model yang sudah di training dan hasil terbaik adalah pengujian 8 yang mana menghasilkan presentase akurasi 100% pada setiap objeknya. Sedangkan rata – rata akurasi total yang dihasilkan sebesar 81.94% pada seluruh pengujian.

Fadlia dan Kosasih (2020) dalam penelitiannya untuk membuat sistem yang dapat menklasifikasikan jenis kendaraan berdasarkan jumlah roda kendaraan yaitu kendaraan roda dua dan kendaraan roda empat menggunakan metode *convolutional neural network*. Data gambar yang digunakan berupa gambar kendaraan yang diambil secara acak melalui internet berdasarkan pilihan kendaraan roda yang ditentukan menggunakan aplikasi yaitu Fatkun Batch Downloader. Data gambar yang didapatkan berjumlah 90 citra gambar untuk data latih dan 30 citra gambar untuk data uji. Gambar terdiri dari 40 gambar mobil, 40 gambar motor dan 40 gambar sepeda. Model yang dibuat menggunakan beberapa jenis *layer* (lapisan) yaitu lapisan konvolusi 2D, lapisan *pooling*, lapisan *dropout*, lapisan *flatten* dan lapisan *dense*. Penelitian yang dilakukan menggunakan proses konvolusi sebanyak 4 kali dengan menggunakan fungsi aktivasi *ReLu* (*Rectified Linear Unit*) yang memberikan tahap pelatihan lebih cepat. Melalui model tersebut dihasilkan akurasi sebesar 94.44% untuk model data latih dan akurasi sebesar 73.33% untuk model data uji.

Harani, dkk (2019) melakukan penelitian untuk membuat sistem yang mampu mendeteksi objek dan melakukan pengenalan karakter dari plat nomor kendaraan Indonesia yang dilakukan menggunakan metode *convolutional neural network*. Objek yang dilakukan deteksi dan pengenalan adalah berupa karakter yang ada pada plat nomo kendaraan, kelas yang digunakan untuk klasifikasi berjumlah 37 kelas yang terdiri dari huruf A hingga Z, angka 0 hingga 9 serta objek *plate*. Jumlah data yang digunakan adalah sebanyak 1615 gambar data plat nomor yang dibagi dalam data training dan data testing. Proses training data memerluka waktu sekitar 6 jam dengan training sebanyak 100.000 steps. Sehingga diketahui model menghasilkan average precision (AP) untuk kelas D sebesar 34%. Sedangkan untuk cross validation atau juga model data uji menghasilkan nilai loss sebesar 0.5278 dengan memerlukan waktu uji model sekitar 4.5 jam 10 detik.

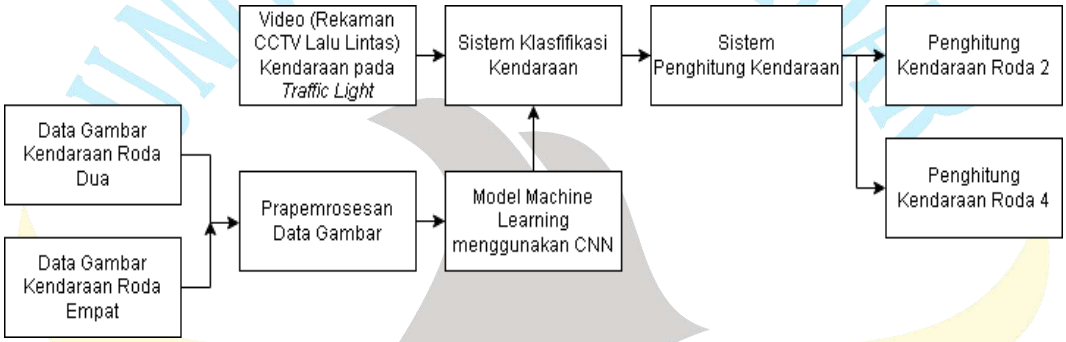
Ma'ali and Hendriyawan A (2019) telah melakukan penelitian mengenai rancang bangun sistem pengendali lampu lalu lintas menggunakan pengolahan citra digital kendaraan dengan metode Faster R-CNN, yang memberikan akurasi pembacaan objek kendaraan sebesar 97,027%. Namun akurasi tersebut masih belum optimal ditambah proses compute model pengolahan citra digital yang begitu lambat. Menurut Ma'ali, hal tersebut terjadi dikarenakan kurangnya dataset gambar berdasarkan golongan kendaraan yang terdiri dari kendaraan mobil, motor, bus, truk. Kebutuhan tersebut juga diimbangi dengan kebutuhan perangkat komputer pengelolaan untuk pengolahan citra digital seperti GPU (*Graphic Processing Unit*). Dewi (2018) menggunakan metode convolutional neural network untuk penelitiannya mengenai klasifikasi gambar meja dan kursi motif ukiran Jepara. Menggunakan framework Tensorflow untuk mengolah datset yang terdiri dari motif meja dan kursi ukiran Jepara yang berjumlah 500 gambar. Gambar dibagi menjadi dua untuk data training sejumlah 470 data gambar dan data testing sejumlah 30 data gambar. Proses training menggunakan jumlah iterasi sebanyak 250.00 step menghasilkan tingkat akurasi sebesar 98%.

2.2 Dasar Teori

Penelitian yang dilakukan menggunakan beberapa referensi teori untuk mendukung hasil penelitian secara maksimal . Berikut beberapa pengetahuan

pendukung akan digunakan dalam penelitian. Teori yang digunakan menjadi landasan dalam penelitian perancangan sistem *machine learning* untuk klasifikasi kendaraan pada persimpangan *traffic light*.

Blok diagram pada gambar dibawah ini menjelaskan gambaran penelitian tentang sistem yang dirancang. Perancangan sistem *machine learning* untuk klasifikasi kendaraan pada persimpangan *traffic light* memerlukan sistem diagram sebagai alur proses operasi sistem pada penelitian. Diagram blok sistem secara keseluruhan yang menunjang landasan teori pada penelitian ini yang ditunjukkan pada Gambar2.1,



Gambar 2. 1 Blok Diagram Sistem

Blok diagram menjelaskan kebutuhan komponen penelitian perancangan sistem *machine learning* untuk klasifikasi kendaraan pada persimpangan *trafffic ligh*t. Komponen pendukung penelitian pada blok diagram diatas dijelaskan sebagai berikut:

2.2.1 Pengolahan Citra Digital

Citra atau dapat disebut sebagai gambar yaitu sebuah gambar bidang dua dimensi yang tersusun atas banyak piksel yang adalah bagian terkecil daripada citra. Secara umum, citra terbentuk atas kumpulan kotak persegi empat yang teratur sehingga memiliki jarak antar piksel yang sama pada seluruh bagian citra. Citra yang sebagai keluaran daripada sebuah sistem perekaman data memiliki sifat diantaranya (Dey, 2018):

a. optik yang dapat berupa foto,

b. analog yang dapat berupa sinyal video seperti gambar yang muncul di layar *handphone* maupun monitor televisi, dan
c. digital yang dapat disimpan pada sebuah media penyimpanan magnetik.

Citra juga bisa dikelompokkan dua diantaranya citra diam serta citra bergerak. Citra bergerak merupakan kumpulan citra yang ditampilkan secara beruntun (*sekuensial*), sehingga memberikan kesan sebagai gambar yang bergerak. Kumpulan citra yang tergabung menjadi satu rangkaian disebut sebagai *frame* (Sonka et al., 2014). Misalkan pada sebuah video atau film yang tampil pada layar lebar yang hakikatnya tersusun atas ratusan bahkan sampai ribuan *frame*.

Penjelasan citra diatas memberikan definisi jelas mengenai citra digital. Sehingga citra digital dapat diartikan sebagai suatu matriks yang mana indeks baris serta kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya disebut dengan elemen gambar atau juga piksel yang memiliki nilai tingkat derajat keabuan pada suatu titik tersebut. Citra digital memiliki bentuk matriks yang memiliki ukuran M (baris/tinggi) x N (kolom/lebar) dan tersusun seperti dibawah ini:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,m-1) \\ f(1,0) & f(1,1) & \cdots & f(1,m-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix} \text{.....(2.1)}$$

Suatu citra $f(x,y)$ memiliki fungsi matematis dapat dituliskan berikut:

$0 \leq x \leq M-1$

$0 \leq y \leq N-1$

$0 \leq f(x,y) \leq G-1$ .....(2.2)

Melalui fungsi matematis diatas simbol M, N dan G diartikan sebagai berikut:

M = banyak baris pada *array* citra

N = banyak kolom pada *array* citra

G = banyak skala keabuan (*graylevel*)

Suatu proses mengubah sebuah citra analog ke citra digital disebut sebagai digitasi. Digitasi sendiri merupakan proses mengubah gambar, teks, maupun suara dari benda yang nampak ke sebuah data elektronik serta dapat disimpan dan diproses untuk keperluan lain. Citra digital pada komputer akan dibaca sebagai

bentuk grid atau elemen piksel dalam bentuk matriks 2 dimensi. Setiap kumpulan piksel tersebut mempunyai nilai angka yang memiliki representasi *channel* warna. Angka di setiap piksel disimpan secara berurutan oleh komputer sehingga seringkali nilainya dikurangi guna proses kompresi ataupun pengolahan citra tertentu.

Citra digital dapat dibagi menjadi beberapa jenis diantaranya:

1. Citra Biner

   Citra biner (*binary*) merupakan citra yang hanya memiliki dua nilai derajat keabuan yaitu hitam dan putih. Piksel dari warna tersebut memiliki nilai 1 yang berarti warna putih dan nilai 0 yang memiliki warna hitam.

2. Citra Keabuan

   Citra keabuan (*grayscale*) yaitu citra yang pada setiap piksel mengandung satu lapisan dimana memiliki nilai intensitas dengan nilai 0 (hitam) – 255 (putih).

3. Citra Warna

   Citra warna merupakan citra digital yang mempunya informasi warna di setiap pikselnya. Sistem warna pada citra warna terdapat beberapa macam diantaranya RGB, CMYK, HSV, dll. RGB merupakan model warna yang umum digunakan terdiri atas tiga warna dasar merah, hijau, dan biru yang digabungkan dalam membentuk sebuah susunan warna yang lebih luas memiliki rentang nilai 0 – 255.

Secara keseluruhan pengolahan citra digital dapat diartikan sebagai proses pengolahan serta analisis suatu citra digital yang banyak melibatkan persepsi visual. Proses pengolahan ini memiliki ciri data masukan serta informasi keluaran berbentuk sebuah citra digital. Teknik pengolahan citra digital menggunakan komputer dalam melakukan digitalisasi pola bayangan maupun warna pada gambar yang ada.

2.2.2 *Artificial Intelligence* (AI)

Artficial Intelligence atau kecerdasan buatan yaitu salah satu cabang ilmu pengetahuan pada bidang komputer yang memiliki hubungan dengan pemanfaatan

mesin dalam upaya memecahkan persoalan yang kompleks dengan cara yang terstruktur. Hal tersebut umumnya dilakukan dengan mengikuti karakteristik dana analogi berpikir dari kecerdasan manusia serta menerapkannya dengan algoritma yang diketahui oleh komputer (Budiharto, 2018).

Artificial Intelligence ini merupakan sebuah konsep besar yang memiliki banyak algoritma atau metode yang terbentuk melalui pola yang diberikan ke suatu sistem. Komputer belajar mengenali pola tersebut kemudian membentuk sebuah struktur yang dapat menyelesaikan sebuah permasalahan.

2.2.3 *Computer Vision*

Computer Vision adalah salah satu bidang keilmuan pada AI yang mengakuisisi, mengolah, memahami dan mengambil informasi atau keputusan dari suatu image atau streaming video (Davies, 2017). *Computer Vision* menjadikan sebuah komputer berperilaku layaknya manusia karena mendekati kemampuan manusia dala menggunakan salah satu indera pengelihatan yaitu menangkap informasi dalam bentuk visual. *Computer Vision* dapat dirumuskan sebagai sebuah gabungan sistem dari kamera, komputer dan *pattern recognition*. *Computer Vision* membuat komputer atau sistem bertindak layaknya manusia dengan memiliki indra penglihatan atau juga diartikan mampu menerima informasi secara visual. Kemampuan komputer dalam menerima informasi secara visual ini terdapat beberapa macam diantaranya

- *Object Detection* : Kemampuan untuk mengenali suatu objek yang terlihat pada citra dan mengetahui batas objek yang dikenali melalui proses matematis berkaitan dengan jarak antar data.
- *Recognition* : Kemampuan menempatkan label kelas pada suatu objek.
- *Interprating Motion* : Kemampuan penafsiran gerakan pada citra bergerak.
- *3D Inference* : Kemampuan penafsiran suatu citra 3D dari 2D yang nampak.

- *Description* : Kemampuan menugaskan properti pada suatu objek.

Object detection adalah salah implementasi yang dilakukan pada penelitian ini. *Object detection* memiliki arti suau kemampuan dalam menentukan sebuah keberadaan objek dan ruang yang melingkupi serta lokasi pada suatu gambar. Hal ini dapat terjadi karena terdapat perlakuan dimana dikenali dua objek kelas, dimana satu kelas dikenali sebagai objek dan kelas lainnya dikenali sebagai non-objek. Deteksi objek dibagi lagi menjadi dua yaitu *soft detection* yang hanya mendeteksi adanya objek saja dan *hard detection* yang mendeteksi objek beserta lokasi daripada objek. Komputer dalam mengenali atau mendeteksi objek melakukan beberapa tahapan diantaranya sebagai berikut:

a. Ekstraksi Fitur

Ekstraksi fitur adalah suatu proses indeksasi pada database citra beserta isi dari citra. Matematisnya dapat dijelaskan sebagai *encode* dari venktor n dimensi yang juga dikenal dengan vektor fitur. Vektor fitur memiliki komponen yang dihitung menggunakan pemrosesan fitur dan teknik analisis serta dipergunakan dalam membandingkan citra satu dengan lain. Terdapat 3 jenis ektraksi fitur yaitu *low level*, *middle level* dan *high level*. Fitur *low level* yaitu ekstraksi fitur berdasar isi visual contoh warna dan tekstur. Fitur *Middle Level* yaitu ekstraksi fitur berdasar wilayah citra yang dilakukan dengan melakukan segmentasi_citra. Sedangkan fitur *high level* yaitu ekstraksi fitur berdasar informasi semantik yang terdapat pada citra. Macam – macam fitur pada ekstraksi fitur citra diantaranya sebagai berikut:

1) Warna

Warna sebagai salah satu fitur visual yang dipergunakan pada *Content Based Image Retrieval* (CBIR). Fitur warna sangat baik apabila dipergunakan dalam pengolahan citra karena mempunyai hubungan kuat terkait objek pada sebuah citra, yang juga melatarbelakangi *background substraction*, penskalaan objek, orientasi, perspektif serta ukuran.

2) Bentuk

Bentuk juga fitur yang terdapat pada suatu citra yang sangat esensial dalam segmentasi citra karena mampu mendeteksi objek maupun memberi batas wilayah objek pada citra dengan background. Pengolahan citra yang digunakan dalam menentukan fitur bentuk yaitu deteksi tepi, threshold, segmentasi dan perhitungan jarak antar data seperti *mean*, *median* serta standard deviasi pada setiap lokal citra.

3) Tekstur

Fitur lainnya yang diektraksi adalah tekstur. Tekstur yaitu fitur pada citra yang berkaitan dengan tingkat kekasaran (*roughnessi*), granularitas (*granulation*) dan keteraturan (*regularity*) susunan yang terdapat pada piksel. Aspek fitur tekstur dapat digunakan dalam dasar daripada klasifikasi, segmantasi serta interpretasi citra.

b. Segmentation

Segmentation atau segmentasi pada citra adalah salah satu langkah yang penting dalam analisis pada pengolahan data citra, yang memiliki tujuan untuk membagi citra gambar ke beberapa bagian. Tiap bagian yang dimaksud memiliki korelasi kuat dengan objek atau wilayah dari gambar aslinya. Bagian yang dimaksud umumnya adalah *foreground* dan *background* atau objek individu pada gambar. Segmentasi wilayah pada gambar dapat dibangun menggunakan beberapa fitur seperti warna, tepi ataupun *neighbor similarity*. Segmentasi dilakukan pada penelitian ini yang termasuk dalam bagian proses pengolahan citra yang bertujuan untuk untuk menemukan objek kendaraan pada gambar tangkapan kamera CCTV lampu lalu lintas.

c. Pattern Recognition

Pattern recognition diartikan juga pengenalan pola merupakan suatu sistem yang mempelajari berbagai teknik matematis diantaranya statistika, jaringan syaraf tiruan, *support vector machine,* dan sejenisnya untuk melakukan klasifikasi pola yang berbeda. Input dari *pattern recognition* dapat berupa data gambar. Teknik ini banyak digunakan untuk melakukan pengenalan pola pada *computer vision*. Pengenalan pola mempunyai pengertian bidang studi yang dapat melakukan proses analisis gambar yang

bentuk masukan dapat berupa gambar itu sendiri ataupun dapat berupa citra digital serta bentuk keluarannya adalah label atau deskripsi (Davies, 2017). Tujuan daripada pengenalan pola yaitu komputer memiliki kemampuan yang sama seperti manusia dalam melakukan pengenalan suatu objek berdasarkan pola tertentu.

2.2.4 *Machine Learning*

Machine Learning merupakan sebuah tipe dari *artificial intelligence* yang memberikan komputer kemampuan untuk belajar tanpa secara eksplisit diprogram (Müller and Guido, 2016). Sederhananya *machine learning* berfokus pada program program komputer yang mampu mengajarkan diri sendiri, sehingga memungkinkan komputer belajar serta menyelesaikan tugasnya sendiri tanpa adanya instruksi langsung dari pengguna. *Machine learning* dapat diibaratkan sebagai seorang manusia yang belajar seperti manusia melalui banyaknya pengalaman. Sehingga dapat dikatakan, bahwasanya *machine learning* merupakan teknik-teknik untuk belajar dalam mengambil suatu keputusan melalui berbagai data yang ada (Suyanto, 2017). *Machine learning* memiliki hubungan dengan *computational statistics* yang berfokus di sebuah prediksi atau pengambilan keputusan berdasar penggunaan komputer. Implementasi dari machine learning diantaranya *image processing, text analysis*, *finance*, *search and recommendation engine* serta *speech understanding*.

Machine learning memiliki tiga kategori utama diantaranya (Müller and Guido, 2016):

a. *Supervised Learning*

Supervised learning merupakan kategori *machine learning* yang mempelajari sebuah kumpulan data yang dilengkapi label atau kelas untuk menunjukkan klasifikasi data yang ada. Pembelajaran akan menghasilkan sebuah model prediksi dari data yang telah memiliki label.

b. *Unsupervised Learning*

Unsupervised learning merupakan kategori *machine learning* yang mempelajari kumpulan daya tanpa adanya label atau kelas sehingga mencari

struktur data yang ada kemudian melakukan pengelompokan berdasar informasi yang dimiliki.

c. *Reinforcement Learning*

Reinforcement learning adalah kategori *machine learning* yang melakukan pembelajaran terhadap apa yang akan dilakukan dengan memetakan situasi dalam suatu aksi untuk mendapatkan *reward* yang maksimal atau justru sebaliknya *punishment.*

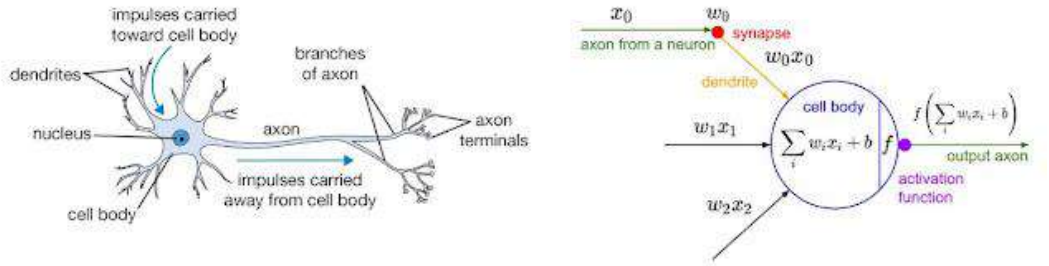
2.2.5 *Deep Learning*

Deep learning merupakan sebuah pembaruan konsep mendalam dari algoritma *Artificial Neural Network* (ANN) dengan banyak lapisan atau *layers* pada susunan lapisan *neuron* nya. Secara definisi menurut Goodfellow, dkk (2016), deep learning adalah sebuah pendekatan dalam penyelesaian masalah pada sistem pembelajaran komputer yang menggunakan konsep hierarki. Konsep hierarki inilah yang membuat komputer memiliki kemampuan mempelajari konsep yang lebih kompleks dengan menggabungkan konsep – konsep yang lebih sederhana seperti lapisan neuron pada algoritma artificial neural network. Lapisan *neuron* sederhana terdiri dari beberapa *layer* saja pada ANN kemudian dengan konsep *deep learning* lapisan tersebut semakin banyak untuk menyelesaikan permasalahan yang lebih kompleks.

