Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IS184853

RANCANG BANGUN APLIKASI PENGHITUNGAN DAN PENGKLASIFIKASIAN KENDARAAN BERMOTOR DENGAN MENGGUNAKAN METODE YOLOV2

APPLICATIONS DEVELOPMENT FOR VEHICLE COUNTING AND CLASSIFICATION USING YOLOV2

YOGA NUGRAHA PRYANDIKA
NRP 052115 4000 0079

Dosen Pembimbing
Renny Pradina K., S.T., M.T., SCJP

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR - IS184853

RANCANG BANGUN APLIKASI PENGHITUNGAN DAN PENGKLASIFIKASIAN KENDARAAN BERMOTOR DENGAN MENGGUNAKAN METODE YOLOV2

YOGA NUGRAHA PRYANDIKA
NRP 052115 4000 0079

Dosen Pembimbing
Renny Pradina K., S.T., M.T., SCJP

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

UNDERGRADUATE THESIS - IS184853

APPLICATIONS DEVELOPMENT FOR VEHICLE COUNTING AND CLASSIFICATION USING YOLOV2

YOGA NUGRAHA PRYANDIKA
NRP 052115 4000 0079

Supervisor
Renny Pradina K., S.T., M.T., SCJP

INFORMATION SYSTEM DEPARTMENT
Information Technology and Communication Faculty
Sepuluh Nopember Institute of Technology
Surabaya 2019

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI PENGHITUNGAN DAN PENGKLASIFIKASIAN KENDARAAN BERMOTOR DENGAN MENGGUNAKAN METODE YOLOV2

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

YOGA NUGRAHA PRYANDIKA
NRP. 0521 15 4000 0079

Surabaya, 17 Juli 2019
KEPALA
DEPARTEMEN SISTEM INFORMASI

Mahendrawati ER, ST, M.Sc, Ph.D
NIP. 197610112006042001

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI PENGHITUNGAN DAN PENGKLASIFIKASIAN KENDARAAN BERMOTOR DENGAN MENGGUNAKAN METODE YOLOV2

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh::

YOGA NUGRAHA PRYANDIKA
NRP. 0521 15 4000 0079

Disetujui Tim Penguji : Tanggal Ujian : 8 Juli 2019
Periode Wisuda : September 2019

Renny Pradina K., S.T., M.T., SCJP **(Pembimbing I)**

Faisal Johan Atletiko, S.Kom., M.T. **(Penguji I)**

Nur Aini Rakhmawati, S.Kom., M.Sc.Eng, Ph.D **(Penguji II)**

Halaman ini sengaja dikosongkan

RANCANG BANGUN APLIKASI PENGHITUNGAN DAN PENGKLASIFIKASIAN KENDARAAN BERMOTOR DENGAN MENGGUNAKAN METODE YOLOV2

Nama Mahasiswa	**: Yoga Nugraha Pryandika**
NRP	**: 05211540000079**
Departemen	**: Sistem Informasi FTIK-ITS**
Pembimbing I	**: Renny Pradina K., S.T., M.T., SCJP**

ABSTRAK

Beberapa tahun terakhir pertumbuhan kendaraan sangat meningkat pesat sehingga menyebabkan lalu lintas pada kota besar menjadi sangat padat. Pemerintah telah melakukan banyak hal untuk mengatasi hal tersebut semisal dengan melakukan pelebaran jalan dan penambahan jalan tol namun hal tersebut masih belum bisa mengurangi jumlah angka kepadatan pada jalanan di Indonesia.

Salah satu upaya untuk mengatasi hal tersebut Pemerintah Surabaya telah membangun suatu sistem yang disebut Surabaya Intellegent Transportation Systems (SITS). Sistem ini dilakukan untuk melakukan rekayasa lalu lintas pada jalan Surabaya. Untuk mendukung kegiatan pemerintah tersebut kami membangun sistem yaitu deteksi dan klasifikasi kendaraan.

Pada penelitian kali ini menggunakan metode YOLOv2 dimana dapat menghasilkan akurasi dan performa yang lebih tinggi dari yang pernah dilakukan sebelumnya. Hasil dari penelitian ini diharapkan dapat terbentuknya suatu aplikasi penghitungan dan klasifikasi kendaraan dengan performa dan akurasi yang tinggi.

Dengan memanfaatkan data sejumlah 21198 yang diantaranya terdapat 8936 mobil, 462 bus, 1681 truk, dan 10119 motor. YOLOv2 dapat menghasilkan mAP 60.63% dengan threshold pada angka 0.5 dan juga YOLOv2 mampu mengeluarkan hasil deteksi dengan frame rate 20 hingga 35 Frame Per Seconds yang dimana hampir menyamai dengan waktu nyata

Kata Kunci: deteksi kendaran, klasifikasi kendaraan, YOLOv2

APPLICATIONS DEVELOPMENT FOR VEHICLE COUNTING AND CLASSIFICATION USING YOLOV2

Name : Yoga Nugraha Pryandika
NRP : 05211540000079
Department : Information System FTIK-ITS
Supervisor : Renny Pradina K., S.T., M.T., SCJP

ABSTRACT

In the last few years the growth of vehicle has greatly increased, causing traffic in the large cities to be very congested. The government has done many thing to overcome this. Such as by widening the road and adding toll roads but it still cannot reduce the number of densities of the roads in Indonesia

One effort to overcome this the government of Surabaya has developing a system. The system is called Surabaya Intellegent Transportation Systems (SITS). That system was made to do traffic engginering for the roads in Surabaya. And then for support the government activities. We build a system for vehicle detection and classification.

In this research we are using a metode called YOLOv2 or we can say You Only Look Once. Where it can produce more accuracy and higher performance than ever done before. The result of this research is expected that the application of vehicle counting and classification can be formed with high performance and accuracy.

By utilizing 21198 data, there are 8936 cars, 462 buses, 1681 trucks and 10119 motorbikes. YOLOv2 can reach mAP up to 60.63% with threshold in 0.5 and YOLOv2 is capable of issuing detection results with a frame rate of 20 to 35 frames per second which is almost equal to real time

Keywords: vehicle detection, vehicle classification, YOLOv2

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Dengan mengucapkan rasa syukur kepada Tuhan Yang Maha Pengasih dan Maha Penyayang atas izin-Nya penulis dapat menyelesaikan buku yang sederhana ini dengan judul Rancang Bangun Aplikasi Penghitungan Dan Pengklasifikasian Kendaraan Bermotor Dengan Menggunakan Metode Yolov2. Dalam penyelesaian Tugas Akhir ini, penulis diiringi oleh pihak-pihak yang selalu memberi dukungan, saran, dan doa sehingga penelitian berlangsung dengan lancar. Secara khusus penulis mengucapkan terima kasih dari lubuk hati terdalam kepada:

1. Tuhan, yang selalu menemani dan membimbing penulis dalam segala aspek kehidupan.
2. Bapak Projek Pryonggo dan Ibu Mufatihah selaku kedua orang tua Yoga Nugraha Pryandika, Nanda Widya Pryandika selaku kakak kandung yang tiada henti memberikan dukungan dan semangat
3. Ibu Renny Pradina K., S.T., M.T., SCJP, Selaku dosen pembimbing dan sebagai narasumber yang senantiasa meluangkan waktu, memerikan ilmu dan petunjuk.
4. Bapak Faisal Johan Atletiko, S.Kom., M.Kom., dan Radityo Prasetianto W., S.Kom., M.Kom., selaku dosen penguji yang telah memberi saran dan kritik untuk perbaikan tugas akhir
5. Mas Willy yang telah menyempatkan waktunya untuk memberikan ilmu dan bimbinganya dalam pengerjaan Tugas akhir ini
6. Seluruh dosen Departemen Sistem Informasi ITS yang telah memberikan ilmu yang bermanfaat kepada penulis.
7. Mas Antok yang telah membantu dalam Pengambilan data di Jalan Raya Waru.
8. Kawan – kawan seperjuangan saya, teman satu bimbingan dengan saya diantaranya Azzam, Andira, Supri, Faris, Magrid, Wenny, Evia, Salim.