2.2.6 *Artificial Neural Network*

Artificial neural network merupakan algoritma kecerdasan buatan yang membentuk suatu jaringan melalui model sistem saraf otak manusia (disebut *neuron*) dalam melakukan sebuah tugas pengenalan pola, khususnya klasifikasi (Suyanto, 2017). Jaringan ini meniru cara kerja daripada jaringan syaraf yang ada pada tubuh manusia, yang mana dibentuk dari node-node yang saling terhubung satu sama lain. Node yang dimaksud terhubung melalui sebuah *link* yang dinamakan weight atau bobot. *Neural network* dirancang pertama kalinya oleh Warren McCulloch dan Walter Pitts tahun 1943 yang pada saat itu dikenal McCulloch-Pitts neurons mengenai dua neuron aktif secara bersamaan kemudian kekuatan yang timbul terhubung antara *neuron* yang seharusnya bertambah. Setelah

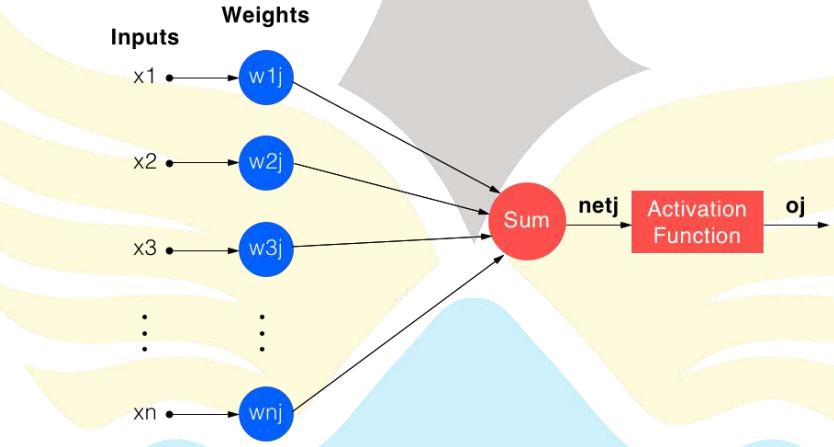
itu tahun 1957, Frank Rosenblatt memperkenalkan dan mengembangkan kumpulan besar jaringan syaraf tiruan yang dinamakan *perceptron*(Müller and Guido, 2016).



Sumber: (Dewi, 2018)

Gambar 2. 2 Ilustrasi Jaringan Syaraf secara Biologis dan Tiruan beserta Model Matematisnya

Neuron dapat diartikan sebagai sebagai unit yang mengolah informasi yang merupakan dasar daripada proses suatu jaringan syaraf tiruan. Detail model operasi yang dikerjakan neuron dapat terlihat pada gambar dibawah ini.



Sumber: (Fadlia and Kosasih, 2020)

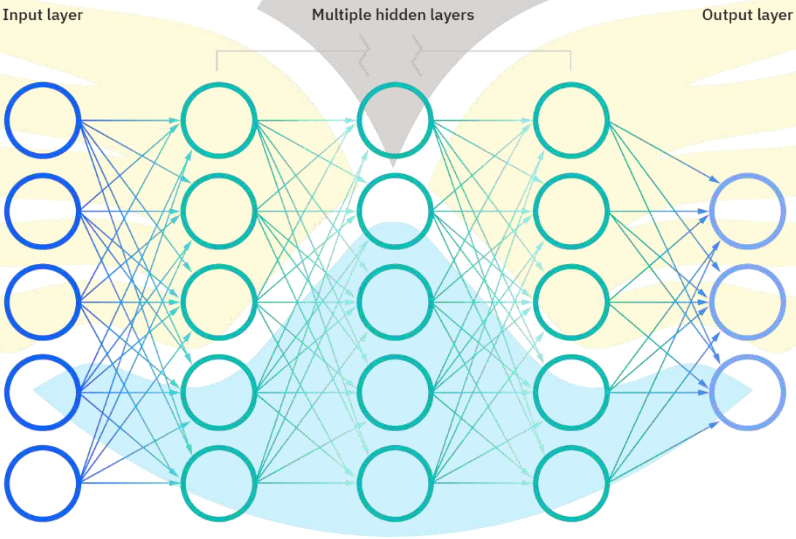
Gambar 2. 3 Ilustrasi Jaringan Syaraf Tiruan

Gambar diatas menunjukkan beberapa bagian dasar dari suatu syaraf tiruan diantaranya:

1. Satu set sebuah sinapsis atau penghubung yang setiap set digolongkan oleh bobot.
2. Satu nilai sum yang memiliki peran menambah setiap sinyal input yang masuk. Melalui pertimbangan dari bobot sinapsis tiap *neuron*.

3. Satu fungsi aktivasi yang memiliki peran dalam membatasi nilai amplitudo output dari *neuron*. Tujuan utamanya adalah untuk membatasi jarak amplitudo yang diperbolehkan sinyal *output* menjadi suatu angka yang terbatas.

Multilayer neural network merupakan suatu algoritma *neural network* yang terdiri atas banyak lapisan dimana multilayer ini yang menjadi suatu pengembangan konsep yang dinamakan *deep learning*. Multilayer *neural network* mempunyai karakteristik dengan banyak lapisan yang mana tiap node pada satu lapisan terhubung dengan tiap node pada *layer* berikutnya. Arsitektur umpan maju menggunakan metode *supervised learning* yang memiliki perbedaan pada susunannya yang terdiri dari satu atau lebih *hidden layer*. *Hidden layer* ini merupakan lapisan yang tidak nampak oleh *input* ataupun *output* pada jaringan tersebut. Peran dari *hidden layer* ini yaitu untuk melakukan intervensi antara *input* eksternal dan *output* dari jaringan melalui penambahan satu atau lebih *hidden layer*, jaringan yang memiliki *multilayer* dapat menghasilkan statistik tingkat tinggi dari input. *Multilayer neural network* dapat digambarkan sebagai berikut.



Sumber: (Fadlia and Kosasih, 2020)

Gambar 2. 4 Ilustrasi Multilayer Neural Network

Semakin banyak *layer* yang ada maka akan semakin kompleks juga sistemnya karena setiap sumber node pada input layer dari jaringan memiliki masing – masing elemen dari pola aktivasi, yang berarti sinyal *input* diterapkan ke

neuron – neuron pada layer berikutnya, begitu juga seterusnya hingga memenuhi semua *layer* yang ada. Secara umum model jaringan *multilayer* ini terdapat dua mekanisme kerja yaitu.

a. *Training* (Pelatihan)

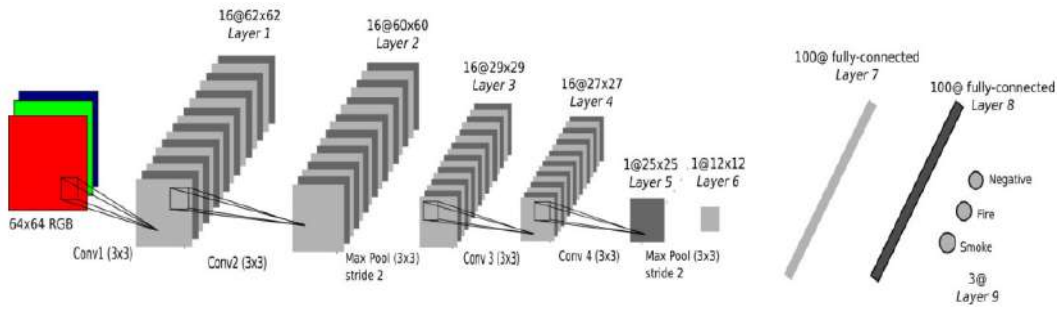
   Mekanisme *training* memberikan jaringan untuk dilatih sehingga menghasilkan data sesuai dengan target yang diharapkan melalui satu atau lebih pasangan data. Semakin lama waktu pelatihan yang diberikan maka kinerja jaringan juga semakin baik.

b. *Testing* (Pengujian)

   Mekanisme *testing*, suatu jaringan akan diuji untuk dapat mengenali sesuatu target dengan yang diharapkan melalui proses *training* yang sudah diberikan.

2.2.7 *Convolutional Neural Network*

Convolutional neural network atau juga dikenal ConvNet (singkatan dari *Convolutional Network*) merupakan model *deep learning* pertama yang merupakan bagian dari pengembangan algoritma *neural network* yang mana terdiri atas banyak layer yang awalnya memang diaplikasikan pada analisis citra (Alom et al., 2018). Nama *convolutional neural network* tersebut menandakan jaringan dibentuk menggunakan operasi matematika yang dinamakan konvolusi. Konvolusi merupakan sebuah operasi linear, sehingga convolutional neural network dapat diartikan juga sebagai neural network yang menggunakan konvolusi setidaknya pada salah satu lapisannya (LeCun et al., 2015). ConvNet merupakan model algoritma terbaik yang dikhususkan untuk menyelesaikan masalah yang berkaitan dengan citra seperti *recognition* dan *detection*.

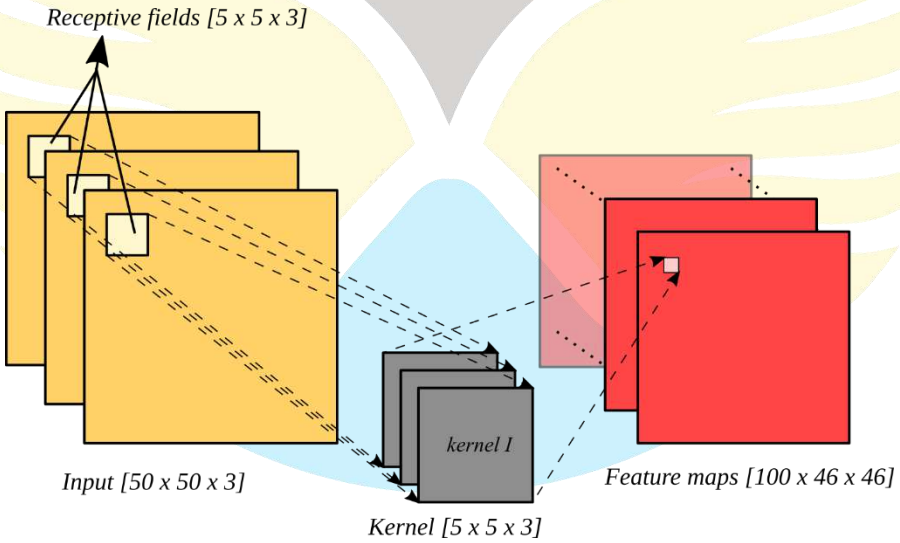


Sumber: (Harani et al., 2019)

Gambar 2. 5 Convolutional Neural Network

Sumber: https://mti.binus.ac.id/2019/07/11/deteksi-video-api-dan-asap-menggunakan-convolutional-neural-network/

Convolutional neural network (CNN) merupakan jenis algoritma *Deep Neural Network* dikarenakan kedalaman atau kompleksitas jaringan yang tinggi serta banyak diaplikasi pada pengelolaan yang berkaitan dengan citra atau gambar. *ConvNet* secara teknis merupakan arsitektur yang dapat di *training* serta terdiri atas beberapa tahap. *Input* dan *output* pada masing – masing tahap adalah beberapa array yang disebut dengan *feature mAP*. *Output* pada masing – masing tahap merupakan feature mAP hasil pengolahan yang berasa dari semua lokasi pada *input*.



Sumber: (Harahap et al., 2019)

Gambar 2. 6 Feauture MAP

2.2.8 *Python*

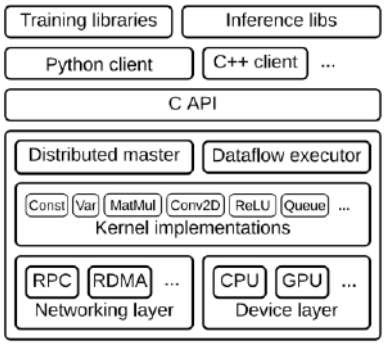
Python adala salah satu bahasa pemrogrmana yang sejak awal bertujuan untuk membangun sebuah *source code* yang mudah dibaca oleh para pengembang ataupun orang awam yang baru belajar. *Python* dikembangkan dengan berbagai *library* ataupun *framework* yang lengkap sehingga memungkinkan pada programmer untuk membangun sebuah aplikasi yang mutakhir menggunakan source code yang nampak sederhana. Hal ini memungkinkan *Python* bekerja pada *environment machine learning* dengan lebih mudah dan nyaman dalam membangun sebuah model (Müller and Guido, 2017).

2.2.9 *Tensorflow*

Tensorflow yaitu salah satu teknologi berupa *library software* yang dikembangkan oleh Tim Google Brain dalam organisasi penelitian Google Smart Machine, yang memiliki tujuam melakukan modeling machine learning dan penelitian jaringan syaraf tiruan. Tensorflow merupakan gabungan aljabar komputasi dengan teknik pengoptimalan kompilasi, mempermudah penghitungan banyak ekspresi matematis uang mana masalah utama adalah waktu yang dibutuhkan dalam melakukan perhitungan. Berikut adalah beberapa fitur utama Tensorflow diantaranya (Abadi et al., 2016):

a. Mendefinisikan, mengoptimalkan dan menghitung secara efisien ekspresi matematis yang melibatkan *array multidimension* (*tensors*),
b. pemrograma yang mendung artificial intelligence dan machine learning,
c. penggunaan GPU yang transparan, mengotomatisasi manajemen dan optimalisasi memori yang sama dan data yang digunakan. Tensorflow bisa menulis kode yang sama dan menjalankan baik di CPU dan GPU. Khususnya, Tensorflow akan mengetahui bagian perhitungan yang harus dipindah ke GPU, dan
d. skalabilitas komputasi yang tinggi di seluruh mesin dan kumpulan data yang besar.

Tensorflow memiliki banyak fitur yang membantu dalam proses *training*, *testing* dan *deploy* model machine learning. Kerangka kerja dari *Tensorflow* adalah sebagai berikut:



Sumber : (Abadi et al., 2016)

Gambar 2. 7 Arsitektur *Framework Tensorflow*

2.2.10 YOLOv3

YOLOv3 merupakan algoritma pengembangan convolutional neural network yang lebih spesifik dalam melakukan deteksi objek. Menggunakan single shot layer atau memfokuskan pada layer deteksi setiap objek dalam satu frame gambar. YOLOv3 memiliki kompleksitas yang rumit pada layer konvolusi karena sebanyak 23 layer digunakan dalam layer ini, namun terdapat layer YOLO yang memberikan fokus pada objek yang dipilih saja tidak keseluruhan gambar dilakukan proses pengolahan pengenalan citra. Sehingga algoritma YOLO memiliki kemampuan komputasi yang terbilang ringan dan cepat mampu melakukan training model dengan berbagai objek serta gambar ribuan sekalipun dalam training. Menggunakan YOLOv3 juga sangat mudah karena terdapat library darknet yang digunakan dengan mengimportnya ke dalam sistem komputer (Redmon and Farhadi, 2018). YOLOv3 pada penelitian ini akan digunakan untuk melakukan model *training machine learning* pada dataset gambar latih yang telah dikumpulkan. Model YOLO memiliki arsitektur jaringan yang sangat kompleks dengan jumlah total 23 *layer* yang teridiri dari 13 *layer convolutional*, 6 *layer maxpool*, 1 *layer router* 13, 1 *layer route* 19 8, 1 *layer* up-*sampling* dan 2 *layer* YOLO, dapat terlihat seperti gambar di bawah ini:

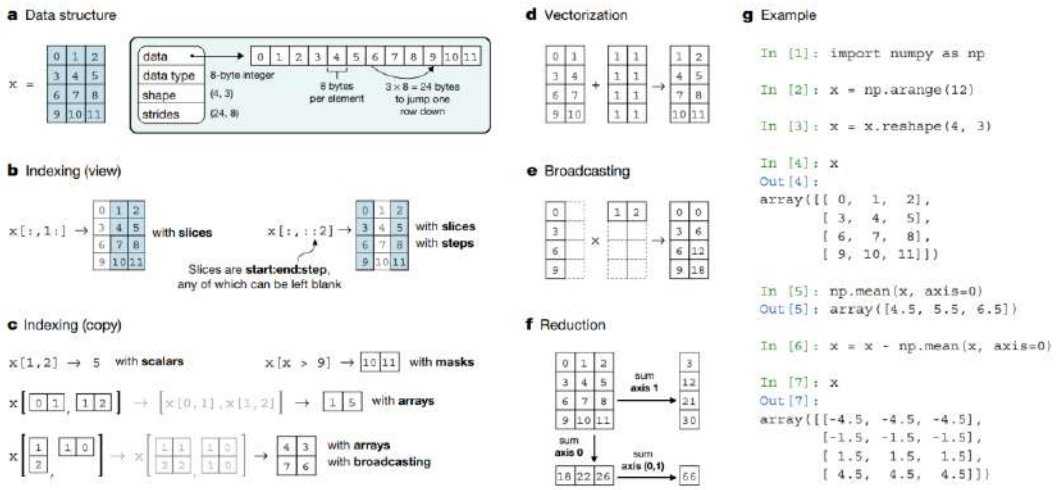
Layer	Type	Filters	Size/Stride	Input	Output
0	Convolutional	16	3 × 3/1	416 × 416 × 3	416 × 416 × 16
1	Maxpool		2 × 2/2	416 × 416 × 16	208 × 208 × 16
2	Convolutional	32	3 × 3/1	208 × 208 × 16	208 × 208 × 32
3	Maxpool		2 × 2/2	208 × 208 × 32	104 × 104 × 32
4	Convolutional	64	3 × 3/1	104 × 104 × 32	104 × 104 × 64
5	Maxpool		2 × 2/2	104 × 104 × 64	52 × 52 × 64
6	Convolutional	128	3 × 3/1	52 × 52 × 64	52 × 52 × 128
7	Maxpool		2 × 2/2	52 × 52 × 128	26 × 26 × 128
8	Convolutional	256	3 × 3/1	26 × 26 × 128	26 × 26 × 256
9	Maxpool		2 × 2/2	26 × 26 × 256	13 × 13 × 256
10	Convolutional	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
11	Maxpool		2 × 2/1	13 × 13 × 512	13 × 13 × 512
12	Convolutional	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024
13	Convolutional	256	1 × 1/1	13 × 13 × 1024	13 × 13 × 256
14	Convolutional	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
15	Convolutional	255	1 × 1/1	13 × 13 × 512	13 × 13 × 255
16	YOLO				
17	**Route 13**				
18	Convolutional	128	1 × 1/1	13 × 13 × 256	13 × 13 × 128
19	Up-sampling		2 × 2/1	13 × 13 × 128	26 × 26 × 128
20	**Route 19 8**				
21	Convolutional	256	3 × 3/1	13 × 13 × 384	13 × 13 × 256
22	Convolutional	255	1 × 1/1	13 × 13 × 256	13 × 13 × 256
23	YOLO				

Sumber : (Redmon and Farhadi, 2018)

Gambar 2. 8 Arsitektur Algoritma YOLO

2.2.11 *Numpy*

Numpy merupakan library python yang mampu melakukan komputasi numerik maupun scientific yang dikembangkan untuk pemrograman array. Teknologi numpy mendukung pengolahan Numpy array yaitu struktur data yang memiliki efisiensi dalam menyimpan dan mengakses multidimensional array dan memungkinkan komputasu scientific yang lebih bervariasi (Harris et al., 2020). Numpy pada penelitian ini digunakan untuk melakukan manipulasi array pada gambar 2D secara spesifik dan efisien. Setiap pixel dalam gambar memiliki nilai tertentu tergantung dari warna yang umumnya RGB bernilai antara 0-255. Gambar dibawah merupakan beberapa konsep *fundamental* yang dapat dilakukan menggunakan Numpy:



Sumber : (Harris et al., 2020)

Gambar 2. 9 Beberapa konsep fundamental array pada numpy

2.2.12 OpenCV

OpenCV merupakan *library open source* untuk *computer vision* dan *machine learning*. OpenCV dibangun dengan infrastruktur yang menyediakan pengaplikasi dan akselerasi mengenai pengolahan citra digital dengan sangat mudah. Terdapat lebih dari 2500 algoritma yang dioptimasi dimana secara keseluruhan meliputi pengolahan citra dengan cara klasik maupun modern untuk *computer vision* dan *machine learning* (Brahmbhatt, 2013). Penelitian yang dilakukan berfokus pada pengolahan citra dengan berbasis model *machine learning* YOLO untuk melakukan objek deteksi dan klasifikasi pada kendaraan roda dua dan roda empat pada citra gambar bergerak atau video baik secara realtime maupun rekaman.

2.2.13 *Tuning Hyperparameter*

Machine learning yang menghasilkan model yang baik khususnya pada akurasi yang tinggi dapat dilakukan dengan cara melakukan penyesuaian pada parameter pada saata melakukan *training* model. Terdapat parameter lain yang tidak dapat dipelajari secara langsung melalui prosedur pelatihan model pada umunya namun memiliki pengaruh sangat tinggi dalam menghasilkan model terbaik yaitu *hyperparameter*. Model yang baik bukan hanya mengenai akurasi yang tinggi ataupun nilai *loss* yang sangat kecil namun model yang baik adalah dapat menyesuaikan dengan baik berbagai macam data pada saat pengujian maupun

ketika model diterapkan pada sistem. *Hyperparameter tuning* merupakan suatu kegiatan dalam memilih satu set *hyperparameter* yang optimal untuk diterapkan dalam algoritma pembelajaran atau model *machine learning* (Suyanto, 2017). *Tuning Hyperparameter* digunakan sebagai opsi tambahan yang sifatnya pilihan pada penelitian ini, jika akurasi sudah diatas 80% dan error mse kurang dari estimasi 20 maka penelitian dianggap telah memenuhi kriteria dan tidak perlu dilakukan *hyperparameter tuning*.

2.2.14 Metrik Evaluasi

Klasifikasi menggunakan machine learning menghasilkan sebuah model yang dapat digunakan pada suatu sistem dalam menentukan nilai suatu data baru. Model *machine learning* yang dibuat, baik ataupun buruknya diketahui melalui metrik evaluasi model. Evaluasi model memiliki beberapa evaluasi yaitu akurasi atau tingkat pengenalan, *loss* atau tingkat kesalahan, *recall* atau sensitifitas atau true *positive rate*, *specificity* atau *true negativity rate*, *precision* atau rasio prediksi benar dengan keseluruhan hasil yang diprediksi positif, F-*measure* atau F_1 atau *F-score* atau rata – rata harmonik dari nilai *precision* dan *recall* dan F_β (Suyanto, 2017). *Confusion matrix* digunakan untuk menentukan performa hasil yang didapatkan baik dari model *training machine learning* dan aplikasi sistem klasifikasi yang sudah diintegrasikan dengan model *machine learning*.

Tabel 2. 1 *Confusion Matrix*

Table *Confusion Matrix*		Nilai Aktual		Jumlah
		Positif	Negatif	
Nilai Prediksi	Positif	TP	FP	P
	Negatif	FN	TN	N

Tabel diatas merupakan tabel confusion matrix yang terdiri dari nilai true positive sebagai nilai label positif yang diprediksi positif, true negative sebagai nilai label negatif yang diprediksi negatif, false positive sebagai nilai label negatif yang diprediksi posistif dan false negative sebagai nilai label positif yang diprediksi negatif. Keempat nilai tersebut dapat digunakan untuk menghitung nilai accuracy, error, recall, specifity, precision dan F-score, dijelaskan melalui tabel dibawah ini:

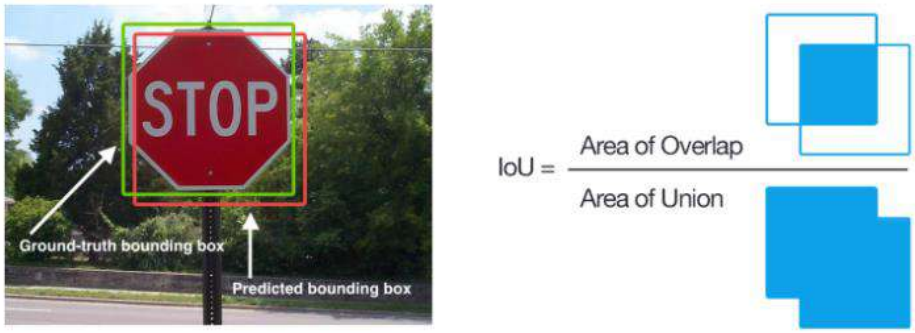
Tabel 2. 2 Metrik evaluasi model klasifikasi

No	Metrik	Rumus
1	Akurasi (tingkat prediksi)	$\frac{TP + TN}{P + N}$
2	*Error rate* (tingkat kesalahan prediksi)	$\frac{FP + FN}{P + N}$
3	*Recall* (sensitifitas)	$\frac{TP}{P}$
4	*Precision* (rasio prediksi benar)	$\frac{TP}{TP + FP}$
5	*Specificity* (rasio *true negative*)	$\frac{TN}{N}$
6	*F-score* (rata – rata harmonik metrik *precision* & *recall*)	$\frac{2 \times precision \times recall}{precision + recall}$

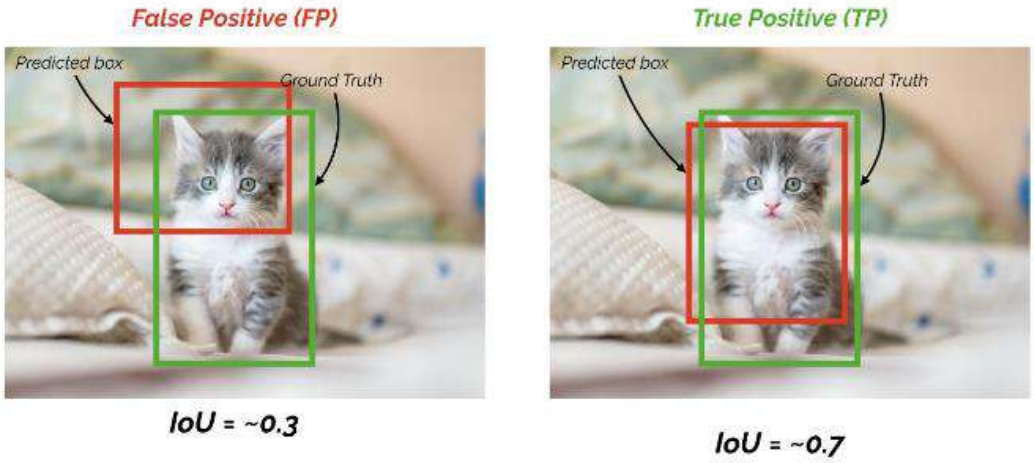
Metrik evaluasi pada penelitian ini yang digunakan ada dua yaitu metrik evaluasi untuk objek deteksi yang dilakukan sistem menggunakan mAP (*mean average precision*) dan menentukan error pada aplikasi sistem klasifikasi menggunakan mse (mean *squared error*). Penjelasan kedua metrik evaluasi tersebut sebagai berikut:

a. Metrik Evaluasi Objek Deteksi MAP (*Mean Average Precision*)

Metrik evaluasi mAP umumnya digunakan untuk mengevaluasi model *machine learning* objek deteksi seperti Fast R-CNN, YOLO, Mask R-CNN dan sejenisnya. Nilai mAP dihitung berdasarkan nilai *precision* dan *recall* dalam rentang 0 – 1. Formula dalam mencari evaluasi mAP adalah dengan menggunakan sub metrik diantaranya confusion matrix, intersection over union (IoU), recal dan precision.



Gambar 2. 10 Intersection over Union



Gambar 2. 11 Menghitung IoU dengan IoU threshold = 0.5

Beberapa tahapan dalam menghitung average precision (AP) adalah sebagai berikut:

1. Mengetahui nilai prediksi dari model.
2. Konversi nilai prediksi ke label kelas klasifikasi.
3. Menghitung nilai confusion matrix – TP, TN, FP dan FN.
4. Menghitung metrik precision dan recall.
5. Menghitung nilai bawah area pada kurva precision – recall.
6. Mengukur average precision.

Melalui tahapan tersebut dapat dihitung nilai mAP dengan mencari nilai average precision (AP) masing – masing kelas dan kemudian rata – rata dengan jumlah kelas yang ada.

$$mAP = \frac{1}{N} \sum_{1=1}^{N} AP_i \quad \text{.....(2.1)}$$

Keterangan :

mAP = *mean Average Precision*

N = jumlah objek yang diuji

AP = *Average Precision*

b. Metrik Evaluasi Loss MSE (*Mean Squared Error*)

MSE atau singkatan dari mean squared error merupakan metrik evaluasi untuk mengetahui *error* dari kumpulan data antara data aktual

dengan data prediksi. Perhitungan MSE yaitu menghitung rata – rata kuadrat kesalahan (*error*) antara nilai perkiraan dengan perkiraan sebenarnya. Persamaan matematis dapat dituliskan sebagai berikut (Harani et al., 2019):

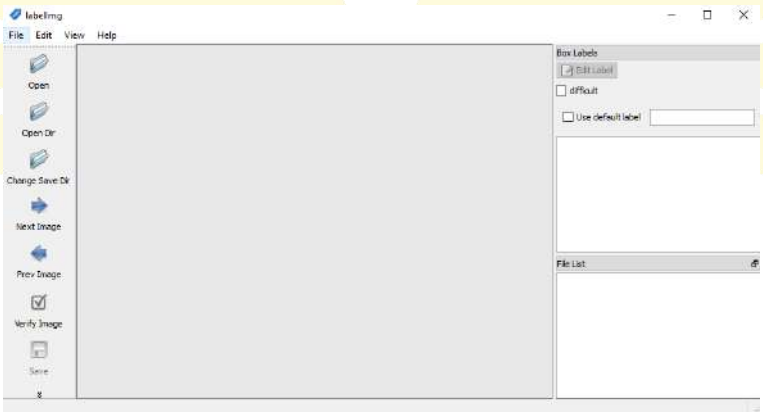
$$MSE\ (Mean\ Squared\ Error) = \sum \frac{(Nilai\ Aktual - Nilai\ Prediksi)^2}{jumlah\ data} \text{.....(2.2)}$$

$$Akurasi = \left(\frac{Prediksi}{Aktual}\right) \times 100\% \text{.....(2.3)}$$

MSE digunakan untuk menemukan *error* dari nilai yang diprediksi oleh aplikasi sistem klasifikasi kendaraan. Sehingga dapat ditarik kesimpulan melalui analisis perbandingan model *training machine learning* dengan penelitian terdahulu.

2.2.15 *LabelImg*

LabelImg merupakan software yang digunakan untuk melakukan labeling gambar atau anotasi objek pada citra atau gambar. Aplikasi LabelImg memiliki kemudahan dalam anotasi objek sebagai tanda pengenal, terdapat dua pilihan anotasi yaitu anotasi VOC yang mana outputnya file ekstensi .xml dan anotasi YOLO output file ekstensi .txt (Dewi, 2018). LabelImg digunakan untuk anotasi objek kendaraan roda dua dan roda empat baik per objek maupun beberapa objek dalam satu frame gambar. Gambar dibawah ini merupakan tampilan LabelImg:



Sumber : https://tzutalin.github.io/labelImg/

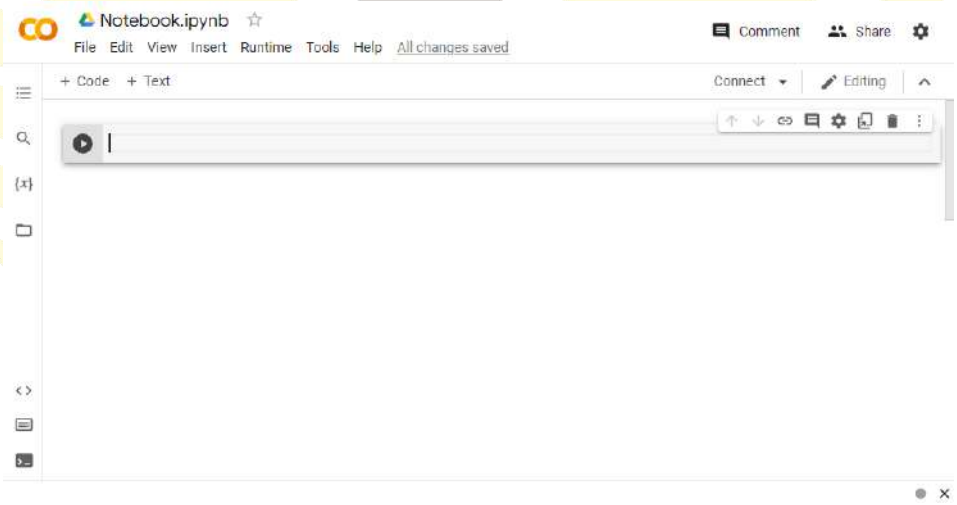
Gambar 2. 12 Antarmuka aplikasi LabelImg

2.2.16 *Tool Python Notebook*

Tool dalam melakukan *training* model *machine learning* pada penelitian ini menggunakan dua *tool* yang dinamakan *notebook*. *Notebook* dalam kaitannya dengan bahasa pemrograman python merupakan *tool code editor* yang memiliki *environment* pengembangan bahasa pemrograman python untuk melakukan pemrograman dalam bentuk *cell*. Dua *tool notebook* yang digunakan ada dua masing – masing dijelaskan sebagai berikut:

1. Google Colaboratory

   Notebook environment python yang dikembangkan oleh Google. Semua pengguna gmail dapat menggunakan nya secara gratis dijalankan melalui cloud dan memiliki fitur GPU (*Graphic Processing Unit*) sehingga sangat efektif, efisiensi serta maksimal penggunaanya dalam melakukan training model machine learning. Google Colaboratory sudah menyediakan berbagai akses library sehingga pengguna hanya melakukan import library saja (Carneiro et al., 2018). Google Colaboratory digunakan untuk melakukan training model machine learning karena mendukung komputasi menggunakan GPU. Berikut tampilan Google Colaboratory:

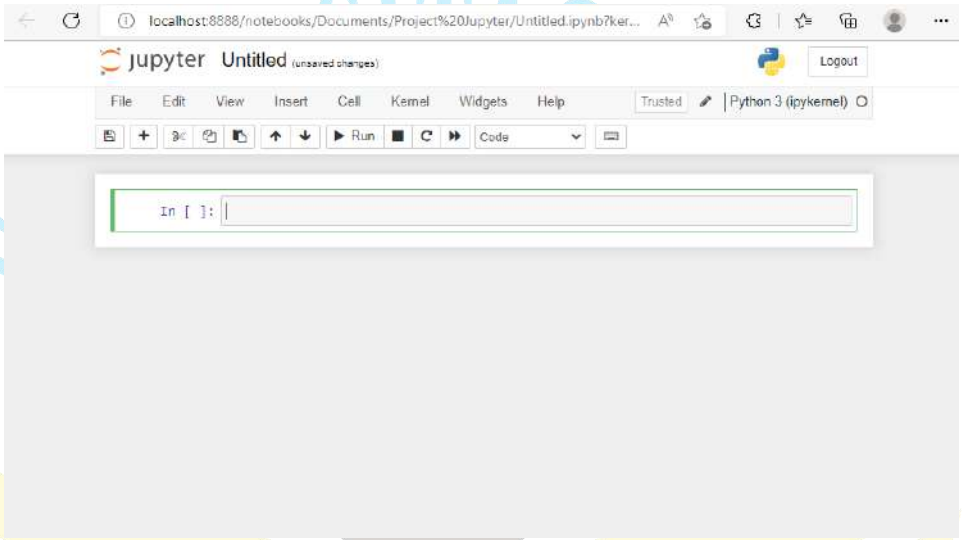


Gambar 2. 13 Antarmuka tool Google Colaboratory

2. Jupyter Notebook

   Jupyter notebook merupakan *tool notebook environment* python yang dikembangkan Anaconda Inc dimana *notebook* Jupyter dapat digunakan

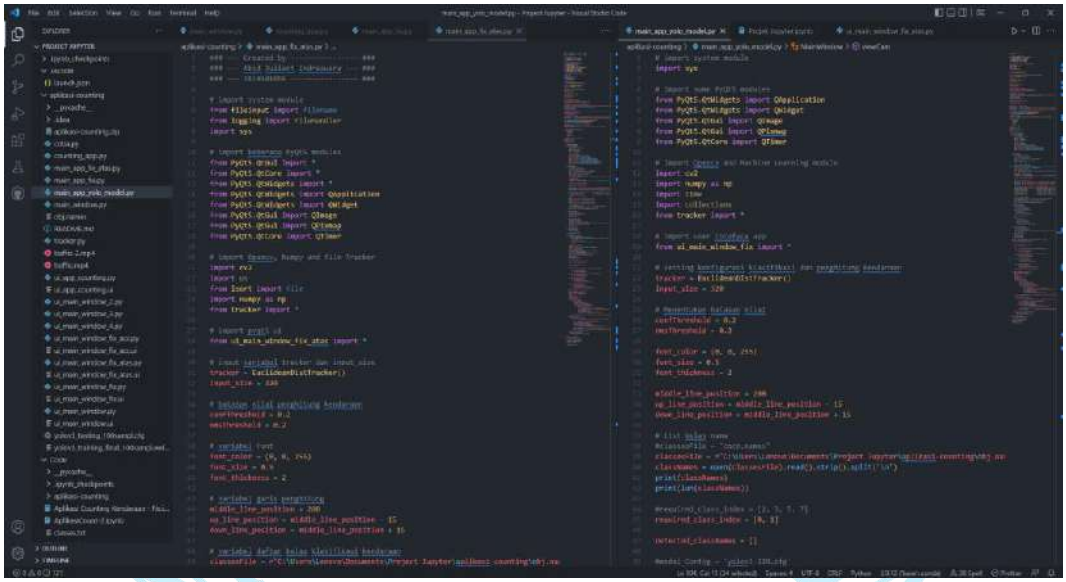
secara lokal dengan sumber daya dan kemampuan device komputer. Tidak seperti Colaboratory, perlu melakukan instalasi library yang dibutuhkan sebelum dapat menggunakan Jupyter namun kelebihannya dapat disimpan otomatis secara lokal dan dapat melakukan visualisasi tampilan gambar bergerak atau video. Jupyter notebook digunakan untuk melakukan visualisasi pengujian model *training machine learning* yang sudah dilakukan pada Google Colaboratory. Berikut tampilan Jupyter Notebook:



Gambar 2. 14 Antarmuka tool Jupyter Notebook

2.2.17 Visual Studio Code

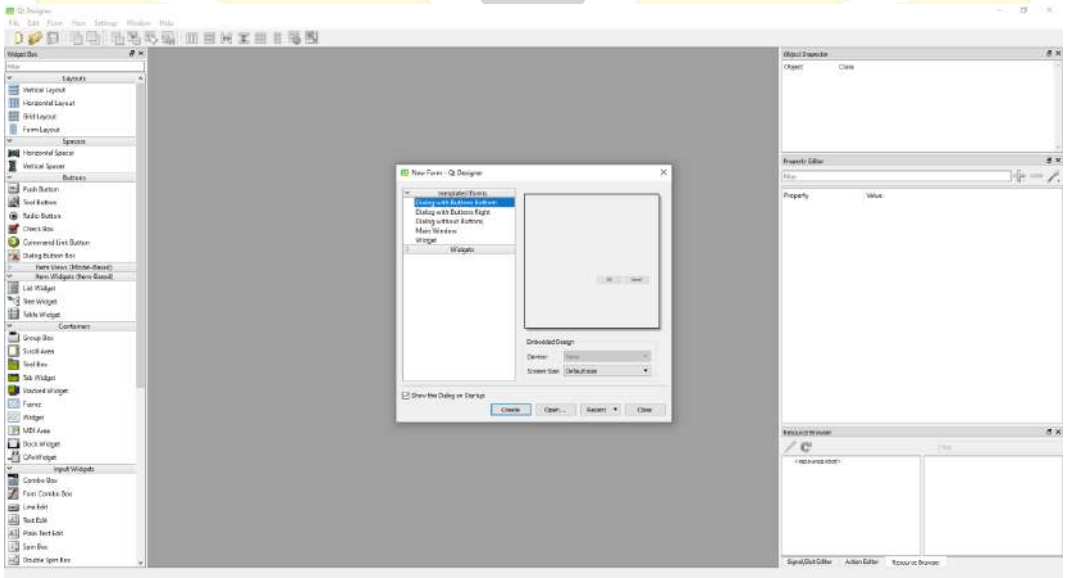
Visual Studio Code atau lebih mudah disingkat VSCode merupakan tool untuk melakukan edit kode pemrograman. VSCode adalah code editor yang mampu menerapkan berbagai environment bahasa pemrograman sehingga mudah berkomunikasi dengan menggunakan extension, dapat melakukan import library dengan mudah serta menerapkannya. Penelitian ini menerapkan GUI Aplikasi sehingga VSCode digunakan untuk melakukan implementasi kode perancangan aplikasi klasifikasi kendaraan karena memudahkan dalam integrasi model machine learning. Berikut tampilan Visual Stuido Code:



Gambar 2. 15 Antarmuka Visual Studio Code

2.2.18 *PyQt5 Designer*

PyQt5 Designer merupakan aplikasi untuk melakukan layouting aplikasi berbasis bahasa pemrograma Python. PyQt5 Designer digunakan dalam membuat antarmuka aplikasi klasifikasi kendaraan. PyQt5 adalah library Python sehingga mudah dalam mengaplikasikan model machine learning yang diintegrasikan ke aplikasi karena model machine learning di training dengan environment Python. Berikut tampilan PyQt5 Designer:



Gambar 2. 16 Antarmukas PyQt5 Designer

BAB III
METODE PENELITIAN

Metode penelitian ini menjelaskan cara penyelesaian yang akan dilakukan oleh penulis dalam penyusunan laporan penelitian. Metode penelitian yang digunakan dalam penyusunan diantaranya lokasi penelitian, alat dan bahan penelitian, waktu penelitian dan tahap penyelesaian penelitian.