9. Safira selaku teman dekat penulis yang telah membantu dan menemani dalam pengerjaan tugas akhir
10. Faisal, Fikri, Faiz, Puji selaku sahabat dari penulis yang telah memberikan semangat canda dan tawa selama perkuliahan
11. Teman teman di BEM FTIF, ISE, ADDI dan LANNISTER yang telah banyak memberi kenangan manis dan pahit selama perkuliahan
12. Berbagai pihak yang tidak bisa disebutkan satu persatu yang telah turut serta menyukseskan penulis dalam menyelesaikan tugas akhir

Penyusunan tugas akhir ini masih jauh dari kata sempurna, untuk itu penulis menerima segala kritik dan saran yang membangun sebagai upaya menjadi lebih baik lagi ke depannya. Semoga buku tugas akhir ini dapat memberikan manfaat untuk pembaca.

Surabaya, 3 Juli 2019

Penulis,

Yoga Nugraha Pryandika

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR KODE

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bab pendahuluan ini akan dijelaskan mengenai bagaimana proses mengidentifikasi masalah dan yang akan dibahas adalah latar belakang masalah, perumusan masalah, batasan masalah, tujuan tugas akhir, serta manfaat kegiatan tugas akhir. Sehingga dari penjelasan dari bab ini, diharapkan dapat memberikan suatu gambararan umum mengenai masalah yang dibahas dan penyelesaianya

1.1 Latar Belakang

Pada saat ini, pertumbuhan pada sektor transportasi mengalami kenaikan yang signifikan terutama pada perkotaan besar seperti pada Surabaya dan Jakarta. Menurut data yang diambil berdasarkan Badan Pusat statisitik perkembangan pengguna kendaraan sejak tahun 2015 mengalami peningkatan yang signifikan [1]. Diketahui setiap tahunya kendaraan di Indonesia bertambah hampir menyentuh 11.5 persen per tahunya [2].

Beberapa hal telah dilakukan oleh pemerintah dengan menambah infrastruktur pada jalan seperti dengan melakukan pelebaran jalan, pelebaran trotoar dan lain lain. Namun hal tersebut tidak dapat mengurangi kemacetan yang ada. Jika hal tersebut tidak segera diatasi dengan tindakan tertentu dapat membuat kepadatan yang berlebih mengingat OJK (Otoritas Jasa Keuangan) telah membuat kebijakan terbaru mengenai pembelian kendaraan bermotor dengan uang muka 0% hal ini tertulis pada Peraturan OJK Nomor 35/POJK.05/2018 Tentang Penyelenggaraan Usaha Perusahaan Pembiayaan [3].

Dari hal tersebut banyak para ahli yang memberikan solusi terhadap kondisi tersebut semisal pada Kota Surabaya yang memiliki sistem sendiri untuk mengatasi kemacetan pada kota tersebut dengan memasang *Surabaya Intelligent Transportation System* (SITS) [4]. Dalam pengembangan sistem tersebut tentunya dibutuhkan teknologi untuk melakukan pendeteksian dan penghitungan terhadap kendaraan tersebut.

Dalam Tugas Akhir ini, dengan menggunakan *You Only Look Once (YOLO)* untuk metode yang akan digunakan, pada penelitian tugas akhir kali ini versi yang digunakan adalah versi yang ke 2 dan untuk *framework* akan menggunakan *darkflow* dimana YOLOv2 ini dapat menghasilkan tingkat presisi yang lebih tinggi dibandingkan dengan metode *R-CNN* dengan menggunakan YOLOv2 ini akan dapat menghasilkan hasil klasifikasi yang lebih cepat dan lebih ringan dari metode yang sudah ada. Dan dapat meghasilkan suatu *streaming video* hasil klasifikasi dengan latensi yang rendah hingga kurang dari 25 *miliseconds* [5].

1.2 Rumusan Masalah

Permasalahan yang akan diselesaikan pada tugas akhir ini adalah

a. Bagaimana merancang penghitungan dan klasifikasi kendaraan dengan menggunakan metode YOLOv2 .
b. Bagaimana merancang dashboard untuk menampilkan jumlah dan jenis kendaraan pada suatu jalan di Surabaya .
c. Apakah dengan data yang didapat, dapat menghasilkan akurasi dan performa sistem yang tinggi

1.3 Batasan Permasalahan

Batasan Permasalahan pada tugas akhir ini adalah sebagai berikut :

a. Untuk objek penelitian adalah jalan meiliki jalur dua arah di area Kota Surabaya
b. Jalan yang dimaksud memiliki jembatan penyebrangan orang (JPO) diatasnya.
c. Pengambilan data akan dilakukan pada waktu siang hari antara pukul 10.00 hingga pukul 12.00 .
d. Klasifikasi kendaraan terbatas pada empat jenis yaitu sepeda motor, mobil, truk, dan bus .

1.4 Tujuan

Tujuan dari tugas akhir ini adalah dapat menjawab permasalahan yang telah dijelaskan sebelumnya maka dari itu tujuan tugas akhir ini adalah sebagai berikut :

a. Merancang arsitektur dan membangun alat untuk penghitungan dan klasifikasi kendaraan bermotor.
b. Rancang bangun dasbor untuk penampilan alat klasifikasi dan penghitungan kendaraan bermotor dengan *User Interface* yang nyaman digunakan.
c. Mengukur Hasil akurasi dan performa yang dihasilkan terhadap sistem yang telah dibuat

1.5 Manfaat

Manfaat dari Tugas Akhir ini adalah diharapkan dapat menghasilkan sistem penghitungan kendaraan yang lebih presisi dan lebih cepat. Selain itu juga dapat melakukan klasifikasi kendaraan, sehingga dapat digunakan oleh dinas tertentu untuk melakukan rekayasa lalu lintas

1.6 Relevansi

Hasil dari tugas akhir ini nantinya diharapkan dapat digunakan sebagai penunjang suatu sistem rekayasa jalan raya contohnya yang terdapat pada kota surabaya yaitu *Surabaya Intelegent Transportation Systems (SITS).*Selain itu sistem tersebut juga dapat digunakan sebagai data penunjang untuk melakukan rekayasa jalan yang lebih mendalam, dengan tujuan mengurangi kemacetan pada kota Surabaya. Tugas akhir ini juga dapat digunakan sebagai pustaka untuk penelitian selanjutnya mengenai penghitungan dan pengklasifikasian kendaraan bermotor. dengan menggunakan metode YOLO ataupun menggunakan metode yang lain yang dapat memberikan hasil yang lebih optimal. Tugas akhir ini juga dapat menjadi pustaka terhadap penggunaan metode YOLO pada penelitian berikutnya. Topik pada tugas akhir ini juga terkait dengan mata kuliah sistem cerdas di Departemen Sistem Informasi. Selain itu topik tugas akhir ini juga sesuai dengan bidang paradigma proses data yang terdapat pada laboratorium Akuisisi Data dan Diseminasi Informasi yaitu bagaimana memanfaatkan teknologi untuk memroses suatu data.

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Bab tinjauan pustaka terdiri dari landasan-landasan yang akan digunakan dalam penelitian tugas akhir ini, mencakup penelitian-penelitian sebelumnya, kajian pustaka, dan metode yang digunakan selama pengerjaan.

2.1 Penelitian Sebelumnya

Pada penelitian ini, digunakan beberapa penelitian terdahulu yang berguna sebagai pedoman dan referensi dalam melaksanakan proses-proses dalam penelitian, seperti yang terdapat pada **Tabel 2.1, Tabel 2.2, dan Tabel 2.3**. Informasi yang disampaikan berisi tentang penelitian sebelumnya, hasil penelitian, dan hubungan penelitian yang akan dilakukan terhadap penelitian sebelumnya dalam rangka tugas akhir ini.