3.1 Lokasi Penelitian

Pengerjaan penelitian ini memerlukan tempat untuk proses perancangan dan pembuatan aplikasi sistem *machine learning* untuk klasifikasi kendaraan pada persimpangan *traffic light*. Lokasi penelitian untuk menyelesaikan tugas akhir dengan judul Perancangan Sistem *Machine Learning* untuk Klasifikasi Kendaraan pada Persimpangan *Traffic Light* berada di Laboratorium Teknik Elektro Universitas Tidar dan di Dinas Perhubungan Kota Magelang untuk survei pendahuluan yang digunakan sebagai tambahan informasi serta referensi kebutuhan aplikasi sistem yang dibuat.

3.2 Waktu Penelitian

Waktu penelitian yang dilaksanakan untuk mengerjakan tugas akhir ini selama 4 bulan. Waktu penelitian 4 bulan yaitu dari bulan Maret 2022 sampai dengan bulan Juni 2022, Tabel 3.1 berikut merupakan rencana kegiatan yang dilakukan selama penelitian.

Tabel 3. 1Rencana Kegiatan Pelaksanaan Tugas Akhir

No	Kegiatan	Maret				April				Mei				Juni			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Survei Pendahuluan		■	■	■												
2	Studi Pustaka				■												
3	Penyusunan dan pengumpulan proposal					■											
4	Tahapan Pengambilan dan						■	■	■								

	Pengumpulan Data															
5	Perancangan Sistem															
6	Pembuatan Sistem															
7	Uji Coba Sistem															
8	Analisis hasil Pengujian															
9	Penyusunan Laporan Akhir															

3.3 Alat dan Bahan

Penelitian ini membutuhkan beberapa komponen pendukung dalam proses pembuatan yaitu alat dan bahan. Tabel 3.2 menunjukkan daftar alat yang dibutuhkan dalam proses penelitian, sebagai berikut:

Tabel 3. 2 Daftar Alat Penelitian

No	Nama Barang	Kuantitas	Satuan
1	Kamera	1	Unit
2	Laptop atau Komputer	1	Unit
3	Visual Studio Code	1	Unit
4	Tool Google Colaboratory	1	Unit
5	Software LabelImg	1	Unit
6	Software PyQt5 Designer	1	Unit
7	Library Tensorflow	1	Unit
8	Library Numpy	1	Unit
10	Library OpenCV	1	Unit
11	Stopwatch	1	Unit

Tabel 3.3 menunjukkan daftar bahan yang dibutuhkan dalam proses penelitian, sebagai berikut:

Tabel 3. 3 Daftar Bahan Penelitian

No	Nama Barang	Kuantitas	Satuan
1	Data Rekaman CCTV Traffic Light Dinas Perhubungan Kota Magelang lokasi persimpangan Gotong Royong	8	Jam
2	Dataset Gambar Kendaraan Roda Dua	1	Unit
3	Dataset Gambar Kendaraan Roda Empat	1	Unit

3.4 Variabel Penelitian

Penelitian yang dilakukan menggunakan beberapa variabel yang berpengaruh pada hasil penelitian ini yaitu variabel terikat dan variabel bebas.

3.4.1 Variabel Terikat

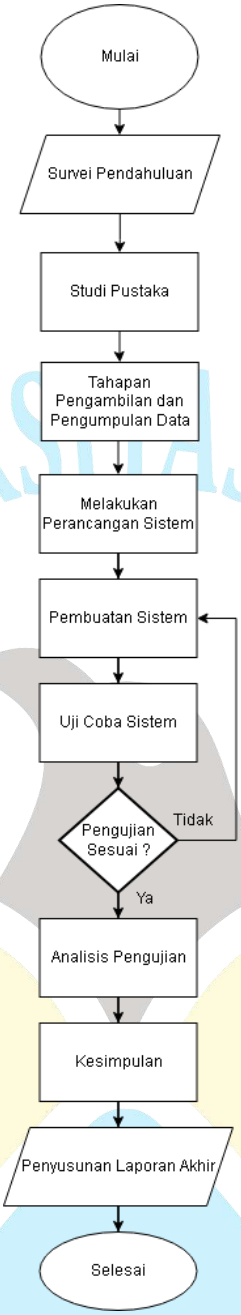
Variabel terikat atau variabel yang dipengaruhi oleh variabel lain pada penelitian ini yaitu kelas atau golongan kendaraan. Data kendaraan diambil melalui public repository. Terdapat setidaknya 2 kelas kendaraan yang digunakan yaitu roda dua dan roda empat.

3.4.2 Variabel Bebas

Variabel bebas yang merupakan variabel yang mempunyai pengaruh terhadap perubahan yang terjadi pada variabel lain. Variabel bebas pada penelitian ini adalah jenis layer, algoritma optimasi, fungsi aktivasi dan metrik evaluasi terdiri dari error serta akurasi.

3.5 Diagram Alir Penelitian

Penelitian ini menggunakan beberapa tahapan meliputi studi pustaka, melakukan perancangan sistem meliputi perancangan alat pengukuran dan perancangan sistem *machine learning*, pembuatan alat, uji coba sistem, analisis pengujian, kesimpulan, dan penyusunan laporan akhir. Diagram alir penelitian ditunjukkan pada Gambar 3.1,



Gambar 3. 1 Diagram Alir Penelitian

3.5.1 Survei Pendahuluan

Langkah ini dilakukan untuk mengetahui kebutuhan dan kondisi Dinas Perhubungan terhadap data yang akan dihasilkan dari penelitian dan melakukan pengambilan data rekaman CCTV lampu lalu lintas yang akan dijadikan objek penelitian untuk melakukan perancangan sistem klasifikasi kendaraan. Diskusi

dengan pihak yang memahami diperlukan untuk mengetahui lebih detail permasalahan sebenarnya kondisi lampu lalu lintas.

3.5.2 Studi Pustaka

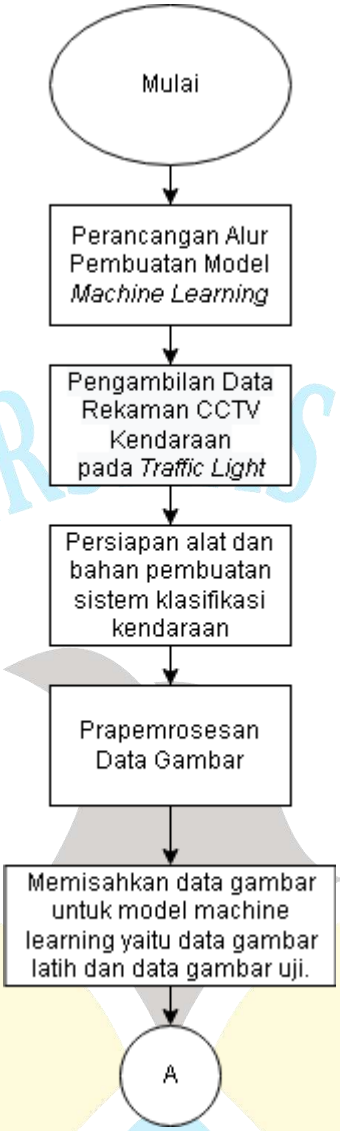
Hasil penelitian-penelitian terdahulu digunakan sebagai referensi dalam penelitian ini. Penulis mengumpulkan studi literatur untuk menunjang penelitian yang dilakukan. Studi pustaka sangatlah diperlukan dalam langkah awal dalam melakukan penelitian karena akan memberikan suatau gambaran awal tentang langkah-langkah yang akan dilakukan, serta mendapatkan pemahaman dalam proses penelitian, penelitian yang pernah dilakukan pendahulunya, dan dapat mengetahui kelebihan dan kekurangannya. Referensi dapat berupa jurnal penelitian sebelumnya, internet, buku dan yang lainnya. Beberapa referensi yang diperlukan dalam penelitian ini yaitu cara kerja sistem algoritma pengolahan citra digital, perancangan sistem klasifikasi objek kendaraan pada *traffic light*, penyusunan dan pembuatan model kecerdasan buatan untuk diintegrasikan ke dalam sistem klasifikasi objek kendaraan.

3.5.3 Parameter yang Digunakan

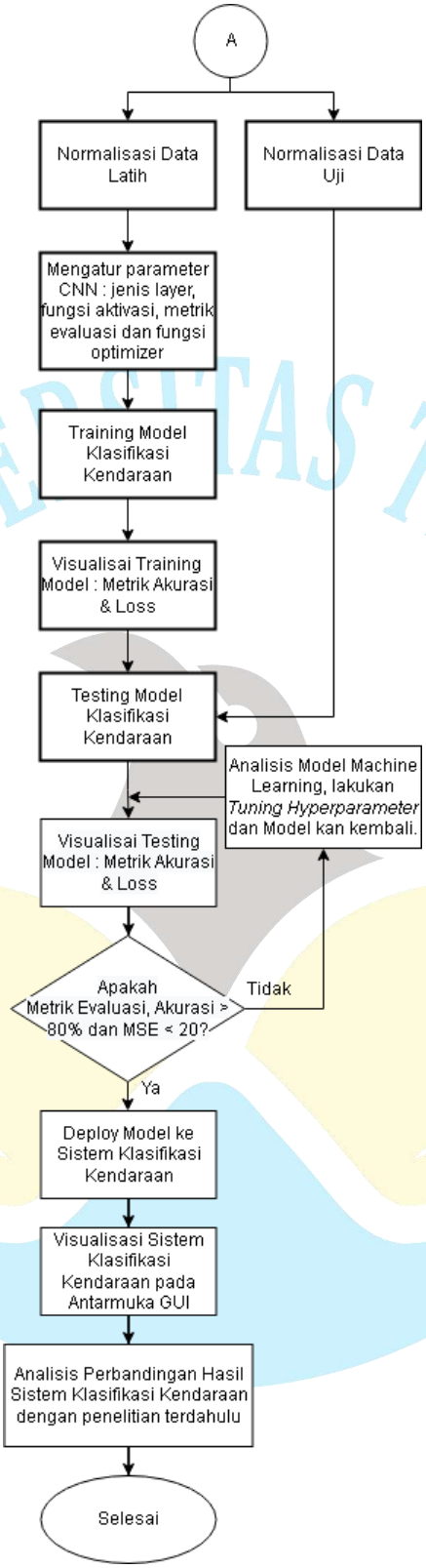
Parameter yang digunakan pada analisis sistem klasifikasi kendaraan ini diantaranya adalah objek kendaraan roda dua dan roda empat pada persimpangan *traffic light*. Data diambil dari hasil rekaman CCTV lampu lalu lintas yang didapatkan melalui Dinas Perhubungan Kota Magelang dan pengambilan video secara manual pada persimpangan traffic light. Klasifikasi yang digunakan adalah berdasarkan dimensi ukuran kendaraan dan pola bentuk gambar kendaraan yang dibedakan menjadi dua golongan yaitu kendaraan roda dua dan kendaraan roda empat.

3.5.4 Perancangan Sistem *Machine Learning*

Perancangan alat berdasarkan berbagai studi literatur sehingga perancangan ini dapat mengurangi kekurangan pada penelitian yang yang pernah dilakukan sebelumnya. Diagram alir cara kerja sistem secara keseluruhan ditunjukkan pada Gambar 3.2, sebagai berikut:



Gambar 3. 2 Diagram Alir Cara Kerja Sistem Bagian Start



Gambar 3. 3 Diagram Alir Cara Kerja Sistem Bagian A

Berdasarkan gambar diagram alir cara kerja sistem pada Gambar 3.2 dapat diuraikan tahap perancangan sistem sebagai berikut:

1. Perancangan Sistem Model *Machine Learning*

Model *machine learning* memiliki karakteristik yang berbeda beda tergantung dengan algoritma yang digunakan. Pengolahan citra menggunakan algoritma *Convolutional Neural Network*. Sebelum membuat model yang dibutuhkan penelitian yaitu melakukan *preprocessing* data gambar yang dibutuhkan. Model rancangan dibuat dalam bentuk *pipeline* menyusun *layer* algoritma *neural network* yang dibutuhkan dalam melakukan penelitian.

2. Perancangan GUI (*Graphical User Interface*) Aplikasi

Perancangan GUI Aplikasi disusun sesuai kebutuhan yaitu fitur yang ada sebatas aplikasi klasifikasi kendaraan yang menjadi fokus pada penelitian. Desain layout menggunakan PyQt5 Designer dan fungsi fitur dibuat dengan kode python. Aplikasi yang sudah memiliki fungsi kemudian diintegrasikan dengan model *machine learning* yang di training.

3.5.5 Pembuatan Sistem *Machine Learning*

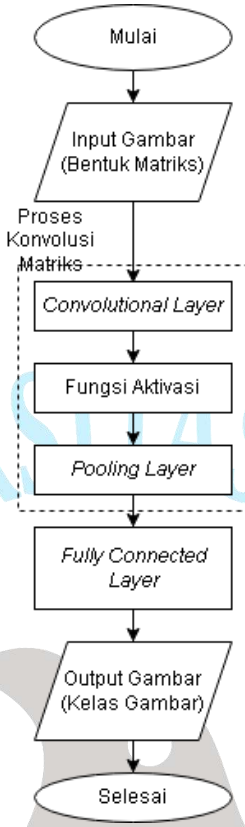
Setelah selesai melakukan tahap perancangan, proses selanjutnya adalah pembuatan alat dilakukan setelah tahap perancangan. Langkah-langkah yang dilakukan pada tahap pembuatan yaitu:

1. Menyiapkan semua alat dan bahan yang dibutuhkan.
2. Membuat diagram sistem yang akan digunakan dalam menentukan perancangan alur pengerjaan penelitian.
3. Melakukan pemrosesan data gambar kendaraan roda dua dan kendaraan roda empat. Pemrosesan data gambar dapat berupa labelling data gambar, penyesuaian ukuran gambar, penyesuaian jenis citra warna yang digunakan untuk training model dan melakukan pembagian data latih serta data uji.
4. Membuat model *machine learning* dengan melakukan pengaturan pada parameter *Convolutional Neural Network* (CNN) diantaranya fungsi aktivasi, metode metrik evaluasi, jenis layer dan fungsi *optimizer*.
5. Melakukan *training model machine* learning menggunakan parameter yang sudah disetel dan data gambar latih yang sudah ditentukan.

6. Melakukan pengujian pada training model dengan mengujinya menggunakan data gambar uji yang telah ditentukan.
7. Menampikan metrik evaluasi dan melakukan analisis pada akurasi serta error model *machine learning* yang sudah dibuat sebelum dilakukan *deploy* atau integrasi model ke sistem aplikasi klasifikasi dan counting untuk mendapatkan akurasi dan *loss* yang maksimal sesuai batas yang ditentukan yaitu akurasi > 80% dan estimasi MSE < 20. Jika model *machine learning* masih belum memenuhi maka akan dilakukan *tuning hyperparameter* untuk memaksimalkan hasil dari akurasi dan *loss*.
8. Membuat sistem klasifikasi dan *counting* kendaraan berdasarkan golongan kendaraan motor, kendaraan mobil, kendaraan truk dan kendaraan bus dengan menggunakan bahasa pemrograman Python library PyQt5 dalam bentuk GUI dekstop.
9. Melakukan *deploy* atau integrasi model *machine learning* ke dalam aplikasi GUI dekstop untuk visualisasi dan evaluasi sistem yang telah dibuat.
10. Sistem telah selesai dan siap untuk dilakukan pengujian aplikasi sistem *counting* yang dibandingkan dengan data real-time kendaraan yang lewat pada lampu lalu lintas.
11. Melakukan analisis dan menarik kesimpulan dari hasil pengujian aplikasi klasifikasi dan *counting* kendaraan yang sudah dibuat.

3.5.6 Training Model Algoritma *Convolutional Neural Network*

Training model machine learning menggunakan algoritma convolutional neural network memiliki beberapa tahapan untuk mendapatkan keluaran gambar yang dapat mengenali kelas kendaraan roda empat dan roda dua dari *input* data gambar *training*. Tahapan training model algoritma convolutional neural network terdapat pada Gambar 3.4



Gambar 3. 4 Diagram Alir Training Model menggunakan Algoritma Convolutional Neural Network

Tahapan utama yang umum pada lapisan CNN terdiri dari *input layer*, beberapa *hidden layer* dan *output layer*. Training model menggunakan algoritma CNN terdapat beberapa tahap pada proses *layer* yaitu *convolutional layer*, fungsi aktivasi, *pooling layer* dan *fully connected layer*. Melalui diagram alir pada Gambar 3.4 dapat dijelaskan sebagai berikut

a. *Input Layer*

   Input layer merupakan proses dimana gambar diubah ke dalam bentuk matematis matriks. Tiap *pixel* pada gambar memiliki nilai tersendiri dalam matriks yang akan diproses untuk dilakukan ekstraksi fitur. Sehingga melalui ekstraksi fitur akan mendapatkan fitur objek yang bisa diklasifikasikan ke dalam kelas objek tertentu.

b. *Convolutional Layer*

Convolutional layer bertujuan untuk menentukan jumlah *hidden layer*. Secara umum mengubah nilai piksel input yang diekstrak sehingga mendapatkan nilai baru dengan menggunakan kernel / *filter*. Nilai matriks input akan berubah nilai nya dengan menggunakan perhitungan dot products kernel / *filter*.

c. Fungsi Aktivasi

Fungsi aktivasi digunakan untuk melakukan normalisasi nilai matriks *output* dari *convolutional layer* sehingga nilai matriks berkurang pada *background* dan nilai matriks bertambah pada fitur objek. Fungsi aktivasi yang digunakan pada penelitian ini adalah ReLu. ReLu atau singkatan dari rectified linear unit merupakan fungsi aktivasi yang membuat pembatas nilai nol, yang artinya jika $x \leq 0$ maka $x = 0$ dan jika $x \geq 0$ maka $x = x$. Fungsi aktivasi diterapkan pada tiap piksel sehingga dapat menyeleksi nilai yang merupaka fitur objek dari gambar.

d. *Pooling Layer*

Pooling layer memiliki tujuan untuk mengurangi nilai pada dimensi matriks. Penelitian ini menggunakan *maxPooling layer* sehingga dapat mengurangi dimensi nilai matriks dari *output* fungsi aktivasi.

e. *Fully Connected Layer*

Ketiga proses yang sudah disebutkan diatas merupakan proses konvolusi sehingga mendapatkan matriks dengan fitur ekstraksi. Selanjutnya matriks dengan fitur ektraksi tersebut diubah bentukanya ke dalam bentuk vektor sehingga dapat diklasifikasikan dalam kelas tertentu.

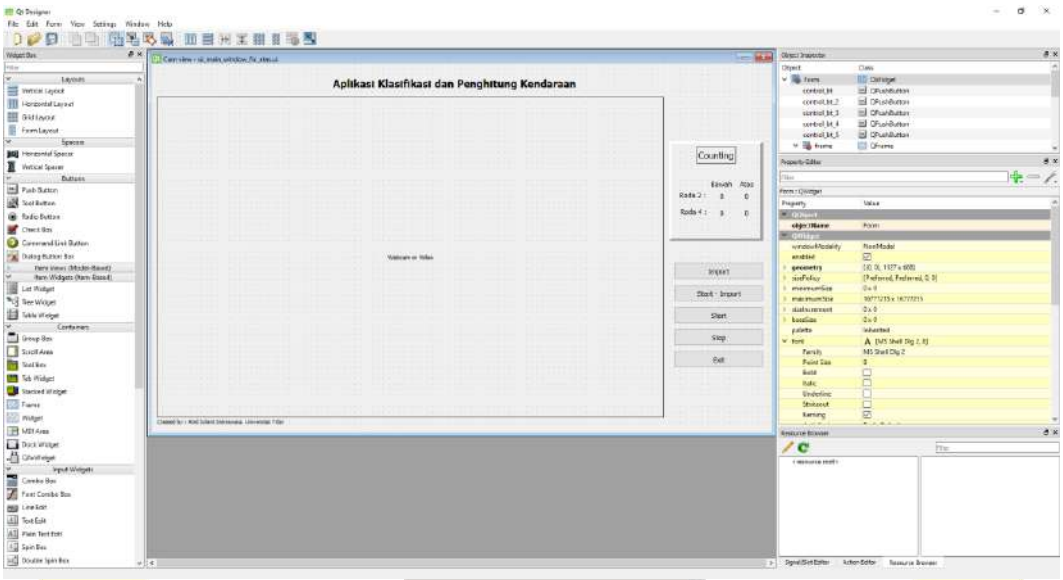
f. *Output Layer*

Ouput layer berupa vektor dengan kelas tertentu yang dijadikan sebagai sebuah bentuk format model *machine learning* tertentu tergantung *framework* yang digunakan untuk *training* model *machine learning*.