Tabel 2.1 Literatur I

Judul Penelitian	The Deep Learning Development for Real-Time Ball and Goal Detection of Barelang-FC[6].
Penulis; Tahun	Susanto, Eko Rudiawan, Riska Analia, Daniel Sutopo P, Hendawan Soebakti ; 2017
Deskripsi Umum Penelitian	Pada penelitian ini dijelaskan mengenai pemanfaatan metode YOLO pada robot sepak bola tim Barelang-FC. YOLO pada penelitian ini dimanfaatkan untuk mendeteksi bola dan gawang pada lapangan yang nantinya akan diimplementasikan pada robot sepak bola tersebut dengan menggunakan dataset dari total 8000 dataset dimana dibagi 2 kelas sehingga satu kelas memiliki 4000 dataset diperoleh hasil

	akurasi untuk mendeteksi bola dan gawang melebihi 60% dengan kecepatan 20 FPS
Keterkaitan Penelitian	Untuk pembuatan solusi pada tugas akhir ini menggunakan dasar yang sama juga pada penelitian ini yaitu menggunakan algoritma YOLO untuk dasar algoritma yang tidak terdapat pada penelitian YOLOv2

Tabel 2.2 Literatur II

Judul Penelitian	Fast Classification and Detection of Fish Images with YOLOv2[7]
Penulis; Tahun	Mengfan Wang, MingyongLiu, Feihu Zhang, Gang Lei, Jiaojiao Guo, Lu Wang ; 2018
Deskripsi Umum Penelitian	Pada penelitian ini pemanfaatan YOLO dengan menggunakan versi yang lebih baru yaitu versi ke 2 dari YOLO atau bisa dinamakan YOLOv2. Sebagai perbandingan dengan menggunakan dataset VOC 2007+2012 pada YOLOv2 dapat menghasilkan 76.8 mAP dengan kecepatan 67 FPS. Sedangkan pada YOLO versi sebelumnya hanya menghasilkan 63.4 mAP dengan kecepatan 45 FPS. Dengan data training sejumlah 3787, dan data validasi sejumlah 1000 dan data testing sejumlah seribu dapat menghasilkan angka mAP 0.912 dengan kecepatan deteksi 28.3 fps
Keterkaitan Penelitian	Dalam penelitian ini algoritma yang akan dipakai untuk melakuan klasifikasi dan

	penghitungan kendaraan adalah algoritma YOLOv2. Pada penelitian tersebut dijelaskan mengenai performa yang dihasilkan oleh YOLOv2 sehingga mendapatkan pembuktian lebih lanjut mengapa menggunakan YOLOv2

Tabel 2.3 Literatur III

Judul Penelitian	A Concise Review on Vehicle Detection and Classification[8]
Penulis; Tahun	Seda Kul, Suleyman Eken, Ahmet Sayar ; 2017
Deskripsi Umum Penelitian	Dalam penelitian ini membahas mengenai berbagai cara mengenai pemrosesan gambar dan pengaplikasianya. Dalam penelitian ini dijelaskan pendeteksian objek dapat diaplikasikan pada sistem *Intellegent Transportation Systems (ITS)*. Penelitian ini menjelaskan juga mengenai alur yang harus dilakukan untuk melakukan pemrosesan gambar pada kendaraan dan juga paad akhir paper terdapat mengenai metode apa saja yang dipakai dan hasil dari metode tersebut
Keterkaitan Penelitian	Pada penelitian ini terdapat dasar solusi pada penelitian ini yaitu alur apa saja yang harus dilakukan untuk melakukan pemrosesan gambar pada kendaraan.

2.2 Dasar Teori

Berikut merupakan dasar teori yang digunakan

2.2.1 Klasifikasi Kendaraan

Untuk klasifikasi jenis kendaraan bermotor menurut peraturan pemerintah no. 55 Tahun 2012 tentang kendaraan [9]. Pada peraturan pemerintah tersebut jenis kendaaan diklasifikasikan menjadi 4 jenis diantaranya adalah :

a. Sepeda motor adalah kendaraan bermotor meroda dua dengan atau tanpa rumah – rumah dan dengan atau tanpa kereta samping, atau Kendaraan Bermotor beroda tiga tanpa rumah – rumah
b. Mobil Penumpang adalah Kendaraan Bermotor angkutan orang yang memiliki tempat duduk maksimal 8 (delapan) orang, termasuk untuk pengemudi atau yang beratnya tidak lebih dari 3.500 (tiga ribu lima ratus) kilogram.
c. Mobil Bus adalah Kendaraan Bermotor angkutan orang yang memiliki tempat duduk lebih dari 8 (delapan) orang, termasuk untuk pengemudi atau yang beratnya lebih dari 3.500 (tiga ribu lima ratus) kilogram.
d. Mobil Barang adalah Kendaraan Bermotor yang dirancang sebagian atau seluruhnya untuk mengangkut barang.

2.2.2 *Deep Learning*

Deep Learning memungkinkan suatu model komputasi yang terdiri dari banyak *processing layers* untuk mempelajari representasi banyak data dengan level abstraksi yang berbeda. Metode ini telah meningkatkan secara dramatis dalam *speech recognition, visual object recognition, object detection* [10].

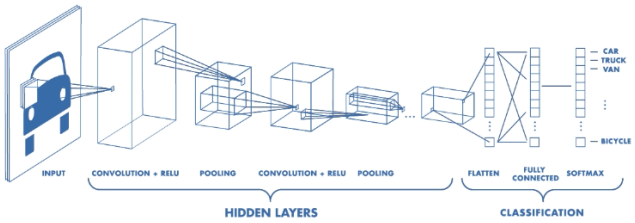
Deep learning juga menemukan suatu struktur rumit dalam data set yang besar dengan menggunakan algoritma *backpropagation* untuk mengindikasi bagaimana suatu mesin harus mengganti internal parameternya yang digunakan untuk mengkomputasi representasi setiap layer dalam representasi later sebelumnya. *Deep convolutional nets* tela membawa terobosan baru dalam pemrosesan gambar, video, pembicaraan, dan audio [11].

2.2.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu metode yang populer untuk pengenalan gambar. CNN pertama kali dikenalkan oleh Kunihiko Fukushima dengan nama *NeoCognition* pada tahun 1980 [12]. Kemudian pada tahun 1998 Yann LeCun mematangkan lagi konsep tersebut pada penelitianya yang bernama LeNet-5 [13].

Konsep dari CNN ini sendiri dapat dilihat pada **Gambar 2.1** terdapat 2 bagian yaitu proses ekstraksi fitur yang terdapat pada *hidden layer* dan proses klasifikasi. Pada proses ekstraksi fitur terdapat proses konvolusi dimana proses ini mengkalikan input dengan suatu *kernel* atau *filter* tertentu. Proses ini terus diulang hingga mengeluarkan hasil yang diinginkan. setelah itu dilakukan proses aktivasi yaitu dengan menggunakan *Leaky ReLu (Rectifier Linear Unit).* Jika proses tersebut selesai maka akan dilakukan proses *pooling* pada hasil konvolusi.

Setelah dilakukan proses ekstraksi fitur berikutnya dilakukan proses *flatten* dimana yang sebelumnya berbentuk 2 dimensi dijadikan 1 dimensi menjadi satu kesatuan baris. Setelah itu di umpan ke layer terakhir yaitu *Fully Connected layers* pada saat disini kan dilakukan proses klasifikasi.

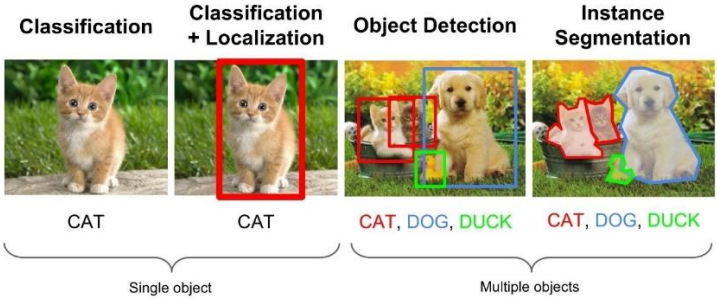


Gambar 2.1 Arsitektur CNN [14]

2.2.4 Deteksi Objek

Deteksi objek adalah suatu proses untuk menemukan suatu instance pada objek dunia nyata seperti wajah, kendaraan, bangunan pada suatu gambar maupun video dapat terlihat pada

Gambar 2.2 bahwa deteksi objek mendeteksi lokasi pada objek dan mengenali suatu objek pada suatu citra [15]. Objek deteksi biasanya algoritmanya terdiri dari ekstraksi fitur dan algoritma pembelajaran untuk mengenali suatu objek. Biasanya deteksi objek digunakan pada penangkap gambar, keamanan, pengawasan, sistem bantuan pengemudi tingkat lanjut [16].



Gambar 2.2 Jenis - jenis Pemrosesan Gambar [17]

Deteksi objek pertama kali dibuat berdasarkan pencocokan template dan part-based model sederhana [18]. Kemudian metode berkembang dengan berdasarkan *statistical classifier* seperti *Neural network, SVM, Naive Bayes*, dan lain lain [19] [20]. Dikarenakan keberhasilan dari berbagai peneliti terhadap metode deteksi objek tersebut dengan menggunakan *stastistical classifier* sehingga dapat memberikan landasan bagi sebagian besar penelitian dalam melakukan training , evaluasi prosedur, dan teknik klasifikasi.

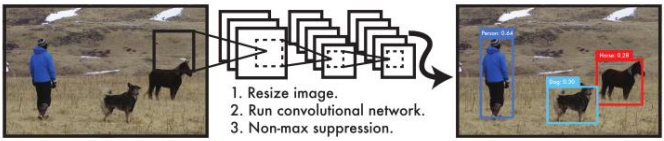
Saat ini sekiranya ada 5 metode untuk melakukan deteksi objek yang pertama adalah *coarse-to-fine and boosted classifier* metode ini bekerja dengan menolak secara efisien suatu rangkaian tes/filter dan *patch* gambar yang tidak sesuai terhadap objek. penelitian yang paling populer dengan menggunakan metode ini adalah *viola and jones* [21]. Berikutnya adalah metode *Dictionary Based*, contoh yang paling terbaik pada metode ini adalah metode *Bag of Word*. Pendekatan ini sederhananya didesain hanya satu objek saja per gambar tapi setelah objek yang telah di deteksi dihilangkan

objek tersisa pada gambar akan terdeteksi[22]. Berikutnya adalah *Deformable Part-Based model* pada pendekatan ini mempertimbangkan objek dan bagian model dan posisi relatif mereka. Pendekatan ini jauh lebih kuat dari pendekata yang lain tetapi pendekatan ini tidak efisien dikarenakan memakan waktu yang cukup lama dan tidak dapat mendeteksi objek dalam skala kecil[18]. Berikutnya adalah pendekatan *deep learning*. Salah satu metode yang sangat sukses menggunakan pendekatan ini adalah *Convolutional Neural Network (CNN)*[23]. Kunci perbedaan dari pendekatan sebelumnya dengan pendekatan ini adalah dalam pendekatan ini representasi fitur bukan dengan didesain oleh user namun sistem mempelajari sendiri terhadap fitur yang ada. Namun kekuranganya diperlukan data pelatihan yang sangat banyak untuk mendapatkan hasil yang maksimal.

2.2.5 Algoritma YOLO

YOLO adalah motode untuk melakukan deteksi objek pada gambar dimana metode ini hanya melihat gambar hanya sekali saja untuk mendeteksi suatu objek (sitasi). Dikarenakan hanya meilihat sekali pada gambar maka jika pada deteksi biasanya hanya menghasilkan 45 fps *(Frame Per Seconds)* pada YOLO akan dihasilkan sebanyak 155 fps dengan konfigurasi *Graphical Processing Unit (GPU)* Titan X, sehingga dapat melakukan streaming video dengan latensi 25 ms[5].

Dapat dilihat pada **Gambar 2.3** proses pada YOLO pada dasarnya sangat sederhana masukkan gambar diawali dengan proses yang dinamakan *preprocessing* dimana proses ini adalah menormalisasikan gambar pada bentuk yang ditentukan berikutnya dilakukan proses konvolusi pada gambar untuk memprediksi *bounding boxes* dan probabilitas kelas pada gambar secara bersamaan. Kemudian menyesuaikan ambang batas untuk mengeluarkan hasil deteksi sesuai dengan *confidence level* pada model.



Gambar 2.3 Sistem Deteksi Pada Yolo [5]

Pada prosesnya input dari gambar dibagi menjadi beberapa *S x S* kisi. Dan setiap sel berfungsi untuk mendeteksi objek ketika titik tegah dari objek pada kotak sel. Setiap sel nya bertujuan untuk memprediksi *bounding box* dan angka confidence dari keakuratan model yang digunakan pada kotak yang mengandung suatu objek. Untuk angka confidence di formulasikan pada (1). Jika tidak ada objek pada sel tersebut. Angka confidence harus bernilai nol. *Confidence* disini mepresentasikan IOU (Intersect Over Union) antara kotak yang diprediksi dan kotak yang dipercaya

$$\Pr(Object) * IOU_{predict}^{truth} \qquad (1)$$

Setiap *bounding box* terdiri dari *(x,y,w,h)* dan *confidence C(Object).* Untuk koordinat (x,y) merepresentasikan mengenai posisi pusat batas kotak deteksi. Dan untuk koordinat (w,h) adalah lebar dan tinggi suatu batas kotak deteksi. Setiap selnya juga memprediksi C. C adalah probabilitas kondisional kelas diepreksikan pada persamaan (2)

$$\Pr(Class_i|Object) \qquad (2)$$

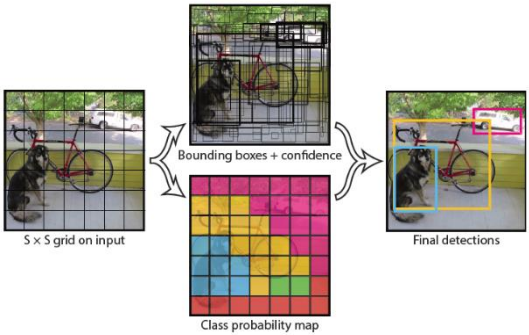
Pada saat melakukan tes probabibilitas kelas kondisional dilakukan pengalian denga kotak prediksi confidence sehingga dapat diutarakan dengan persamaan (3). Dimana dapat memberikan angka *confidence* kelas spesifik pada setiap kotaknya. Angka tersebut mengartikan probabilitas kelas tersebut muncul pada kotak tersebut dan sebagus mana kotak yang telah diprediksi sesuai dengan objek

$$\Pr(Class_i|Object) * \Pr(Object) * IOU_{pred}^{truth} = \Pr(Class_i) * IOU_{pred}^{truth} \quad (3)$$

Deteksi pada sistem ini terlihat pada **Gambar 2.4** membagi gambar menjadi S x S grid dan secara bersamaan memprediksi *bounding boxes*, angka *confidence*, dan probabilitas kelas, dan prediksi tensor dapat dijadikan persamaan (4)

$$S\ x\ S\ x\ (B * 5 + C)\ tensor \quad (4)$$

semisal contoh dengan menggunakan dataset Pascal VOC dengan jumlah pembagian grid menjadi 7 x 7 sehingga S = 7, dan setiap sel terdapat 2 *bounding box* sehingga B = 2. Dikarenakan pada Pascal VOC terdapat 20 kelas sehingga C = 20 sehingga nilai prediksi nya adalah 7 x 7 x 30 tensor.



Gambar 2.4 Model Pada Yolo [5]

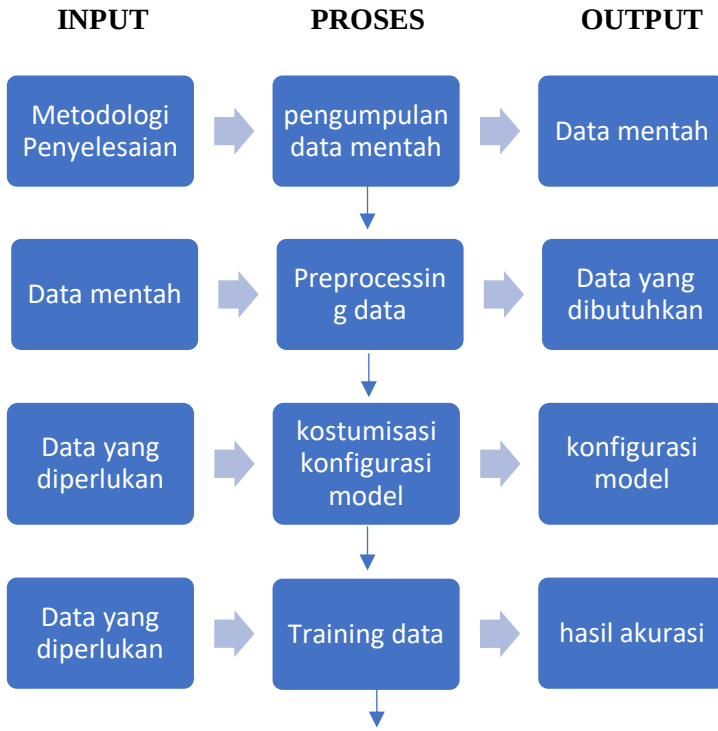
Halaman ini sengaja dikosongkan

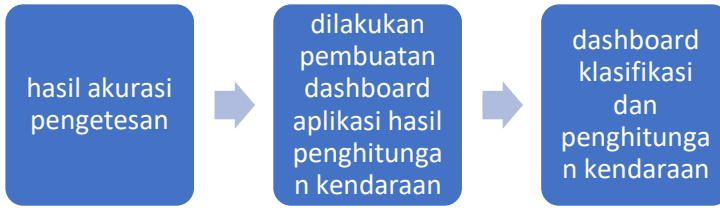
BAB III
METODOLOGI

Pada bab ini dijelaskan mengenai metodologi pengerjaan tugas akhir dan metode apa yang akan digunakan. Dengan demikian pengerjaan tugas akhir dapat dilakukan secara sistematis