3.5.7 Pembuatan Program GUI Aplikasi dan Deploy Model *Machine Learning*

Pembuatan program GUI Aplikasi menggunakan PyQt5 Designer untuk memudahkan dalam pembuatan layout aplikasi Dekstop lebih mudah. Kemudian menambahkan fitur tampilan gambar, start, stop, import video, start cam untuk

video realtime, exit dan fitur penghitung otomatis. Setiap fitur ditambahkan fungsi agar dapat berjalan sesuai dengan perintah yang diberikan dengan membuat kode fungsi pada code editor. Deploy model machine learning dilakukan dengan cara menambahkan file weight dan cfg ke dalam baris kode fungsi agar membaca objek kendaraan yang terbaca.



Gambar 3. 5 Tahap Pembuatan GUI Aplikasi

3.5.8 Pengujian

Penulis melakukan uji coba pada aplikasi yang sudah dibuat untuk memastikan sistem sudah berfungsi dengan baik atau tidak. Alat dapat dikatakan berfungsi dengan baik jika alat mampu menampilkan klasifikasi jenis kendaraan pada golongan roda dua dan roda empat, tingkat akurasi pengenalan jenis kendaraan, penghitungan jumlah jenis kendaraan secara real time serta dengan nilai *error* yang sedikit pada hasil uji coba. Apabila ditemukan aplikasi tidak berfungsi sebagaimana mestinya maka perlu dilakukan pemeriksaan terhadap aplikasi yang sudah dibuat. Proses pengujian dilakukan secara peripheral atau terpisah sesuai bagian masing- masing, proses pengujian terbagi menjadi 2 tahap, diantaranya:

a. Pengujian Sistem *Machine Learning*

Pengujian sistem machine learning yaitu pada model sistem hasil training model data gambar latih menggunakan *Convolutional Neural Network* dibandingkan

dengan data gambar uji yang sudah di split pada preprocessing data. Pengujian model training dengan data gambar uji yang menggunakan metode penghitungan akurasi salah satunya menggunakan MAP (*Mean Average Precision*) dan *Confusion Matrix*.

b. Pengujian Sistem Klasifikasi Kendaraan

Pengujian sisten klasifikasi kendaraan dilakukan setelah model *machine learning* memenuhi akurasi dan *error* yang sesuai serta model berhasil di *deploy* pada aplikasi. Pengujian pada sistem dilakukan dengan membandingkan antara jumlah kendaraan yang berhasil diklasifikasi dan dihitung dibandingkan dengan jumlah kendaraan aslinya dihitung secara manual. Kemudian ditentukan nilai *error* dapat menggunakan MSE (*Mean Average Precision*).

3.5.9 Analisis Data Hasil Pengujian

Penulis akan menganalisis hasil pengujian antara pengujian sistem *machine learning* yang dilakukan dengan data gambar uji dan pengujian sistem klasifikasi dimana model sistem *machine learning* sudah di *deploy* yang diuji menggunakan video rekaman CCTV lampu lalu lintas. Analisis dari dua metode pengujian untuk mengetahui seberapa baik model yang sudah dibuat mampu melakukan klasifikasi dengan baik pada kasus nyata. Kemudian akan dilakukan analisis terhadap model *machine learning* yang sudah dibuat dibandingan dengan model *machine learning* peneliti pendahulu guna mendapatkan sistem dan kesimpulan yang baik..

3.5.10 Kesimpulan

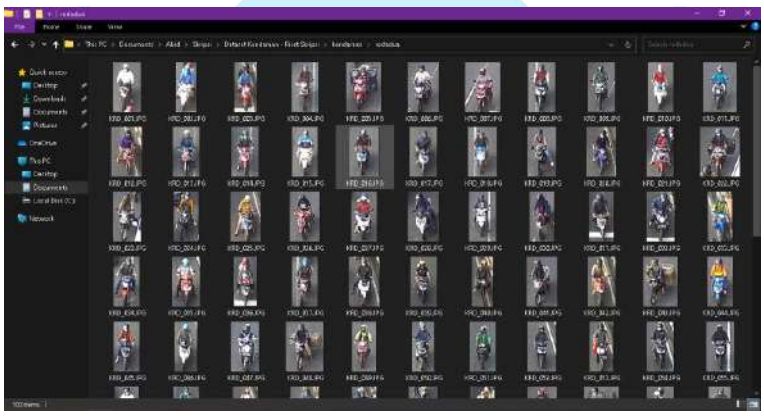
Penulis menarik kesimpulan berdasarkan hasil penelitian dan analisis yang sudah dilakukan. Pada kesimpulan penulis juga menuliskan kekurangan dari sistem yang dibuat.

BAB IV
HASIL DAN PEMBAHASAN

Penelitian tugas akhir dalam pembuatan sistem *machine learning* untuk klasifikasi kendaraan pada persimpangan *traffic light* diperlukan data dalam proses penyusunan laporan penelitian. Sistem klasifikasi kendaraan disusun sesuai dengan tahapan pada metode penelitian, berikut hasil dan analisis penelitian yang telah dilakukan:

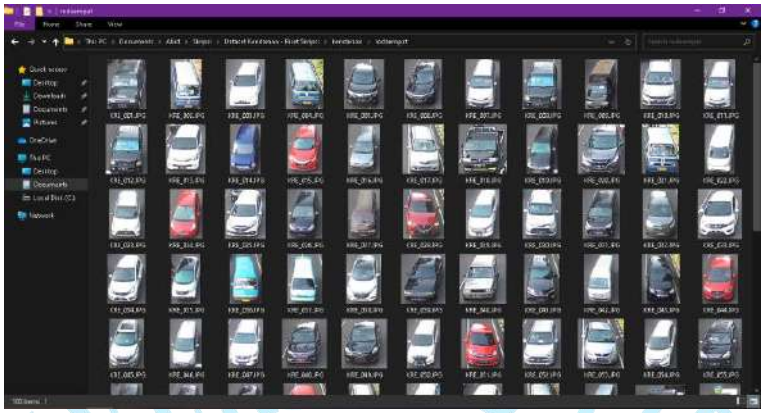
4.1 Pengumpulan *Dataset*

Data rekaman CCTV lampu lalu lintas diambil melalui survei pendahuluan dengan Dinas Perhubungan Kota Magelang. Rekaman dilakukan pada tanggal 22 April 2022 pukul 06.12 WIB dengan durasi 40 menit. Data gambar yang digunakan adalah gambar kendaraan roda dua dan roda empat yang di ambil secara manual *cropping* objek kendaraan dari rekaman CCTV lampu lalu lintas. Data gambar yang diambil disesuaikan environment nyata agar mudah diterapkan dan tepat dalam melakukan klasifikasi. Data gambar yang digunakan terdiri dari 3 jenis yaitu kendaraan roda dua berjumlah 100 data gambar, kendaraan roda empat berjumlah 100 data gambar dan gabungan kedua jenis kendaraan roda dua empat dalam satu frame gambar berjumlah 40 data gambar. Pengambilan gambar objek kendaraan diambil secara acak dari berbagai sudut pandang. Kendaraan roda dua berjumlah 100 buah data gambar, untuk objek gambar yang dikumpulkan terlihat pada Gambar 4.1.



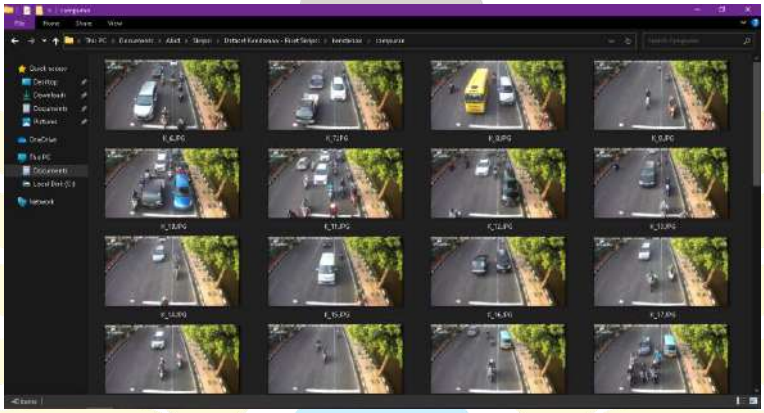
Gambar 4. 1 Data Gambar Kendaraan Roda Dua

Data gambar kendaraan roda berjumlah 100 buah gambar dengan objek gambar diambil sesuai seperti Gambar 4.2.



Gambar 4. 2 Data Gambar Kendaraan Roda Empat

Data gambar kendaraan roda dua dan roda empat dalam satu *frame* digunakan untuk melatih model machine learning yang mampu beradaptasi sesuai *environment* yang terlihat pada kamera CCTV lampu lalu lintas. Setiap objek kendaraan pada *frame* tersebut akan dilakukan anotasi sehingga dapat dikenali oleh sistem, yang terlihat pada Gambar 4.3.



Gambar 4. 3 Data Gambar Kendaraan Roda Dua Empat dalam Satu *Frame*

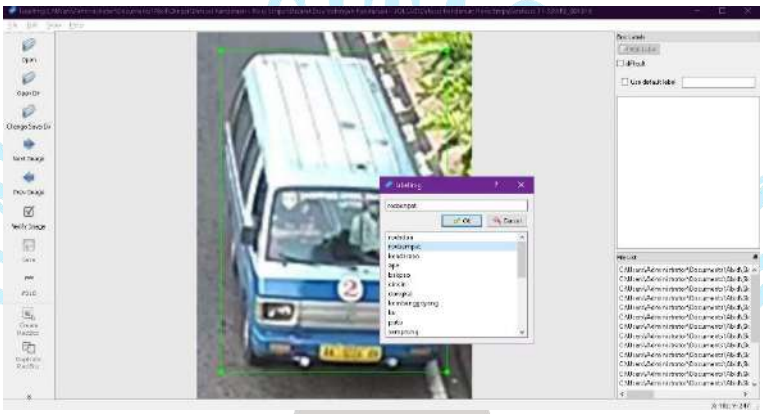
4.2 *Preprocessing* Data Gambar

Data yang diperoleh kemudian dilakukan beberapa pengolahan untuk mempersiapkan data agar siap digunakan dalam melakukan *training* model *machine learning*. Metode yang digunakan yaitu *Convolutional Neural Network* dengan pengembangan yang dinamakan Yolo (*You Only Look Once*).

4.2.1 Pelabelan Data Gambar

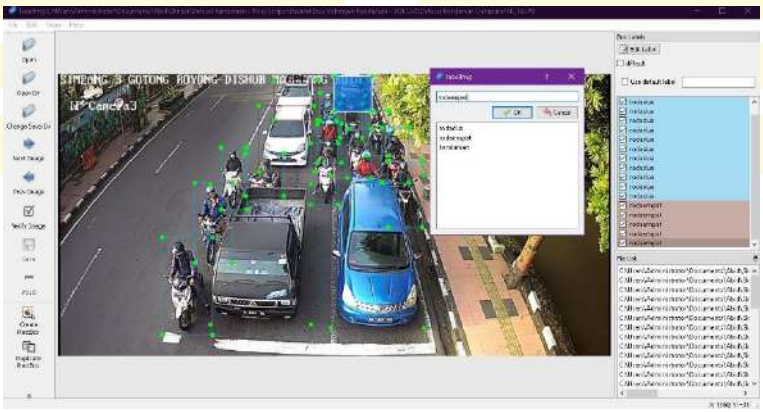
Tahap awal yang dilakukan yaitu pelabelan data gambar masing-masing kelas dan data gambar gabungan dalam satu *frame*. Pelabelan dilakukan sebagai

tanda pengenal gambar yang menyimpan informasi dari gambar yang dilabeli. Label gambar disimpan dalam berkas berekstensi TXT dengan format YOLO. Pelabelan terhadap seluruh data gambar berjumlah 240 dilakukan secara manual dengan aplikasi LabelImg. Label TXT terdiri dari kelas objek, posisi objek koordinat x, posisi objek koordinat y, ukuran panjang objek dan ukuran lebar objek. Anotasi atau pelabelan pengenalan objek kendaraan dilakukan satu per satu menggunakan LabelImg seperti terlihat pada objek mobil Gambar 4.4.



Gambar 4. 4 Pelabelan Data Gambar Objek Kendaraan

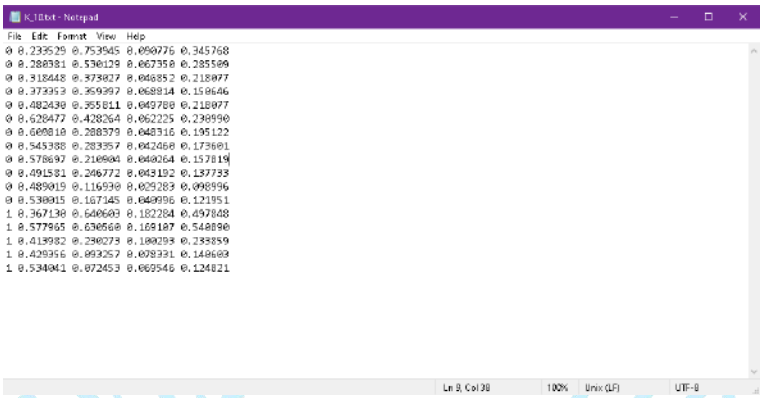
Pelabelan juga dilakukan pada gambar kendaraan roda dua dan roda empat dalam satu *frame*, yang dianotasi setiap objek kendaraan yang ada dalam *frame* tersebut. Proses pelabelan objek kendaraan roda dua dan roda empat dalam satu *frame* terlihat pada Gambar 4.5.



Gambar 4. 5 Pelabelan Data Gambar Gabungan Objek Kendaraan dalam Satu *Frame*

Pelabelan gambar yang sudah dilakukan akan menghasilkan file berekstensi .TXT yang dibaca sebagai anotasi atau label YOLO sehingga akan mudah dibaca

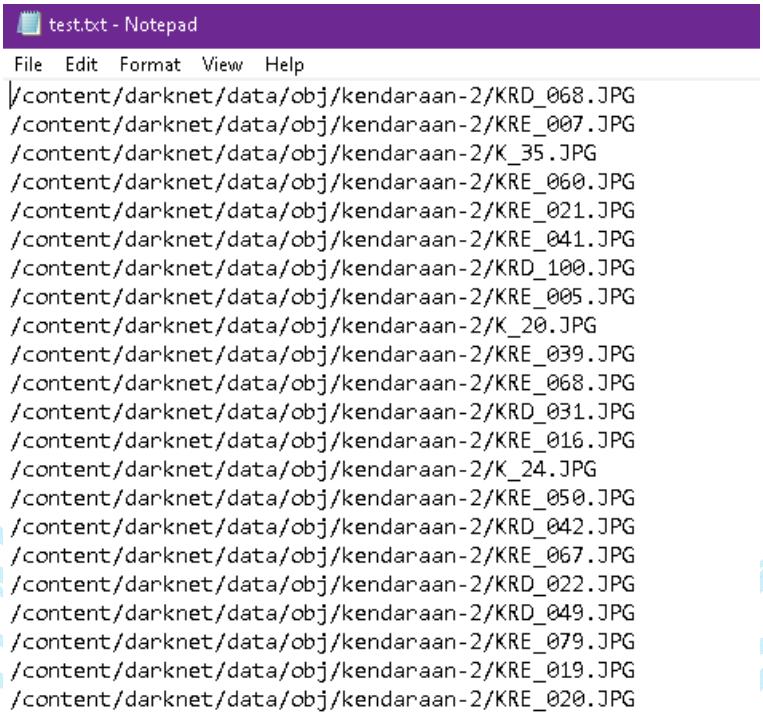
oleh sistem pada *training* model *machine learning*. Hasil pelabelan objek kendaraan pada gambar terlihat pada Gambar 4.6.



Gambar 4. 6 File Label dengan Format YOLO TXT

4.2.2 *Train Test Split* Data Gambar

Sebelum melakukan *training*, data gambar yang sudah dilabeli menghasilkan file label dengan format YOLO TXT sehingga masing-masing gambar memiliki label tersebut, data gambar dan file label tersebut perlu dibagi dalam beberapa bagian yaitu data gambar latih dan data gambar uji. Data gambar latih digunakan untuk melatih model *machine learning* sedangkan data gambar uji dilakukan untuk pengujian model *training*. Data gambar yang dipisahkan tidak perlu spesifik dibagi dalam folder namun membuat dua buah file yang terdiri dari train.txt dan test.txt yang berisi kumpulan *path* file label gambar yang dipisahkan. Total terdapat 240 gambar kemudian penulis membaginya untuk data *testing* sebesar 10% untuk masing-masing kelas gambar atau 22 gambar terbagi dalam gambar K sebanyak 3 buah, kelas roda dua sebanyak 6 buah dan kelas roda empat sebanyak 13 buah.



Gambar 4. 7 File test.txt yang berisi path Data Gambar Uji

4.3 Sistem Machine Learning

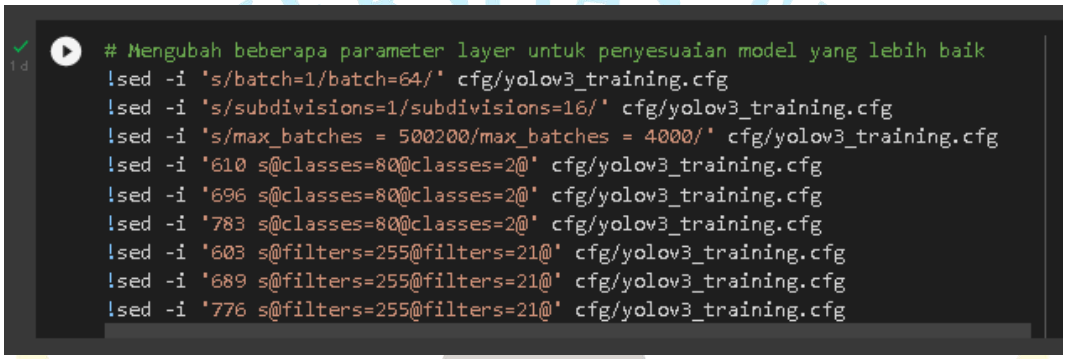
Proses pengolahan citra menggunakan pengembangan algoritma *Convolutional Neural Network* yaitu YOLO (*You Only Look Once*) yang memiliki arsitektur layer lebih kompleks namun menghasilkan akurasi yang baik ketika diterapkan dalam sistem nyata. Setidaknya terdapat beberapa tahapan yang dilakukan dalam pengolahan citra menggunakan YOLO setelah mempersiapkan dataset yang diperlukan.

4.3.1 Konfigurasi Parameter *Training Pipeline*

Parameter yang digunakan dalam Yolov3 disimpan dalam file berekstensi .cfg. Beberapa parameter yang disesuaikan dengan sistem yang akan dibangun diantaranya:

1. *batch* : jumlah sampel yang diproses dalam satu kelompok, pada penelitian ini menggunakan nilai maksimal yaitu 64 sampel yang akan diproses dalam satu kelompok.
2. *subdivision* : membagi batch dalam mini_batch sehingga GPU memproses sampel mini_batch dalam sekali proses, dalam penelitian ini menggunakan 16 subdivision yang artinya terdapat 4 kali proses dalam 64 batch.

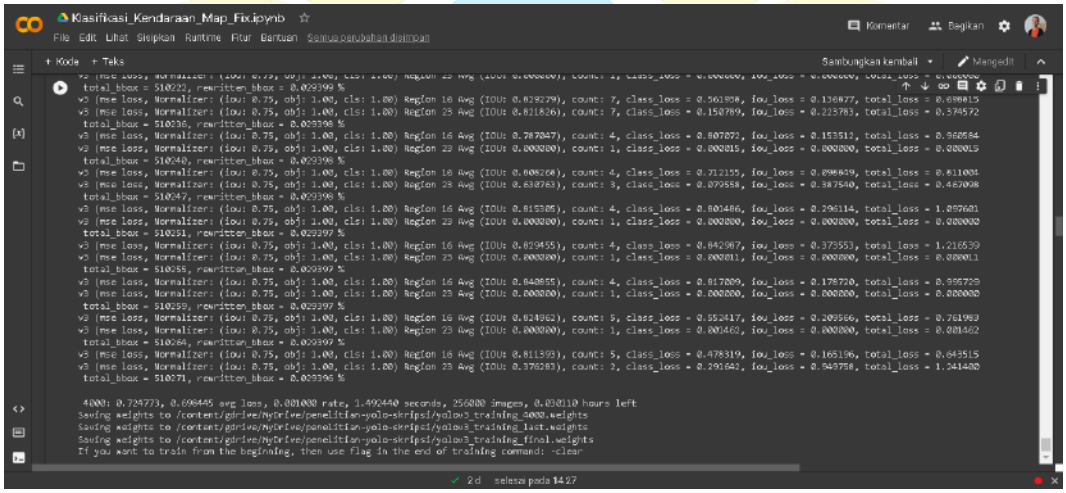
3. *max*_batches : proses training model dilakukan dalam nilai maksimal batch iterasi, pada penelitian menggunakan 4000 untuk max_batch melalui penghitungan jumlah kelas * 2000. Dimana nilai 2000 merupakan nilai minimal max_batch.
4. *classes* : jumlah kelas yang digunakan dalam training model untuk klasifikasi kendaraan roda dua dan roda empa.
5. *filter* : Parameter pada layer konvolusi sebagai fitur penyaring nilai pixel, sesuai dengan paper Yolov3 jumlah filter yang digunakan dalam Yolov3 yaitu (jumlah kelas + 5)*3 sehingga filter yang digunakan dalam penelitian ini adalah 21.