3.1 Diagram Metodologi

Pada bagian ini akan dijelaskan metodologi dari penelitian yang akan digunakan sebagai panduan sistematis agar pengerjaan dari tugas akhir menjadi terarah dan berjalan sesuai rencana. Metodologi yang digunakan penulis dapat dilihat pada gambar 3.1





Gambar 3.1 Diagram Metodologi

3.2 Uraian Metodologi

Berikut merupakan penjelasan dari setiap tahapan yang ada pada metodologi yang digunakan, yaitu:

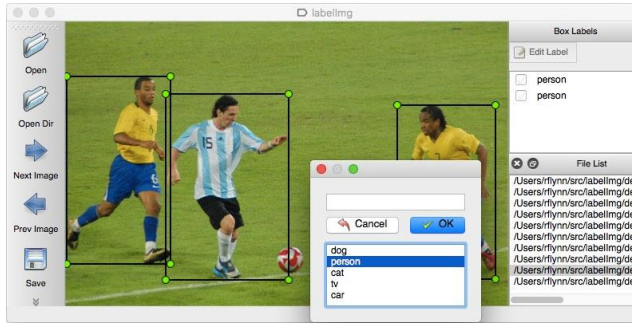
3.2.1 Pengumpulan Data Mentah

Untuk pengumpulan data mentah dilakukan dengan melakukan perekaman video pada jalanan di Surabaya perekaman dilakukan di lokasi pada jalan raya waru tepatnya pada jembatan penyeberangan bungurasih

Untuk perekaman dibagi menjadi 6 posisi yaitu depan kanan, belakang kanan, tengah depan, tengah belakang, kiri depan dan kiri belakang. Durasi perekaman pada masing – masing posisi adalah 10 menit. Diharapkan dari waktu tersebut mendapatkan 5000 gambar tiap kelasnya. setelah data didapatkan lalu dilakukan pemrosesan data yang akan dilakukan pada proses berikutnya

3.2.2 Preprocessing Data

Untuk peprocessing data dilakukan proses anotasi pada suatu objek pada gambar dengan menggunakan *tools Labelimg*. *LabelImg* adalah *tools* yang berguna untuk menganotasi objek pada suatu gambar [24]. Sehingga dapat menghasilkan file anotasi yang berformat PASCAL VOC 2007.



Gambar 3.2 Ilustrasi LabelImg[24]

video yang telah direkam diubah menjadi beberapa gambar dengan menggunakan *FFmpeg*. *FFmpeg* adalah suatu *tools* yang digunakan untuk merubah video menjadi suatu kumpulan gambar [25]. Pada video memproses gambar akan diambil setiap 2 detik dimana diharapkan akan mempunyai gambar yang berbeda setiap 1 jam video akan dibagi menjadi 40 menit pertama diperlukan untuk melakukan training data dan 20 menit untuk pengetesan

3.2.3 Kostumisasi Konfigurasi Model

Setelah dilakukanya pengolahan data berikutnya dilakukan *training* dataset pada model yang telah ada untuk melakukan training digunakan *framework* yang bernama *darkflow* untuk arsitekturnya dapat dilihat pada tabel 3.1

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Tabel 3.1 Arsitektur YOLOv2[26]

Terlihat pada **Tabel 3.1** bahwa YOLOv2 bekerja dengan metode CNN yang telah dimodifikasi. YOLOv2 dari YOLO sebelumnya pada bagian *flatten* dihapuskan dikarenakan untuk mempercepat proses komputasi. Pada YOLOv2 terdapat 19 *layers* konvolusi dan 5 *maxpool layers* [26]. Sehubungan dengan teknik seperti *multi-scale training, batch normalization, dan direct localization prediction.* Berikutnya dilakukan kostumisasi filter pada model yang ada pada *darkflow* dengan persamaan (5)[7].

$$num * (cls + coord) \qquad (5)$$

Untuk cls adalah jumlah kelas yang akan digunakan dimana pada kasus ini *cls = 4.* Berikutnya adalah *coord,* dimana merepresentasikan atribut data pada bounding box yaitu (width, height, x, y) dan C sebagai *confidence level* sehingga *coord = 5.* Dan nilai num adalah 5. Nilai num disini adalah jumlah *bounding box* sehingga nilai total 5 * (4 + 5) = 45.

3.2.4 Training

Selama melakukan training pada model yang telah dibuat. *Loss function* pada model dapat dibagi menjadi 3 yaitu *classification loss, localization loss, confidence loss*[5]. Untuk *Classifiction loss* terlihat pada persamaan (6) jika objek terdeteksi, *classification loss* pada setiap sel nya pangkat error dari probabilitas kondisional setiap kelasnya

$$\sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \qquad (6)$$

Pada persamaan (6) terdapat 1_i^{obj} adalah bernilai 1 jika objek tersebut ada pada sel i, jika tidak ada bernilai 0. Berikutnya merupakan *localization loss*, pada *localization loss* mengukur error yang memprediksi lokasi *boundary box* ukuranya. Yang diukur hanya kotak yang bertanggung jawab untuk mendeteksi objek

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \qquad (7)$$

Pada persamaan (7) bertugas untuk mengkomputasi *loss* mengenai posisi *bounding box* (x,y) yang telah diprediksi. Persamaan ini menjumlahkan setiap semua yang memprediksi *bounding box* setiap sel nya dan untuk (x,y) adalah posisi yang diprediksi dan $(\hat{x}, \hat{y})$ adalah psisi aktual dari *training* data

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \qquad (8)$$

Untuk persamaan (8) adalah mengenai lebar dan tinggi kotak yang telah diprediksi. Penggunaan akar pada persamaan tersebut dikarenakan persamaan ini harus merefleksikan deviasi pada kotak besar kurang penting daripada di kotak kecil. Untuk

mengatasi hal tersebut digunakan akar kuadrat lebar dan tinggi pada *bounding box* dari pada lebar dan tinggi secara langsung

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{obj}\left(C_i - \hat{C}_i\right)^2 + \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{noobj}\left(C_i - \hat{C}_i\right)^2 \quad (9)$$

Untuk persamaan (9) adalah penghitungan mengenai loss yang menghitung angka *confidence* pada setiap prediktor *bounding box. C* adalah nilai *confidence* dan $\hat{C}$ adalah IOU dari *bounding box* yang diprediksi dengan kepercayaan latar. 1obj adalah bernilai 1 ketika ada objek pada sel tersebut bernilai 0 sebaliknya dan untuk 1noobj adalah kebailkanya

3.2.5 Pembuatan Dasbor

Untuk pembuatan dasbor akan menggunakan library dari *matplotlib*. Dimana setiap objek akan dihitung setiap detiknya kemudian dimasukkan pada suatu *list* pada *python* kemudian. Setiap kelas akan memiliki list nya masing masing. Setelah itu list yang terbuat tersebut dimasukkan ke dalam *matplotlib* untuk ditampilkan perkembangan grafik jumlah kendaraan