Gambar 4. 8 Parameter yang disesuaikan dengan kebutuhan sistem

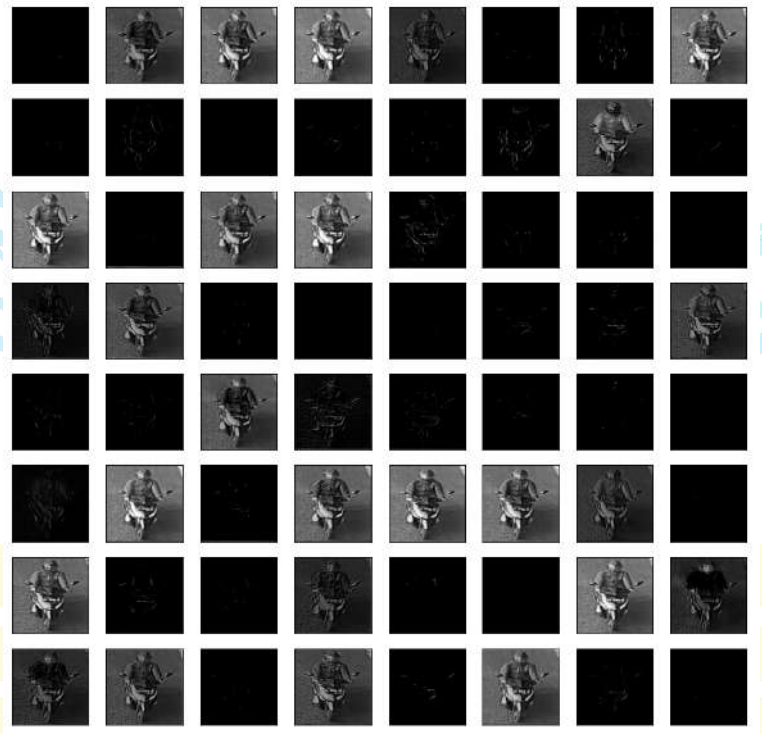
4.3.2 *Training Convolutional Neural Network*

Training model berjalan berdasarkan konfigurasi parameter menghasilkan model *machine learning* dalam ekstensi weight. Model weight akan digunakan untuk diintegrasikan dengan sistem aplikasi dalam mengklasifikasikan serta *counting* kendaraan. *Training* model menghabiskan waktu sebanyak kurang lebih satu setengah jam.



Gambar 4. 9 Proses *Training* Model

Pengolahan citra dalam algoritma *convolutional neural network* dilakukan oleh sistem dengan mengurangi nilai *pixel* pada gambar untuk dapat mengektraksi fitur dari objek dan memisahkannya dengan latar belakang gambar. Proses *filter* menggunakan konvolusi ditunjukkan Gambar 4.10 dengan menggunakan konvolusi filter conv2d (3, 3, 3, 64) dalam dimensi 3 x 3 x 3 x 64 dengan layer maxpooling2d dengan nilai 2.



Gambar 4. 10 Proses *filter* nilai *pixel* menggunakan *layer* konvolusi dan *maxpooling*

4.3 Model Hasil *Training*

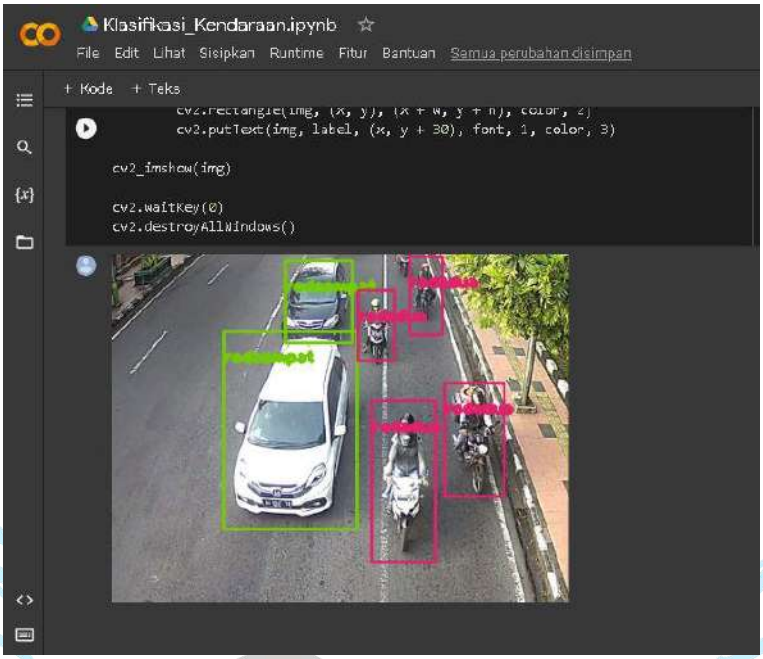
Model hasil *training* menghasilkan satu file model ekstensi weight yang digunakan dalam integrasi aplikasi sistem klasifikasi. File model weight dan file konfigurasi .cfg keduanya terhubung dalam menklasifikasikan objek kendaraan roda dua dan roda empat. Kedua file tersebut terimpan secara otomatis di dalam google drive menggunakan kode program *path destination file*. Gambar 4.11 menunjukkan *file* hasil *model training* yang tersimpan secara otomatis pada google drive.

Gambar 4. 11 File model weight dan cfg yang tersimpan pada drive

Hasil model *training* kemudian selanjutnya diuji dengan menggunakan data gambar uji yang disediakan, sehingga akan menghasilkan *metric evaluate* yaitu *confusion matrix* dan metrik yang digunakan dalam deteksi objek adalah *mean average precision* (mAP).

4.3.1 Hasil Pengujian Model *Training*

Pengujian model *training* menggunakan data gambar uji dengan memanggil fungsi *detector test* dan memanggil file model weight serta konfigurasi cfg sehingga menghasilkan gambar prediksi dari gambar uji. Menggunakan *library* opencv dan memanggil gambar maka akan menghasilkan visual seperti Gambar 4.12



Gambar 4.12 Gambar uji hasil prediksi model *training*

4.3.2 *Confusion Matrix*

Confusion matrix dihasilkan dari pengujian model *training* dengan data gambar uji. *Confusion matrix* digunakan sebagai penentuan nilai *metric evaluate* yaitu *mean average precision* (mAP). *Confusion matrix* yang dihasilkan sebagai berikut:

```
 calculation mAP (mean average precision)...
 Detection layer: 16 - type = 28
 Detection layer: 23 - type = 28
24
 detections_count = 48, unique_truth_count = 34
class_id = 0, name = rodadua, ap = 100.00%       (TP = 11, FP = 0)
class_id = 1, name = rodaempat, ap = 95.01%      (TP = 17, FP = 0)

 for conf_thresh = 0.25, precision = 1.00, recall = 0.82, F1-score = 0.90
 for conf_thresh = 0.25, TP = 28, FP = 0, FN = 6, average IoU = 82.96 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.975057, or 97.51 %
Total Detection Time: 0 Seconds
```

Gambar 4. 13 *Metric evaluate* model *training* terhadap data gambar uji

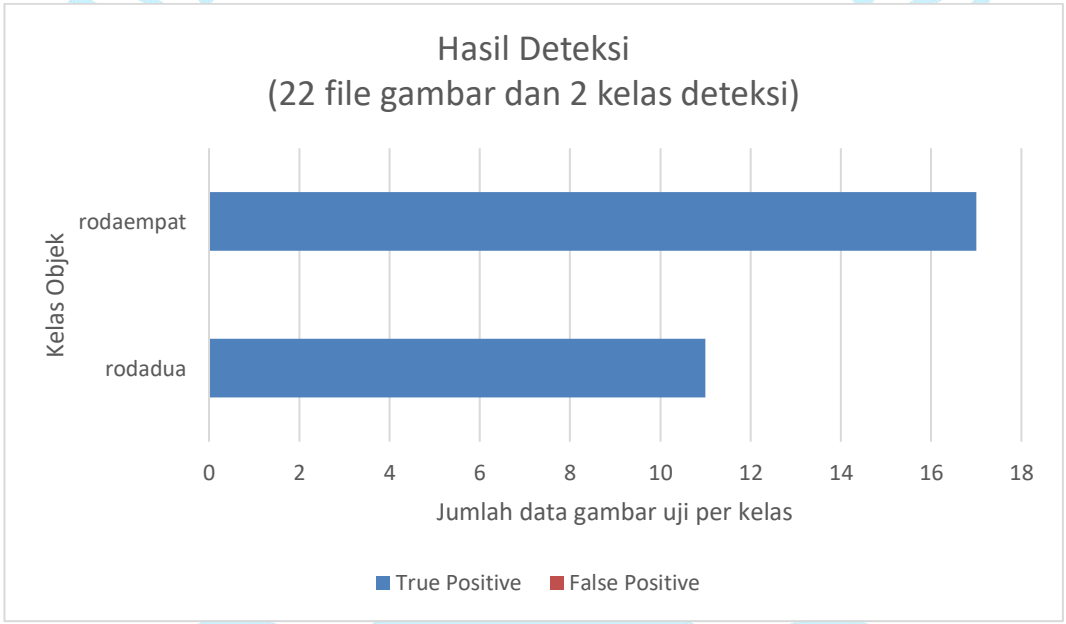
Nilai *confusion matrix* dari kedua kelas dapat dilihat pada Tabel 4.1 dan Gambar 4.14:

Tabel 4. 1 Hasil pengujian model *training* terhadap data gambar uji

[redacted]	*Actual*

		Roda Dua	Roda Empat
Predict	Roda Dua	11	0
	Roda Empat	0	17

Tabel pengujian diatas merupakan tabel *confusion matrix* yang terdiri dari dua kelas kendaraan. Keseluruhan objek yang dideteksi oleh model *training* untuk keseluruhan terdapat 28 dari keseluruhan data gambar uji yaitu 22 data gambar (3 data gambar kendaraan roda dua dan roda empat dalam satu *frame*. Gambar 4.14 menunjukkan grafik hubungan kelas objek roda empat dan roda dua yang diklasifikasikan.



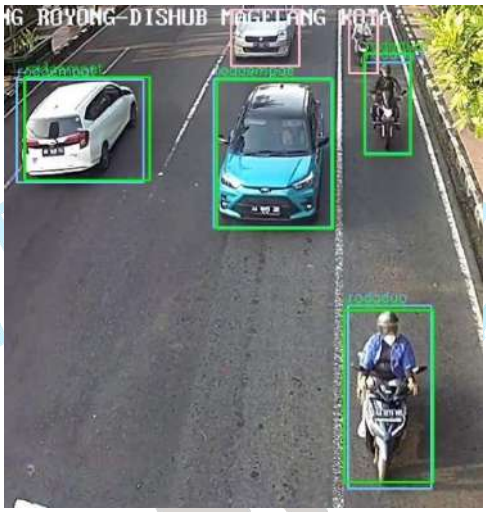
Gambar 4. 14 Grafik hasil deteksi data gambar uji

Metrik evaluasi yangditampilkan oleh detector mAP diperoleh beberapa informasi seperti *ground truth* atau *unique truth* atau objek unik yang dikenali sebagai objek berjumlah 34 objek kendaraan namun hasil deteksi hanya berjumlah 28 objek kendaraan.

4.3.3 *Mean Average Precision* (mAP)

Berdasarkan Gambar 4.13 dapat diketahui bahwa nilai *mean average precision* yang dihasilkan dari pengujian model training terhadap data gambar uji

adalah 97.51%. MAP merupakan metrik evaluasi untuk deteksi objek pada gambar yang memiliki hubungan dengan nilai *Intersection of Union* (IoU). IoU menggambarkan hubungan *ground truth* (objek sebenarnya) dengan *detection truth* (objek yang diprediksi). Gambaran penentuan nilai IoU pada Gambar 2.17:



Gambar 2. 17 IoU yang tampak pada proses training model

Melalui nilai IoU dapat diketahui objek yang diprediksi. Skor prediksi dikonversi ke kelas label sehingga objek dapat diklasifikasikan ke dalam objek kendaraan roda dua atau roda empat. Setelah mengetahui kelas masing – masing objek maka dapat dihitung *confusion matrix* untuk TP, FP, TN dan FN yang digunakan dalam menentukan nilai *precision* serta *recall*. Nilai MAP merupakan perhitungan rata-rata keseluruhan *average precision* (AP) yang diketahui dengan menghitung nilai rata-rata bobot *precision* tiap treshold yang mana nilai bobot meningkat berdasarkan nilai *recall*. Nilai treshold IoU dapat dikonfigurasi secara mandiri semakin kecil maka nilai *precision* dan mAP akan semakin tinggi sehingga deteksi objek kendaraan semakin baik. Penelitian ini menggunakan IoU *treshold* sebesar 0.50.

Nilai mAP dapat ditentukan melalui perhitungan sistem selama model training, karena banyaknya data uji yang digunakan sehingga prediksi dapat dilakukan dengan komputer. Training model yang dilakukan mendapatkan nilai metrik mAP sebesar 97,51%.

4.4 Hasil Pembuatan GUI Aplikasi Klasifikasi dan *Counting* Kendaraan

Aplikasi dirancang menggunakan *PyQT5 Designer* untuk memudahkan dalam pembuatan layout desainnya. GUI dibuat sederhana dengan beberapa fitur utama yaitu fitur tampilan video, fitur penghitung yang terdiri dari roda dua – roda empat untuk sisi atas serta sisi bawah, dan fitur *button* terdiri dari *import file*, *start-import* untuk *webcam*, start untuk *file video*, *stop video* dan *exit* untuk keluar dari aplikasi.

Gambar 4. 15 Tampilan GUI Aplikasi Klasifikasi Kendaraan

4.5 Hasil Pengujian Sistem Klasifikasi

Model training yang sudah dibangun kemudian diintegrasikan dengan aplikasi sistem klasifikasi dengan memanggil konfigurasi .cfg dan model weight. Sampel video yang digunakan diperoleh melalui Dinas Perhubunga Kota Magelang yang diimport kemudian dijalankan, akan terlihat deteksi objek kendaraan dengan klasifikasi kelasnya kemudian jika melewati garis penghitung nilai akan tercatat di bagian *counting.*

Gambar 4. 16 Hasil pengujian aplikasi klasifikasi dan penghitung kendaraan

Pengujian dilakukan dengan melakukan prediksi kendaraan per menit selama total 10 menit dengan waktu awal rekaman CCTV 06:13:34, hasilnya yang didapatkan ditunjukkan pada Tabel 4.4.

Tabel 4. 2 Prediksi Kendaraan menggunakan Aplikasi Klasifikasi dan Penghitung Kendaraan

Pengujian (prediksi)	Waktu	Roda 2			Roda 4			Total Kendaraan
		bawah	atas	total	bawah	atas	total	
1	06:14:34	6	8	14	1	2	3	17
2	06:15:34	15	4	19	7	3	10	29
3	06:16:34	5	10	15	1	4	5	20
4	06:17:34	12	6	18	3	1	4	22
5	06:18:34	15	10	25	5	3	8	33
6	06:19:34	21	6	27	7	4	11	38
7	06:20:34	25	16	41	5	3	8	49
8	06:21:34	12	8	20	3	3	6	26
9	06:22:34	7	3	10	5	3	8	18
10	06:23:34	27	14	41	11	4	15	56
Total		145	85	230	48	30	78	308

Hasil prediksi aplikasi dilihat dari perhitungan yang dilakukan oleh sistem dengan menggunakan garis penghitung acuan. Video rekaman CCTV berdurasi 40 menit terdapat objek kendaraan yang diprediksi yaitu 1007 roda dua bawah, 289 roda empat bawah, 500 roda dua atas dan 167 roda empat atas.

4.3.1 Analisis Pengujian Model Sistem Klasifikasi

Model training yang sudah dibuat kemudian diintegrasikan ke dalam sistem aplikasi klasifikasi da penghitung kendaraan dengan menambahkan konfigurasi .cfg dan model .weight. Garis penghitung yang berada ditengah sebagai acuan untuk menghitung kendaraan yang melintas dari bawah ke atas maupun dari atas ke bawah menggunakan logika sederhana. Analisis model machine learning yang sudah dibuat yag kemudian diintegrasikan dengan aplikasi yaitu dilakukan dengan penghitungan prediksi yang bisa dilakukan oleh aplikasi dengan jumlah kendaraan nyata yang nampak dihitung secara manual. Jumlah kendaraan nyata yang tampak pada video rekaman CCTV ditunjukkan pada Tabel 4.5.

Tabel 4. 3 Hasil penghitungan manual kendaraan

Pengujian (manual)	Waktu	Roda 2			Roda 4			Total Kendaraan
		bawah	atas	total	bawah	atas	total	
1	06:14:34	11	11	22	2	3	5	27
2	06:15:34	15	6	21	7	4	11	32
3	06:16:34	10	15	25	2	4	6	31
4	06:17:34	15	12	27	3	6	9	36
5	06:18:34	17	19	36	5	5	10	46
6	06:19:34	22	17	39	9	5	14	53
7	06:20:34	25	26	51	5	8	13	64
8	06:21:34	14	10	24	7	5	12	36
9	06:22:34	9	11	20	5	3	8	28
10	06:23:34	27	26	53	11	8	19	72
Total		165	153	318	56	51	107	425

Hasil perhitungan manual tersebut dibandingkan dengan perhitungan prediksi aplikasi untuk menentukan nilai akurasi dan *error* dengan *mse* (*mean squared error*). Jumlah total kendaraan dibandingkan sebagai acuan untuk mendapatkan nilai antara sistem klasifikasi dengan perhitungan manual.

Perbandingan dilakukan dengan jumlah total kendaraan antara prediksi aplikasi sistem klasifikasi dengan perhitungan manual aktual dari bawah dan atas secara terpisah. Perbandingan secara terpisah tersebut dilakukan berkaitan dengan posisi kendaraan yang melintas dari atas ke bawah dan dari bawah ke atas yang memiliki pembacaan berbeda. Sehingga perlu dianalisis mendalam untuk menemukan kaitan akurasi dari model yang dibuat. Tabel 4.6 menampilkan perbandingan total kendaraan aktual dengan prediksi yang diambil dari total nilai bawah:

Tabel 4. 4 Perbandingan total kendaraaan aktual dengan prediksi untuk nilai bawah

Pengujian (bawah)	Waktu	Total Kendaraan Aktual	Total Kendaraan Prediksi	Selisih	(Selisih)^2
1	06:14:34	13	7	6	36
2	06:15:34	22	22	0	0
3	06:16:34	12	6	6	36
4	06:17:34	18	15	3	9
5	06:18:34	22	20	2	4
6	06:19:34	31	28	3	9
7	06:20:34	30	30	0	0
8	06:21:34	21	15	6	36
9	06:22:34	14	12	2	4
10	06:23:34	38	38	0	0
Jumlah		221	193	SUM	134

$$MSE\ (Mean\ Squared\ Error) = \frac{134}{10} = 13.4$$

Nilai *mse* yang diperoleh adalah 13.4 yang berarti nilai telah memenuhi ukuran batasan yang diberikan yaitu estimasi dibawah 20, sehingga model *training* telah memenuhi tujuan penelitian dalam memperbaiki sistem penelitian terdahulu.

Tabel 4. 5 Rata – rata akurasi prediksi kendaraan nilai bawah

Pengujian (bawah)	Waktu	Total Kendaraan Aktual	Total Kendaraan Prediksi	akurasi (prediksi / aktual)	akurasi % (akurasi x 100%)
1	06:14:34	13	7	0,538462	53,8461538
2	06:15:34	22	22	1	100
3	06:16:34	12	6	0,5	50
4	06:17:34	18	15	0,833333	83,3333333
5	06:18:34	22	20	0,909091	90,9090909
6	06:19:34	31	28	0,903226	90,3225806
7	06:20:34	30	30	1	100
8	06:21:34	21	15	0,714286	71,4285714
9	06:22:34	14	12	0,857143	85,7142857
10	06:23:34	38	38	1	100
Rata - rata akurasi %					82,5554016

Terlihat error yang didapatkan menggunakan mse adalah sebesar 13.4 ukuran yang masih bisa dikatakan cukup baik dan akurasi yang baik sekitar 82,56%. Perbandingan juga dilakukan pada nilai atas yang merupakan jumlah kendaraan yang dihitung dari bawah ke atas ditunjukkan pada Tabel 4.8:

Tabel 4. 6 Perbandingan total kendaraaan aktual dengan prediksi untuk nilai bawah

Pengujian (atas)	Waktu	Total Kendaraan Aktual	Total Kendaraan Prediksi	Selisih	(Selisih)^2
1	06:14:34	14	10	4	16
2	06:15:34	10	7	3	9
3	06:16:34	19	14	5	25
4	06:17:34	18	7	11	121
5	06:18:34	24	13	11	121
6	06:19:34	22	10	12	144
7	06:20:34	34	19	15	225
8	06:21:34	15	11	4	16
9	06:22:34	14	6	8	64
10	06:23:34	34	18	16	256
Jumlah		204	115	SUM	997

$$MSE\ (Mean\ Squared\ Error) = \frac{997}{10} = 99.7$$

Nilai *mse* yang diperoleh pada sistem perhitungan kendaraan ke arah atas adalah sebesar 99.7 yang berarti nilai tidak memenuhi ukuran batasan yang diberikan yaitu estimasi dibawah 20 bahkan sangat jauh dari harapan, sehingga model *training* untuk menghitung kendaraan ke arah atas perlu ditingkatkan hal ini dapat masuk dalam bagian saran untuk peningkatan performa sistem yang lebih baik. Karena sistem perhitungan kendaraan nilai bawah dan nilai atas sangat berpengaruh satu sama lain.

Tabel 4. 7 Rata – rata akurasi prediksi kendaraan nilai atas

Pengujian (bawah)	Waktu	Total Kendaraan Aktual	Total Kendaraan Prediksi	akurasi (prediksi / aktual)	akurasi % (akurasi x 100%)
1	06:14:34	14	10	0,714286	71,4285714
2	06:15:34	10	7	0,7	70
3	06:16:34	19	14	0,736842	73,6842105
4	06:17:34	18	7	0,388889	38,8888889
5	06:18:34	24	13	0,541667	54,1666667
6	06:19:34	22	10	0,454545	45,4545455
7	06:20:34	34	19	0,558824	55,8823529
8	06:21:34	15	11	0,733333	73,3333333
9	06:22:34	14	6	0,428571	42,8571429
10	06:23:34	34	18	0,529412	52,9411765
Rata - rata akurasi %					57,8636889

Error yang didapatkan melalui mse untuk kendaraan yang melintas menuju atas adalah 99,7 yang sangat besar dan juga akurasinya sekitar 57,86%. Analisis dari ketiga perbandingan model training, nilai bawah dan nilai atas berdasarkan tabel berikut:

Tabel 4. 8 Perbandingan aplikasi sistem nilai bawah, nilai atas da model training

Aplikasi Sistem (bawah)		Aplikasi Sistem (atas)		Model Training	
mse	akurasi	mse	akurasi	map	akurasi
13.4	82.56%	99.7	57.86%	97.5%	82.35%

Melalui tabel perbandingan tersebut diketahui bahwa penerapan model training machine learning nilai akurasinya hampir sebanding dengan nilai akurasi sistem nilai bawah dengan kata lain penerapan model training yang diintegrasikan ke aplikasi berjalan dengan sangat baik. Sedangkan prediksi aplikasi sistem nilai atas menghasilkan akurasi yang kecil sehingga probabilitas memprediksi objek kendaraan hanya satu banding dua dari objek kendaraan yang ada. Hal tersebut dapat terjadi terutama jumlah data gambar yang di training dan pembacaan model training terhadap tiap *frame* rekaman CCTV dimana diambil dari sudut satu sisi jalan.

Pembaharuan sistem yang dilakukan pada penelitian ini adalah pada sistem yang memprediksi serta menghitung dua ruas jalan dari arah atas dan arah bawah dan juga pengujian yang dilakukan per menit dengan data yang lebih banyak yaitu 10 pengujian aplikasi dalam menghitung objek kenaraan yang dilakukan selama 10 menit. Akurasi yang didapatkan dari seluruh total pengujian 82% berpengaruh pada objek yang diklasifikasi secara baik dan metrik evaluasi yang digunakan adalah mAP (*mean average error*) yaitu 97% yang memberikan pengaruh pada ketepatan kebenaran objek asli yang diklasifikasi.

4.3.2 Analisis Hasil Evaluasi Sistem Klasifikasi terhadap Penelitian Terdahulu

Hasil evaluasi sistem dapat dilihat melalui Tabel 4.8, sistem klasifikasi memiliki akurasi penghitungan nilai bawah sebesar 82,56% dengan memiliki fitur penghitungan nilai atas dengan akurasi yang masih dikembangkan sebesar 57,86%, dari hasil tersebut dihubungkan dengan model training memiliki akurasi yang sebanding yaitu sebesar 82,35% dengan metrik MAP untuk objek deteksi sebesar 97,5%.

Jika dibandingkan dengan penelitian yang dilakukan oleh Al Okaishi (2020) akurasi rata-rata seluruh obejk kendaraan yang diprediksi sebesar 95% yang hanya dihitung satu arah dengan total kendaraan yang dihitung sebanyak 106 buah. Penelitian yang dilakukan Pramana, dkk (2020) dengan pengambilan satu arah menggunakan 100 buah data uji algoritma CNN Lenet menghasilkan akurasi sebesar 86%. Terlihat pada Tabel 4.9:

Tabel 4. 9 Perbandingan Aplikasi Sistem Klasifikasi dengan Penelitian Terdahulu

Penelitian	Algoritma	Akurasi	Arah Klasifikasi	Jumlah Data Uji
Abid (2022)	YOLOv3	82%	2	425
Al Okaishi, dkk (2020)	CNN	95%	1	106
Pramana, dkk (2020)	CNN Lenet	86%	1	100

Pengembangan sistem klasifikasi yang diimplementasikan ke dalam aplikasi dengan menggunakan algoritma YOLO untuk dua arah klasifikasi yang berbeda lebih efektif dan efisien dengan mempertahankan akurasi diatas 80% yang mampu mempertahankan model *machine learning* untuk data yang lebih banyak.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian beserta pengujian yang sudah dilakukan, penelitian "Perancangan Sistem *Machine Learning* untuk Klasifikasi Kendaraan pada Persimpangan *Traffic Light*", penulis menarik kesimpulan, bahwa:

1. Performa sistem machine learning yang diterpkan ke dalam aplikasi dapat berjalan dengan baik. Hal tersebut dibuktikan dari nilai akurasi 82,56% dengan estimasi nilai *error* mse (*mean squared error*) sebesar 13,4 serta metrik evaluasi untuk objek deteksi menggunakan mAP (*mean average precision*) didapatkan nilai yang sangat tinggi dalam menklasifikasikan kendaraan yaitu 97,5%, dan
2. aplikasi yang dibuat memiliki fitur menghitung kendaraan dari dua arah yang berbeda dengan pengujian terhadap 425 data gambar uji yang nilai akurasinya dipertahankan sebesar 82% dari proses training model dengan aplikasi sistem klasifikasi yang apabila dibandingkan dengan penelitian terdahulu masih lebih baik aplikai pada penelitian ini.

DAFTAR PUSTAKA

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., 2016. Tensorflow: Large-Scale Machine Learning On Heterogeneous Distributed Systems. Arxiv Preprint Arxiv:1603.04467.

Al Okaishi, W., Zaarane, A., Slimani, I., Atouf, I., Benrabh, M., 2019. A Traffic Surveillance System In Real-Time To Detect And Classify Vehicles By Using Convolutional Neural Network. Presented At The 2019 International Conference On Systems Of Collaboration Big Data, Internet Of Things & Security (Syscobiots), Ieee, Pp. 1–5.

Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M.S., Van Esesn, B.C., Awwal, A.A.S., Asari, V.K., 2018. The History Began From Alexnet: A Comprehensive Survey On Deep Learning Approaches. Arxiv 2018. Arxiv Preprint Arxiv:1803.01164.

Bps, 2018. Data Jumlah Kendaraan Bps. Badan Pusat Staistik.

Brahmbhatt, S., 2013. Practical Opencv. Apress.

Budiharto, W., 2018. Pemrograman Python Untuk Ilmu Komputer Dan Teknik. Andi (Anggota Ikapi).

Carneiro, T., Da Nóbrega, R.V.M., Nepomuceno, T., Bian, G.-B., De Albuquerque, V.H.C., Reboucas Filho, P.P., 2018. Performance Analysis Of Google Colaboratory As A Tool For Accelerating Deep Learning Applications. Ieee Access 6, 61677–61685.

Davies, E.R., 2017. Computer Vision: Principles, Algorithms, Applications, Learning. Academic Press.

Dewi, S.R., 2018. Deep Learning Object Detection Pada Video Menggunakan Tensorflow Dan Convolutional Neural Network.

Dey, S., 2018. Hands-On Image Processing With Python: Expert Techniques For Advanced Image Analysis And Effective Interpretation Of Image Data. Packt Publishing Ltd.

Fadlia, N., Kosasih, R., 2020. Klasifikasi Jenis Kendaraan Menggunakan Metode Convolutional Neural Network (Cnn). Jurnal Ilmiah Teknologi Dan Rekayasa 24, 207–215.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. Mit Press.

Harahap, M., Elfrida, J., Agusman, P., Rafael, M., Abram, R., Andrianto, K., 2019. Sistem Cerdas Pemantauan Arus Lalu Lintas Dengan Dcn (Deep Convolutional Network), In: Seminar Nasional Aptikom (Semnastik) 2019. Pp. 367–376.

Harani, N.H., Prianto, C., Hasanah, M., 2019. Deteksi Objek Dan Pengenalan Karakter Plat Nomor Kendaraan Indonesia Menggunakan Metode Convolutional Neural Network (Cnn) Berbasis Python. Jurnal Teknik Informatika 11, 47–53.

Harris, C.R., Millman, K.J., Van Der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., 2020. Array Programming With Numpy. Nature 585, 357–362.

Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep Learning. Nature 521, 436–444.

Ma'ali, A.M., Hendriyawan A, M.S., 2019. Rancang Bangun Sistem Pengendali Lampu Lalu Lintas Berdasarkan Pengenalan Citra Digital Kendaraan Menggunakan Metode Faster R-Cnn (Phd Thesis). University Of Technology Yogyakarta.

Müller, A.C., Guido, S., 2016. Introduction To Machine Learning With Python: A Guide For Data Scientists. O'reilly Media, Inc.

Nurolan, A.T., 2020. Deteksi Dan Klasifikasi Jenis Kendaraan Berbasis Pengolahan Citra Dengan Metode Convolutional Neural Network (Cnn).

Redmon, J., Farhadi, A., 2018. Yolov3: An Incremental Improvement. Arxiv Preprint Arxiv:1804.02767.

Setiawan, T., 2020. Implementasi Convolutional Neural Network Untuk Pengenalan Warna Kendaraan.

Sonka, M., Hlavac, V., Boyle, R., 2014. Image Processing, Analysis, And Machine Vision. Cengage Learning.

Suyanto, D., 2017. Data Mining Untuk Klasifikasi Dan Klasterisasi Data. Bandung: Informatika Bandung.

LAMPIRAN

Lampiran 1. Surat Permohonan Izin Survei

RISET, DAN TEKNOLOGI
UNIVERSITAS TIDAR
FAKULTAS TEKNIK
Alamat : Jalan Kapten Suparman 39 Magelang 56116
Telp. (0293) 364113 Fax. (0293) 362438
Laman : www.untidar.ac.id Surel : teknik@untidar.ac.id

Nomor : T/769/UN57-F5/KM-07/2022 Magelang, 23 Maret 2021
Lampiran : 1 Berkas
Hal : Surat Permohonan Izin Survey

Kepada Yth. Kepala Dinas Perhubungan Kota Magelang
Jl. Jendral Sudirman No. 84, Tidar Selatan, Kecamatan Magelang Selatan,
Kota Magelang, Jawa Tengah 56125.

Disampaikan dengan hormat, dalam rangka kegiatan Penelitian Skripsi mahasiswa Jurusan Teknik Elektro, Fakultas Teknik, Universitas Tidar, maka dengan ini kami mohon kepada Bapak/Ibu untuk mengizinkan mahasiswa tersebut dibawah ini, untuk melakukan survey di perusahaan yang Bapak/Ibu pimpin, mahasiswa atas nama:

Nama	:	Abid Juliant Indraswara
NPM	:	1810501059
Jurusan	:	Teknik Elektro
No.Telepon	:	088802739605
Judul Skripsi	:	Sistem Klasifikasi Kendaraan Pada Persimpangan *Traffic Light* Menggunakan Optimalisasi Layer Algoritma

Bersamaan dengan surat ini, kami lampirkan proposal skripsi dan lingkup data yang diharapkan. Demikian permohonan ini kami sampaikan, atas perkenaan dan kerja sama Bapak/Ibu diucapkan terima kasih.

Dekan,

Ir. Sarto Nisworo, M.T., IPU
NIP. 19590928199103 1001

Lampiran 2. Lembar Disposisi Dinas Perhubungan Kota Magelang

KOTA MAGELANG

PEMERINTAH KOTA MAGELANG

DINAS PERHUBUNGAN

Jl. Jend. Sudirman No. 84 Telp. (0293) 362205

MAGELANG 56125

LEMBAR DISPOSISI

Surat dari	: UNTID	**Diterima tgl**	: 21-3-22
Tanggal Surat	: 23-4.22	**No. Agenda**	:
No. Surat	: T / 769 / ... / UM.07	**Diteruskan kepada**	:

Sekretaris
Kabid LL dan Perparkiran ✓
Kabid PKB, Angkutan & Terminal

Ytk. Kasie ... & ...

- ... dgn saran kp. Kadis
- Hubungi ... jalan

08/05-22

Hubungi YGS → untuk fasilitasi dan agendakan ketemu saya.

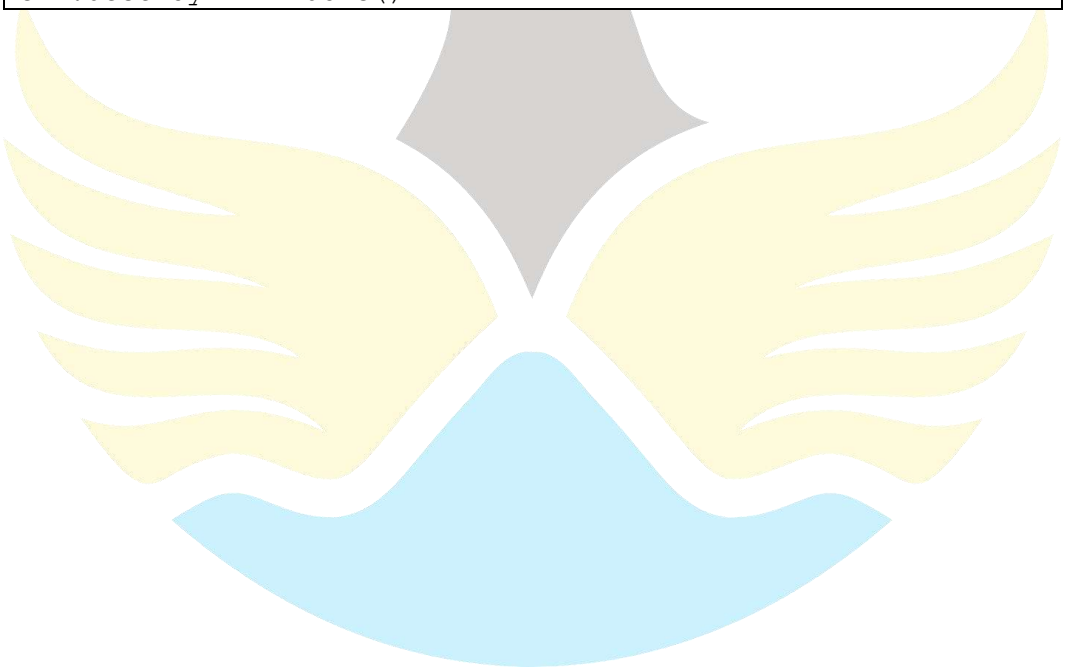
Lampiran 3. Kode Program Training Model Machine Learning

```
# Cek NVIDIA CUDA
!nvidia-smi

# Menghubungkan colab dengan gdrive
from google.colab import drive
drive.mount('/content/gdrive')

!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive

# Clone Darknet
!git clone https://github.com/AlexeyAB/darknet

# Konfigurasi Darkent untuk penggunaan Algoritma Yolov3
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile

# Compile Konfigurasi Darknet
!make

# Copy file yolov3.cfg untuk digunakan dalam model
!cp cfg/yolov3-tiny.cfg cfg/yolov3_training.cfg

# Mengubah beberapa parameter layer untuk penyesuaian model y
ang lebih baik
!sed -i 's/batch=1/batch=64/' cfg/yolov3_training.cfg
!sed -
i 's/subdivisions=1/subdivisions=16/' cfg/yolov3_training.cfg
!sed -
i 's/max_batches = 500200/max_batches = 4000/' cfg/yolov3_tra
ining.cfg
!sed -i '610 s@classes=80@classes=2@' cfg/yolov3_training.cfg
!sed -i '696 s@classes=80@classes=2@' cfg/yolov3_training.cfg
!sed -i '783 s@classes=80@classes=2@' cfg/yolov3_training.cfg
!sed -
i '603 s@filters=255@filters=21@' cfg/yolov3_training.cfg
!sed -
i '689 s@filters=255@filters=21@' cfg/yolov3_training.cfg
!sed -
i '776 s@filters=255@filters=21@' cfg/yolov3_training.cfg
```

```
# Membuat file direktori data training
!echo -e 'rodadua\nrodaempat' > data/obj.names
!echo -
e 'classes = 2 \ntrain = data/train.txt \nvalid = data/test.t
xt\n names = data/obj.names\n backup = /content/gdrive/MyDriv
e/penelitian-yolo-skripsi/evaluate-mAP' > data/obj.data

# Save model konfigurasi yolov3_training.cfg
!cp cfg/yolov3_training.cfg /content/gdrive/MyDrive/penelitia
n-yolo-skripsi/evaluate-mAP/yolov3_training.cfg
!cp data/obj.names /content/gdrive/MyDrive/penelitian-yolo-
skripsi/evaluate-mAP/classes.txt

# Unzip Dataset Gambar Kendaraan
!mkdir data/obj
!unzip /content/gdrive/MyDrive/penelitian-yolo-
skripsi/evaluate-mAP/kendaraan.zip -d data/obj

# train_test_split data gambar kendaraan
import glob
import os

image_list = '/content/darknet/data/obj/kendaraan-2'

percentage_test = 10

file_train = open('data/train.txt', 'w')
file_test = open('data/test.txt', 'w')

counter = 1
index_test = round(100/percentage_test)
for file in glob.iglob(os.path.join(image_list, '*.JPG')):
  title, ext = os.path.splitext(os.path.basename(file))
  if counter == index_test:
    counter = 1
    file_test.write(image_list + "/" + title + '.JPG' + "\n")
  else:
    file_train.write(image_list + "/" + title + '.JPG' + "\n"
)
    counter = counter + 1

# Donwload pre-
trained weights untuk mengambil layer konvolusi
!wget https://pjreddie.com/media/files/darknet53.conv.74
```

```
# Mulai Training Model
!./darknet detector train data/obj.data cfg/yolov3_training.c
fg darknet53.conv.74 -dont_show

# Mencari Yolov3 MAP Evaluate
!./darknet detector mAP /content/darknet/data/obj.data /conte
nt/darknet/cfg/yolov3_training.cfg /content/gdrive/MyDrive/pe
nelitian-yolo-skripsi/evaluate-
mAP/yolov3_training_last.weights

# Mencari Yolov3 Recall
!./darknet detector recall /content/darknet/data/obj.data /co
ntent/darknet/cfg/yolov3_training.cfg /content/gdrive/MyDrive
/penelitian-yolo-skripsi/evaluate-
mAP/yolov3_training_last.weights

# hasil deteksi menggunakan data gambar uji
!./darknet detector test /content/darknet/data/obj.data /cont
ent/darknet/cfg/yolov3_training.cfg /content/gdrive/MyDrive/p
enelitian-yolo-skripsi/evaluate-
mAP/yolov3_training_last.weights -dont_show -
ext_output < data/test.txt > data/result.txt

# Testing model training dengan data gambar uji
from google.colab.patches import cv2_imshow
#%matplotlib inline
#from matplotlib import pyplot as plt
class_ids = []
confidences = []
boxes = []
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:
            # Deteksi objek
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
```

```
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
#print(indexes)
font = cv2.FONT_HERSHEY_PLAIN
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        color = colors[class_ids[i]]
        cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
        cv2.putText(img, label, (x, y + 30), font, 1, color,
3)

cv2_imshow(img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



Lampiran 4. Kode Program Pengujian Model *Training*