3.2.6 Penyusunan Laporan Tugas AKhir

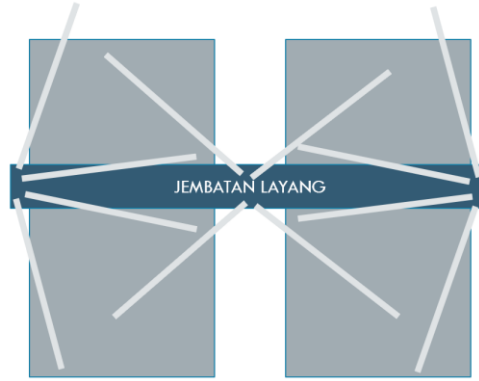
Proses terakhir dalam pengerjaan penelitian tugas akhir ini adalah dengan membuat laporan tugas akhir sebagai bentuk dokumentasi atas terlaksananya tugas akhir ini. Laporan tugas akhir dibuat dengan format yang telah ditentukan. Proses pengerjaan laporan dilakukan sejak awal hingga berakhirnya proses pengerjaan penelitian tugas akhir

BAB IV
PERANCANGAN

Pada bab ini dilakukan pembuatan model klasifikasi yang dibutuhkan. Perancangan yang akan digunakan dalam mengklasifikasikan kendaraan dengan menggunakan arsitektur YOLOv2. Sebelum memulai klasifikasi, harus ditentukan dulu kelas kelas yang dibutuhkan untuk menyelesaikan permasalahan pada tugas akhir

1.1 Pengumpulan Data

Pengumpulan data dilakukan dengan melakukan perekaman di Jalan Raya Waru tepatnya pada jembatan penyebrangan pada Terminal Bungurasih.



Gambar 4.1 Model Pengambilan Gambar

Perekaman dilakukan pada pukul 10.00 hingga 12.00 dengan merekam 6 sisi Jalan Raya Waru dimana sisi sisi tersebut terlihat pada **Gambar 4.1** dimana setiap sisinya dilakukan perekaman selama 20 menit sehingga waktu total adalah 120 menit. Spesifikasi kamera yang akan digunakan untuk perekaman dapat dilihat pada tabel 4.1

Tabel 4.1 Konfigurasi kamera yang digunakan

Nama Kamera	Sony a7II
Resolusi Video	1440 x 1080
Kecepatan frame	25 fps
Lensa yang digunakan	50mm

Gambar 4.2 Gambar Jalan Bungurasih

Contoh hasil perekaman dari keenam sisi seperti yang dijelaskan sebelumnya terlihat pada gambar 4.2. hasil dari perekaman tersebut nantinya akan dibagi antara *training* dan juga *testing*. Total perekaman adalah 60 menit dan data yang akan digunakan untuk training adalah 40 menit dan data untuk testing adalah 20 menit

4.1.1 Ekstraksi gambar dari video

Dari video tersebut diambil gambar setiap dua detiknya dengan menggunakan *tools* yang bernama *FFmpeg*. Perintah yang digunakan untuk mendapatkan gambar dari video input dengan *FFmpeg* dapat dilihat pada kode

```
ffmpeg -i <input_video> -vf fps=1/2
<namaoutput>.jpg -hide_banner
```

dengan membaginya menjadi 1 gambar per 2 detik dari 20 menit video didapatkan gambar dengan jumlah 603 gambar per video

nya sehingga total gambar yang didapatkan adalah 3618 gambar.

4.2 Praproses Data

Pada proses ini dilakukan pemrosesan data yang nantinya akan digunakan untuk pelatihan pada arsitektur yang telah dibuat. Praproses disini adalah proses memberikan label pada suatu objek pada gambar. Hal ini dilakukan untuk memberikan suatu data pembelajaran pada arsitektur yang dibuat sehingga dapat menghasilkan nilai bobot yang diinginkan. Untuk membuat label diperlukan *tools* yang dinamakan *LabelImg*

Gambar 4.3 Contoh gambar yang telah teranotasi menggunakan *LabelImg*

Untuk melakukan pelabelan pertama adalah membuat kelas apa saja yang akan digunakan dalam bentuk *file .txt*. kemudian untuk memulai pelabelan membuka *labelImg* pada *command line*, beserta lokasi gambar yang akan diberikan anotasi

Ketika *tools* tersebut telah terbuka dan gambar telah terbuka maka proses pelabelan dapat dilakukan. Pelabelan dilakukan dengan membuat suatu *box* dengan menekan tombol *create new box.* Setelah itu box dapat digambarkan pada lokasi objek

Berikut contoh pelabelan gambar dengan *tools labelImg*. terdapat bagaimana setiap gambar akan dilakukan anotasi untuk

membedakan setiap objek pada gambar. Untuk klasifikasinya terdapat pada tabel 4.1

Tabel 4.2 Jenis Klasifikasi

Gambar	Nama
	Mobil
	Motor
	truk
	Bus

Setalah gambar diberikan label akan memberikan keluaran berupa *file* anotasi yang mereferensikan lokasi piksel, ukuran objek pada gambar dan klasifikasi terhadap objek pada gambar

Nantinya file anotasi tersebut berisi nama file gambar, lokasi piksel object dan ukuran object, lokasi gambar, serta kelas objek dapat digunakan untuk keperluan pelatihan

4.3 Arsitektur YOLOv2

Untuk tugas akhir ini memakai *library darkflow* dimana *darkflow* adalah alternatif bahasa pemrograman *python library darknet* yang dimana *darknet* adalah *library* dari pencipta YOLO

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Gambar 4.4 arsitektur YOLOv2 pada darknet-19

Pada arstektur darknet-19 ini terdapat 19 layer konvolusi dimana setiap layernya terdapat filter, aktivasi, *batch normalization.*

4.3.1 Perancangan kostumisasi Arsitektur YOLOv2

kostumisasi dari arsitektur YOLOv2 yang dilakukan adalah mengubah filter konvolusi terakhir dari YOLOv2, Untuk pengubahanya menggunakan rumus pada persamaan (5)

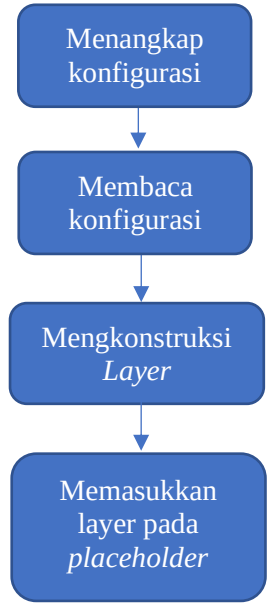
$$num * (cls + coord) \qquad (5)$$

Dimana pada rumus tersebut *num* adalah bernilai 5 dimana nilai ini telah ditentukan oleh pembuat arsitektur tersebut. Kemudian *cls* adalah jumlah kelas yang akan dipakai pada model yang akan digunakan. Dimana berjumlah 4 kelas diantaranya adalah bus, truk, motor, mobil. Untuk *coord* adalah bernilai 5 dimana merupakan x,y,*width*,*height* dan *confidence level* pada setiap *bounding box*-nya. Sehingga total filter yang dibutuhkan adalah 5 * (4 + 5) bernilai 45.

Selain pengubahan filter juga terdapat pengubahan pada masukan gambar. Gambar masukan akan dikecilkan ukuranya untuk standarnya adalah memakai ukuran 608x608 dari ukuran tersebut diubah menjadi 416x416 hal ini dilakukan agar pelaksanaan pelatihan pada sistem dapat dilakukan pada sistem yang dimiliki

4.4 Perancangan pemuat arsitektur

Gambar 4.5 adalah bagaimana arsitektur dimuat

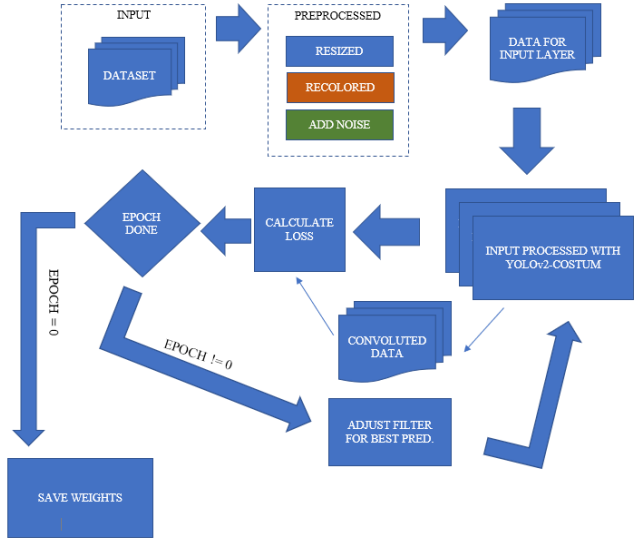


Gambar 4.5 Diagram perancangan konfigurasi

Untuk pemuat layer pertama adalah menangkap konfigurasi mana yang akan dipakai nantinya. Lalu sistem membaca konfigurasi yang akan dimasukkan tiap barisnya. Setelah dibaca kemudian konfigurasi tersebut dibentuk dengan membuat suatu variable yang nantinya diisi suatu parameter yang berhubungan terhadap arisitektur tersebut. Kemudian *layer* yang terbentuk tersebut dimasukkan pada *placeholder tensorflow* untuk digunakan sebagai *tesing* maupun *training*