```
import cv2
import numpy as np
import time
import collections
from tracker import *

tracker = EuclideanDistTracker()
input_size = 320

# Menentukan batasan nilai
confThreshold = 0.2
nmsThreshold = 0.2

font_color = (0, 0, 255)
font_size = 0.5
font_thickness = 2

# Garis penghitung
middle_line_position = 150
up_line_position = middle_line_position - 15
down_line_position = middle_line_position + 15

# List kelas nama
classesFile = "obj.names"
classNames = open(classesFile).read().strip().split('\n')
print(classNames)
print(len(classNames))

required_class_index = [0, 1]

detected_classNames = []

model_Config = 'yolov3_testing_100sampl.cfg'
model_Weights = 'yolov3_training_last_100sampl.weights'

net = cv2.dnn.readNetFromDarknet(model_Config, model_Weights)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableBackend(cv2.dnn.DNN_TARGET_CPU)

np.random.seed(42)
colors = np.random.randint(0, 255, size=(len(classNames), 3), dtype='uint8')

def find_center(x, y, w, h):
    x1=int(w/2)
```

```
    y1=int(h/2)
    cx = x+x1
    cy=y+y1
    return cx, cy

temp_up_list = []
temp_down_list = []
up_list = [0, 0]
down_list = [0, 0]

# Fungsi penghitung kendaraan
def count_vehicle(box_id, img):

    x, y, w, h, id, index = box_id

    center = find_center(x, y, w, h)
    ix, iy = center

    if (iy > up_line_position) and (iy < middle_line_position):

        if id not in temp_up_list:
            temp_up_list.append(id)

    elif iy < down_line_position and iy > middle_line_position:
        if id not in temp_down_list:
            temp_down_list.append(id)

    elif iy < up_line_position:
        if id in temp_down_list:
            temp_down_list.remove(id)
            up_list[index] = up_list[index]+1

    elif iy > down_line_position:
        if id in temp_up_list:
            temp_up_list.remove(id)
            down_list[index] = down_list[index] + 1

    cv2.circle(img, center, 2, (0, 0, 255), -1)

# Membentuk bounding box
def postProcess(outputs,img):
    global detected_classNames
    height, width = img.shape[:2]
    boxes = []
    classIds = []
    confidence_scores = []
```

```
    detection = []
    for output in outputs:
        for det in output:
            scores = det[5:]
            classId = np.argmax(scores)
            confidence = scores[classId]
            if classId in required_class_index:
                if confidence > confThreshold:
                    # print(classId)
                    w,h = int(det[2]*width) , int(det[3]*height)
                    x,y = int((det[0]*width)-w/2) , int((det[1]*height)-h/2)
                    boxes.append([x,y,w,h])
                    classIds.append(classId)
                    confidence_scores.append(float(confidence))

    indices = cv2.dnn.NMSBoxes(boxes, confidence_scores, confThreshold,
nmsThreshold)

    if len(indices) > 0:
        for i in indices.flatten():
            x, y, w, h = boxes[i][0], boxes[i][1], boxes[i][2], boxes[i][3]

            color = [int(c) for c in colors[classIds[i]]]
            name = classNames[classIds[i]]
            detected_classNames.append(name)
            cv2.putText(img,f'{name.upper()}',
                    (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)

            cv2.rectangle(img, (x, y), (x + w, y + h), color, 1)
            detection.append([x, y, w, h, required_class_index.index(classIds[i])])

    boxes_ids = tracker.update(detection)
    for box_id in boxes_ids:
        count_vehicle(box_id, img)

# Variabel video
cap = cv2.VideoCapture('traffic-3.mp4')

# Fungsi membaca realtime – video berjalan
def realTime():
    while True:
        success, img = cap.read()
        img = cv2.resize(img,(0,0),None,0.5,0.5)
        ih, iw, channels = img.shape
        blob = cv2.dnn.blobFromImage(img, 1 / 255, (input_size, input_size), [0,
0, 0], 1, crop=False)
```

```
        net.setInput(blob)
        layersNames = net.getLayerNames()
        outputNames = [(layersNames[i[0] - 1]) for i in
net.getUnconnectedOutLayers()]

        outputs = net.forward(outputNames)

        postProcess(outputs,img)

        cv2.line(img, (0, middle_line_position), (iw, middle_line_position), (255,
0, 255), 2)
        cv2.line(img, (0, up_line_position), (iw, up_line_position), (0, 0, 255), 2)
        cv2.line(img, (0, down_line_position), (iw, down_line_position), (0, 0,
255), 2)

        cv2.putText(img, "Atas", (110, 20), cv2.FONT_HERSHEY_SIMPLEX,
font_size, font_color, font_thickness)
        cv2.putText(img, "Jumlah", (160, 20), cv2.FONT_HERSHEY_SIMPLEX,
font_size, font_color, font_thickness)
        cv2.putText(img, "Roda 2 :            "+ str(down_list[0]), (20, 40),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        cv2.putText(img, "Roda 4 :            "+ str(down_list[1]), (20, 60),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        cv2.putText(img, "Mobil :             "+ str(down_list[0]), (20, 40),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        cv2.putText(img, "Motor :             "+ str(down_list[1]), (20, 60),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        cv2.putText(img, "Bus   :             "+ str(down_list[2]), (20, 80),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        cv2.putText(img, "Truk  :             "+ str(down_list[3]), (20, 100),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)

        cv2.imshow(', img)

        if cv2.waitKey(1) == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

if __name__ == '__main__':
    realTime()
```

Lampiran 5. Kode Program UI Aplikasi Klasifikasi

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file
'ui_main_window_fix_atas.ui'
#
# Created by: PyQt5 UI code generator 5.9.2
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(1127, 668)
        self.image_label = QtWidgets.QLabel(Form)
        self.image_label.setGeometry(QtCore.QRect(10, 60, 921, 581))
        self.image_label.setFrameShape(QtWidgets.QFrame.Box)
        self.image_label.setAlignment(QtCore.Qt.AlignCenter)
        self.image_label.setObjectName("image_label")
        self.label = QtWidgets.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(330, 10, 441, 51))
        font = QtGui.QFont()
        font.setPointSize(14)
        font.setBold(True)
        font.setWeight(75)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.control_bt_5 = QtWidgets.QPushButton(Form)
        self.control_bt_5.setGeometry(QtCore.QRect(950, 400, 161, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.control_bt_5.setFont(font)
        self.control_bt_5.setObjectName("control_bt_5")
        self.control_bt_2 = QtWidgets.QPushButton(Form)
        self.control_bt_2.setGeometry(QtCore.QRect(950, 480, 161, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.control_bt_2.setFont(font)
        self.control_bt_2.setObjectName("control_bt_2")
        self.frame = QtWidgets.QFrame(Form)
        self.frame.setGeometry(QtCore.QRect(940, 140, 171, 181))
        self.frame.setFrameShape(QtWidgets.QFrame.Panel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
```

```
        self.frame.setLineWidth(5)
        self.frame.setObjectName("frame")
        self.image_label_2 = QtWidgets.QLabel(self.frame)
        self.image_label_2.setGeometry(QtCore.QRect(19, 90, 51, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.image_label_2.setFont(font)
        self.image_label_2.setObjectName("image_label_2")
        self.image_label_4 = QtWidgets.QLabel(self.frame)
        self.image_label_4.setGeometry(QtCore.QRect(75, 90, 41, 20))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.image_label_4.setFont(font)
        self.image_label_4.setAlignment(QtCore.Qt.AlignCenter)
        self.image_label_4.setObjectName("image_label_4")
        self.image_label_5 = QtWidgets.QLabel(self.frame)
        self.image_label_5.setGeometry(QtCore.QRect(75, 120, 41, 20))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.image_label_5.setFont(font)
        self.image_label_5.setAlignment(QtCore.Qt.AlignCenter)
        self.image_label_5.setObjectName("image_label_5")
        self.image_label_3 = QtWidgets.QLabel(self.frame)
        self.image_label_3.setGeometry(QtCore.QRect(20, 120, 51, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.image_label_3.setFont(font)
        self.image_label_3.setObjectName("image_label_3")
        self.label_2 = QtWidgets.QLabel(self.frame)
        self.label_2.setGeometry(QtCore.QRect(50, 10, 71, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_2.setFont(font)
        self.label_2.setFrameShape(QtWidgets.QFrame.Box)
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setObjectName("label_2")
        self.image_label_6 = QtWidgets.QLabel(self.frame)
        self.image_label_6.setGeometry(QtCore.QRect(120, 90, 41, 20))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.image_label_6.setFont(font)
        self.image_label_6.setAlignment(QtCore.Qt.AlignCenter)
        self.image_label_6.setObjectName("image_label_6")
        self.image_label_7 = QtWidgets.QLabel(self.frame)
        self.image_label_7.setGeometry(QtCore.QRect(120, 120, 41, 20))
        font = QtGui.QFont()
```

```
        font.setPointSize(10)
        self.image_label_7.setFont(font)
        self.image_label_7.setAlignment(QtCore.Qt.AlignCenter)
        self.image_label_7.setObjectName("image_label_7")
        self.image_label_8 = QtWidgets.QLabel(self.frame)
        self.image_label_8.setGeometry(QtCore.QRect(120, 70, 41, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.image_label_8.setFont(font)
        self.image_label_8.setAlignment(QtCore.Qt.AlignCenter)
        self.image_label_8.setObjectName("image_label_8")
        self.image_label_9 = QtWidgets.QLabel(self.frame)
        self.image_label_9.setGeometry(QtCore.QRect(70, 70, 51, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.image_label_9.setFont(font)
        self.image_label_9.setAlignment(QtCore.Qt.AlignCenter)
        self.image_label_9.setObjectName("image_label_9")
        self.control_bt_4 = QtWidgets.QPushButton(Form)
        self.control_bt_4.setGeometry(QtCore.QRect(950, 360, 161, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.control_bt_4.setFont(font)
        self.control_bt_4.setObjectName("control_bt_4")
        self.control_bt_3 = QtWidgets.QPushButton(Form)
        self.control_bt_3.setGeometry(QtCore.QRect(950, 520, 161, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.control_bt_3.setFont(font)
        self.control_bt_3.setObjectName("control_bt_3")
        self.control_bt = QtWidgets.QPushButton(Form)
        self.control_bt.setGeometry(QtCore.QRect(950, 440, 161, 31))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.control_bt.setFont(font)
        self.control_bt.setObjectName("control_bt")
        self.image_label_10 = QtWidgets.QLabel(Form)
        self.image_label_10.setGeometry(QtCore.QRect(10, 640, 261, 16))
        font = QtGui.QFont()
        font.setPointSize(7)
        self.image_label_10.setFont(font)
        self.image_label_10.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.Ali
gnLeft|QtCore.Qt.AlignVCenter)
        self.image_label_10.setObjectName("image_label_10")

        self.retranslateUi(Form)
```

```
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Aplikasi Klasifikasi
Kendaraan"))
        self.image_label.setText(_translate("Form", "Webcam or Video"))
        self.label.setText(_translate("Form", "Aplikasi Klasifikasi dan Penghitung
Kendaraan"))
        self.control_bt_5.setText(_translate("Form", "Start - Import"))
        self.control_bt_2.setText(_translate("Form", "Stop"))
        self.image_label_2.setText(_translate("Form", "Roda 2 :"))
        self.image_label_4.setText(_translate("Form", "0"))
        self.image_label_5.setText(_translate("Form", "0"))
        self.image_label_3.setText(_translate("Form", "Roda 4 :"))
        self.label_2.setText(_translate("Form", "Counting"))
        self.image_label_6.setText(_translate("Form", "0"))
        self.image_label_7.setText(_translate("Form", "0"))
        self.image_label_8.setText(_translate("Form", "Atas"))
        self.image_label_9.setText(_translate("Form", "Bawah"))
        self.control_bt_4.setText(_translate("Form", "Import"))
        self.control_bt_3.setText(_translate("Form", "Exit"))
        self.control_bt.setText(_translate("Form", "Start"))
        self.image_label_10.setText(_translate("Form", "Created by : Abid Juliant
Indraswara, Universitas Tidar"))
```

Lampiran 6. Kode Program Aplikasi Klasifikasi Kendaraan Integrasi Mode *Training*

```
### --- Created by ---------- ###
### --- Abid Juliant Indraswara --- ###
### --- 1810501059 ---------- ###

# import system module
from fileinput import filename
from logging import FileHandler
import sys

# import beberaoa PyQt5 modules
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QWidget
from PyQt5.QtGui import QImage
from PyQt5.QtGui import QPixmAP
from PyQt5.QtCore import QTimer

# import Opencv, Numpy and file Tracker
import cv2
import os
from isort import file
import numpy as np
from tracker import *

# import pyqt5 ui
from ui_main_window_fix_atas import *

# input variabel tracker dan input_size
tracker = EuclideanDistTracker()
input_size = 320

# batasan nilai penghitung kendaraan
confThreshold = 0.2
nmsThreshold = 0.2

# variabel font
font_color = (0, 0, 255)
font_size = 0.5
font_thickness = 2
```

```
# variabel garis penghitung
middle_line_position = 200
up_line_position = middle_line_position - 15
down_line_position = middle_line_position + 15

# variabel daftar kelas klasifikasi kendaraan
classesFile = r"C:\Users\Lenovo\Documents\Project Jupyter\aplikasi-
counting\obj.names"
classNames = open(classesFile).read().strip().split('\n')
print(classNames)
print(len(classNames))

# index kelas klasifikasi
required_class_index = [0, 1]
detected_classNames = []

# variabel model Yolov3
model_Config = r"C:\Users\Lenovo\Documents\Project Jupyter\aplikasi-
counting\yolov3_testing_100sampl.cfg"
model_Weights = r"C:\Users\Lenovo\Documents\Project Jupyter\aplikasi-
counting\yolov3_training_final_100sampl.weights"

net = cv2.dnn.readNetFromDarknet(model_Config, model_Weights)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableBackend(cv2.dnn.DNN_TARGET_CPU)

# random color untuk font dan garis penghitung
np.random.seed(42)
colors = np.random.randint(0, 255, size=(len(classNames), 3), dtype='uint8')

# fungsi mencari nilai tengah objek
def find_center(x, y, w, h):
    x1=int(w/2)
    y1=int(h/2)
    cx = x+x1
    cy=y+y1
    return cx, cy

# variabel daftar nilai penghitung
temp_up_list = []
temp_down_list = []
up_list = [0, 0]
down_list = [0, 0]

class MainWindow(QWidget):
    # class constructor
```

```
    def __init__(self):
        # call QWidget constructor
        super().__init__()
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        # create a timer
        self.timer = QTimer()
        # set timer timeout callback function
        self.timer.timeout.connect(self.viewCam)
        # set control button callback clicked  function
        self.ui.control_bt.clicked.connect(self.controlTimer)
        self.ui.control_bt_2.clicked.connect(self.stopTimer)
        self.ui.control_bt_3.clicked.connect(self.exitButtonClick)
        self.ui.control_bt_4.clicked.connect(self.importButtonClick)
        self.ui.control_bt_5.clicked.connect(self.controlTimer2)

    # view camera
    def viewCam(self):
        # read image in BGR format
        ret, image = self.cap.read()
        # convert image to RGB format
        #image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image = cv2.resize(image,(0,0),None,0.5,0.5)
        # get image infos
        height, width, channel = image.shape
        step = channel * width
        blob = cv2.dnn.blobFromImage(image, 1 / 255, (input_size, input_size),
[0, 0, 0], 1, crop=False)

        net.setInput(blob)
        layersNames = net.getLayerNames()
        outputNames = [(layersNames[i[0] - 1]) for i in
net.getUnconnectedOutLayers()]

        outputs = net.forward(outputNames)

        self.postProcess(outputs,image)

        cv2.line(image, (0, middle_line_position), (width, middle_line_position),
(255, 0, 255), 2)
        cv2.line(image, (0, up_line_position), (width, up_line_position), (0, 0,
255), 2)
        cv2.line(image, (0, down_line_position), (width, down_line_position), (0,
0, 255), 2)
```

```
        #cv2.putText(image, "Jumlah", (160, 20),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        #cv2.putText(image, "Roda 2 :            "+ str(down_list[0]), (20, 40),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        #cv2.putText(image, "Roda 4 :            "+ str(down_list[1]), (20, 60),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)

        # menampilkan counting pada GUI
        # atas
        self.ui.image_label_6.setText(str(up_list[0]))
        self.ui.image_label_7.setText(str(up_list[1]))
        # bawah
        self.ui.image_label_4.setText(str(down_list[0]))
        self.ui.image_label_5.setText(str(down_list[1]))

        # create QImage from image
        qImg = QImage(image.data, width, height, step,
QImage.Format_RGB888)
        # show image in img_label
        self.ui.image_label.setPixmAP(QPixmAP.fromImage(qImg))

    # start/stop timer
    def controlTimer(self):
        # create video capture
        self.cap = cv2.VideoCapture(0)
        # start timer
        self.timer.start(1)

    def controlTimer2(self):
        # create video capture
        print(self.file_name)
        filename = ''.join(self.file_name)
        #filename = 'C:\\Users\\Lenovo\\Documents\\Project Jupyter\\aplikasi-
counting\\traffic.mp4'
        self.cap = cv2.VideoCapture(filename)
        # start timer
        self.timer.start(1)

    def stopTimer(self):
        self.timer.stop()
        self.cap.release()

    def exitButtonClick(self):
        self.close()

    def importButtonClick(self):
```

```
        filedir = QFileDialog.getOpenFileName(self,
            "Pilih file", "C:\\Users\\Lenovo\\Documents\\Project Jupyter\\aplikasi-
counting"
            #,"All Files (*)",
            #'Pilih file', os.curdir,
            #'Video File (*.mp4);; Video File (*.avi)',
            ,"*.mp4"
            )
        file_name = filedir[0]
        pengganti = "\\\\"
        self.file_name = file_name.replace("/", pengganti)
        n_index = len(file_name)
        if not filedir[0]: return
        #self.fileLabel.setText('Nama file: ' + filedir[0])
        fileHandle = QFile(filedir[0])
        if not fileHandle.open(QIODevice.ReadOnly) : return
        #stream = QTextStream(fileHandle)
        #self.textEdit.setPlainText(stream.readAll())
        print(n_index)
        print(self.file_name)
        fileHandle.close()

    def count_vehicle(self, box_id, image):

        x, y, w, h, id, index = box_id

        center = find_center(x, y, w, h)
        ix, iy = center

        if (iy > up_line_position) and (iy < middle_line_position):

            if id not in temp_up_list:
                temp_up_list.append(id)

        elif iy < down_line_position and iy > middle_line_position:
            if id not in temp_down_list:
                temp_down_list.append(id)

        elif iy < up_line_position:
             if id in temp_down_list:
                 temp_down_list.remove(id)
                 up_list[index] = up_list[index]+1

        elif iy > down_line_position:
            if id in temp_up_list:
                temp_up_list.remove(id)
```

```
            down_list[index] = down_list[index] + 1

        cv2.circle(image, center, 2, (0, 0, 255), -1)

    def postProcess(self, outputs, image):
        global detected_classNames
        height, width = image.shape[:2]
        boxes = []
        classIds = []
        confidence_scores = []
        detection = []
        for output in outputs:
            for det in output:
                scores = det[5:]
                classId = np.argmax(scores)
                confidence = scores[classId]
                if classId in required_class_index:
                    if confidence > confThreshold:
                        # print(classId)
                        w,h = int(det[2]*width) , int(det[3]*height)
                        x,y = int((det[0]*width)-w/2) , int((det[1]*height)-h/2)
                        boxes.append([x,y,w,h])
                        classIds.append(classId)
                        confidence_scores.append(float(confidence))

        indices = cv2.dnn.NMSBoxes(boxes, confidence_scores, confThreshold,
nmsThreshold)

        if len(indices) > 0:
            for i in indices.flatten():
                x, y, w, h = boxes[i][0], boxes[i][1], boxes[i][2], boxes[i][3]

                color = [int(c) for c in colors[classIds[i]]]
                name = classNames[classIds[i]]
                detected_classNames.append(name)
                cv2.putText(image,f'{name.upper()}',
                        (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)

                cv2.rectangle(image, (x, y), (x + w, y + h), color, 1)
                detection.append([x, y, w, h, required_class_index.index(classIds[i])])

        boxes_ids = tracker.update(detection)
        for box_id in boxes_ids:
            self.count_vehicle(box_id, image)

if __name__ == '__main__':
```

```
app = QApplication(sys.argv)

# create and show mainWindow
mainWindow = MainWindow()
mainWindow.show()

sys.exit(app.exec_())
```

Lampiran 7. Tampilan Aplikasi Klasifikasi Kendaraan