4.5 Perancangan training

proses pada training berawal dari pengambilan dataset dan anotasi. File *dataset* yang didapatkan akan dimasukkan pada *layer* yang telah terbuat sebelumnya. Kemudian nanti akan terbentuk suatu *loss* dari hasil *training* tersebut untuk penggambaranya terdapt pada gambar 4.6



Gambar 4.6 Rancangan Training

Pada tahap pertama masukan berupa dataset dan anotasi dilakukan pengubahan ukuran terlebih dahulu. Dataset tersebut diubah menjadi lebar dan tinggi sesuai dengan input pada layer konfigurasi yang sebelumnya telah dibuat selain itu juga

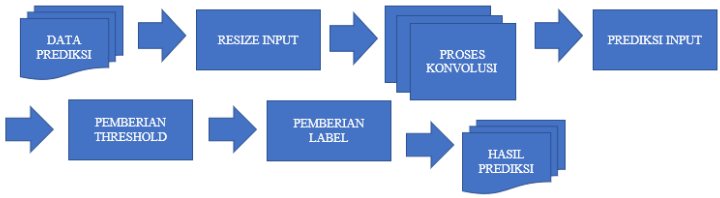
terdapat pengubahan warna dan penambahan *noise* untuk keperluan penambahan data pada proses pelatihan.

Data yang telah diolah tersebut, kemudian akan dilakukan pemrosesan pada layer konvolusi yang telah dibuat sebelumnya. Sehingga menghasilkan data yang telah terkonvolusi. Data yang sudah terkonvolusi berikutnya dilakukan prediksi. hasil prediksi tersebut akan dibandingkan dengan hasil anotasi dataset. Hasil pembandingan tersebut akan menghasilkan loss. Hasil loss tersebut yang nantinya akan menjadi patokan, untuk memberi nilai filter pada iterasi berikutnya

Ketika iterasi selesai nilai hasil akurasi akan disimpan dalam bentuk ckpt, pb atau dalam bentuk *weights*.

4.6 Perancangan prediksi

Prediksi dilakukan dengan memasukkan gambar yang akan diprediksi. Gambar tersebut sebelum dilakukanya prediksi, terlebih dahulu dilakukan preprocessing dengan mengubah ukuran gambar sesuai dengan masukan layer. Penggambaranya terdapat pada gambar 4.7



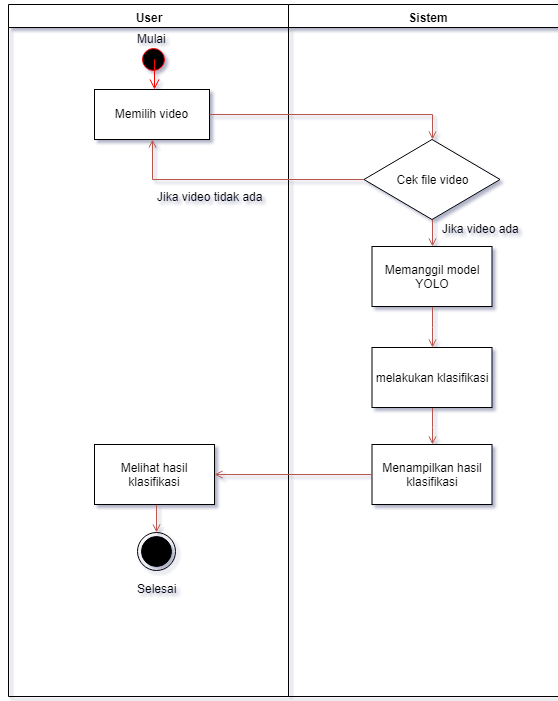
Gambar 4.7 Proses Prediksi

Setelah diubah ukuran gambar data input dimasukkan pada proses konvolusi pada YOLOv2. Hasil konvolusi tersebut berguna untuk melakukan prediksi. Prediksi dilakukan dengan menggunakan menggunakan *K-means.* Dimana pada *darkflow* dilanjutkan ke *darknet* dengan menggunakan *cython.* Untuk penentuan *anchor* pada *bounding boxes.* Nantinya keluaran prediksi adalah lokasi objek yang diprediksi, indeks klasifikasi, ukuran objek.

Setelah dilakukan prediksi berikutnya adalah *postprocessing*. Dengan memberikan label dan pemberian *threshold*. *threshold* adalah pemberian batasan pada prediksi untuk muncul atau tidaknya prediksi tersebut. Dengan mengacu pada *confidence level* suatu prediksi, semisal suatu prediksi memiliki *confidence level* diatas *threshold* yang bernilai 0.6 maka prediksi tersebut dapat muncul jika nilai tersebut dibawah nilai *threshold* yang telah ditentukan. Maka prediksi tidak dapat muncul

4.7 Perancangan penampil klasifikasi

Untuk menampilkan hasil klasifikasi tersebut tentunya diperlukan sebuah sistem untuk mengambil masukkan dan mengeluarkan output hasil klasifikasi tersebut. Dan juga agar pengguna lain dapat melihat hasil klasifikasi tersebut. *Activity diagram* dari perangkat lunak ini terdapat pada gambar 4.8

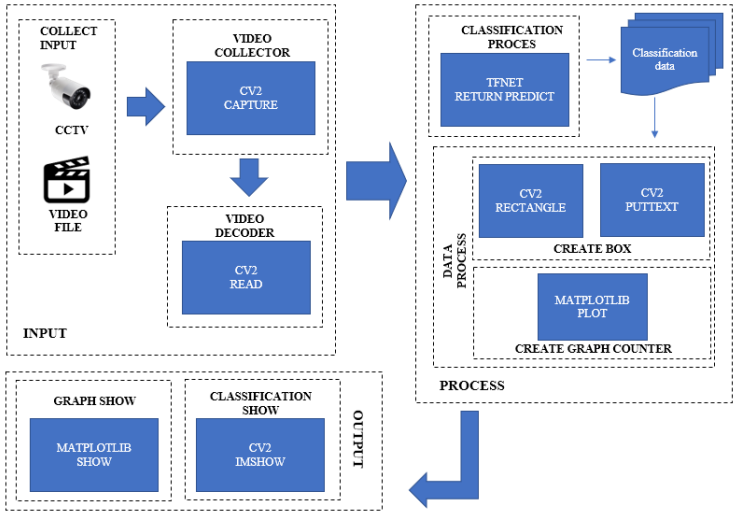


Gambar 4.8 *Activity* Diagram sistem klasifikasi

Pada sistem pengguna dapat memilih video untuk diklasifikasikan. Kemudian sistem mengambil masukkan dari pengguna. Berikutnya sistem memanggil *darkflow* dan menginputkan video dari user untuk melakukan deteksi dan klasifikasi kendaraan pada video tersebut.

4.8 Arsitektur sistem penampil klasifikasi

Untuk arsitektur dari activity diagram tersebutdibuatlah arsitektur untuk sistem diantaranya pada Gambar 4.9



Gambar 4.9 Arsitektur Sistem penampil klasifikasi

Pada sistem untuk masukkan, keluaran, dan proses uraianya adalah sebagai berikut

a. Untuk *input* ada pengambilan gambar yang dilakukan menggunakan *library CV2 capture.* Lalu setelah video didapatkan video di*decode* dengan menggunakan *CV2 read*
b. Untuk prosesnya terdapat dua proses yaitu terdapat proses klasifikasi dan menampilkan hasil klasifikasi dan hasil penghitungan dengan menggunakan *library matplotlib*. Klasifikasi dilakukan dengan memanggil

darkflow. Dengan TFnet dengan menggunakan *method return_predict*

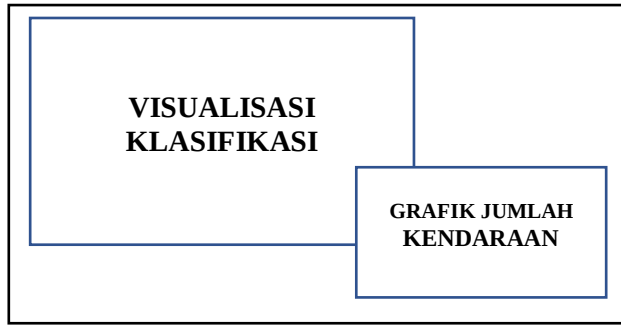
c. Lalu *output* dengan menggunakan cv2 dan library dari *matplotlib*

4.9 Antarmuka Sistem

Untuk antarmuka sistem terdapat dua komponen diantaranya adala visualisasi klasifikasi dan grafik jumlah kendaraan dan fingsi kedua komponen tersebut adalah sebagai berikut

a. Visualisasi Klasifikasi, fungsi dari visualisasi klasifikasi adalah untuk menampilkan bagaimana proses klasifikasi bekerja dan pada posisi mana objek berada

b. Grafik jumlah kendaraan, fungsi dari grafik tersebut adalah untuk menampilkan jumlah kendaraan dan pertumbuhan dari jumlah kendaraan



Gambar 4.10 Antarmuka penelitian

Untuk tampilan dapat terlihat pada **gambar 4.10** yang merupakan rancangan visualisasi dari penelitian pada tugas akhir .

Halaman ini sengaja dikosongkan

BAB V
IMPLEMENTASI

Pada bagian ini akan dijelaskan mengenai proses implementasi dan bagaimana dalam memproses data, bagaimana memanfaatkan model YOLOv2, berdasarkan pada proses perancangan sebelumnya

5.1 Lingkungan Implementasi

Pada penelitian pada tugas akhir kali ini menggunakan komputer dengan spesifikasi sebagai berikut

Tabel 5.1 Spesifikasi hardware

Prosessor	Intel i7-4790
Memori	8 GB DDR3
GPU	RTX 2060 6GB
OS	Linux Ubuntu 16.04
Arsitektur	64-Bit

Selain itu terdapat package, tools yang menunjang pengerjaan dari tugas akhir ini diantaranya adalah sebagai berikut

Tabel 5.2 Package yang digunakan

Bahasa Pemrograman	Python 3.6.3
IDE	Pycharm 2019.1
Package	• Opencv-python • Cython • Scikit-image • Matplotlib • Tensorflow 1.12 • Tensorflow-gpu 1.12 • Tensorboard 1.12.2 • Numpy

5.2 Pelabelan gambar

Pelabelan gambar diperuntukan data yang dibutuhkan untuk melakukan training. Setiap objek pada gambar yang didapatkan diberikan label. Pelabelan dilakukan dengan menggunakan tools yaitu LabelImg. Dari melakukan pelabelan akan mengelarkan file berbentuk xml dimana file tersebut adalah anotasi dari gambar yang telah diberi label tersebut

```
<annotation>
    <folder>dataset</folder>
    <filename>video1_data0001.jpg</filename>
    <path>/home/yoga/PycharmProjects/vehicle-
detection/dataset/video1_data0001.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>1440</width>
        <height>1080</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>mobil</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>675</xmin>
            <ymin>461</ymin>
            <xmax>804</xmax>
            <ymax>593</ymax>
        </bndbox>
    </object>
```

Kode 5.1 Format dataset PASCAL VOC 2007

File xml berisi anotasi yang mereferensikan gambar yang teranotasikan. Pada **Kode 5.1** merupakan bagaimana isi dari file anotasi tersebut. Untuk anotasi menggunakan format dari pascal VOC 2007. Isi dari xml tersebut adalah mereferensikan lokasi gambar, ukuran gambar, klasifikasi objek, dan lokasi objek.

Data yang sudah terlabeli tersebut berikutnya akan digunakan sebagai training yang nantinya akan menghasilkan bobot. Bobot tersebut digunakan untuk membentuk suatu model

5.3 Arsitektur YOLOv2

YOLO v2 terdiri dari 19 layer konvolusi 5 layer maxpool. Setiap layer konvolusinya terdiri dari fungsi filterisasi, batch normalization, stride, pad dan fungsi aktivasi. Sebelum menuju sistem terlebih dahulu membuat suatu file konfigurasi untuk nanti dimasukan pada sistem yang akan menjalankan prediksi dan melakukan training. File konfigurasi terdapat pada Kode 5.2

```
1. [convolutional]
2. batch_normalize=1
3. filters=32
4. size=3
5. stride=1
6. pad=1
7. activation=leaky
```

Kode 5.2 Layer Konvolusi

Untuk urutan pertama adalah melakukan *Batch Normaization.* Hal ini dilakukan untk meningkatkan stabilitas dari neural network dengan mempertahankan skala yang seragam untuk semua fitur yang mencegah fitur yang memiliki nilai yang sangat besar untuk mempengaruhi nilai bobot.

Setelah itu dilakukanya proses filterisasi gambar. Gambar input dilakukan filter dengan ukuran senilai pada *size* dan berjumlah pada *filter* yang terdapat pada konfigurasi tersebut.

Kemudian setelah itu dibuat konfigurasi untuk melakukan *maxpool* yang terdapat pada kode

```
1. [maxpool]
2. size=2
3. stride=2
```

Kode 5.3 Kode maxpool

Berikutnya pada konfigurasi terakhir adalah mendefinisikan *softmax* dari arsitektur versi YOLO dan jumlah kelas yang akan digunakan

```
[region]
anchors =  0.57273, 0.677385, 1.87446, 2.06253, 3.33843, 5.47434, 7.88282, 3.52778, 9.77052, 9.16828
bias_match=1
classes=80
coords=4
num=5
softmax=1
jitter=.3
rescore=1

object_scale=5
noobject_scale=1
class_scale=1
coord_scale=1

absolute=1
thresh = .1
random=1
```

Kode 5.4 Konfigurasi terakhir

5.3.1 Kostumisasi arsitektur YOLOv2

kostumisasi arsitektur YOLOv2 digunakan untuk menyesuaikan filter pada YOLOv2 dterhadap jumlah kelas yang dimiliki pada dataset. Untuk hal pertama yang dilakukan adalah menghitung filter yang akan dimasukkan dengan mengggunakan rumus yang telah dibahas pada bab perancangan dan dari perhitungan tersebut dapat dihasilkan dengan nilai 45 dan untuk pengubahanya terdapat pada layer terakhir

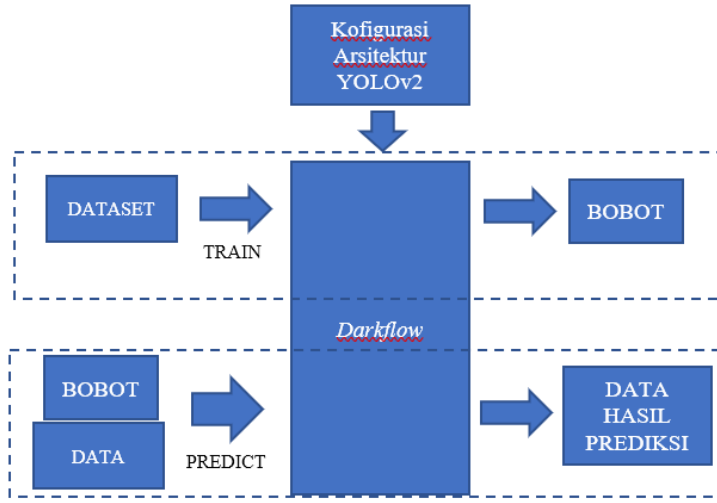
```
[convolutional]
size=1
stride=1
pad=1
filters=45
activation=linear
```

```
[region]
anchors =  0.57273, 0.677385, 1.87446, 2.06253, 3
.33843, 5.47434, 7.88282, 3.52778, 9.77052, 9.168
28
classes=4
coords=4
num=5
```

Kode 5.5 Potongan kode konfigurasi

5.4 Pembuatan Pemuat konfigurasi

Sistem ini bertujuan untuk menjalankan konfigurasi yang sebelumnya dibuat. Sistem nantinya akan dapat dijalankan melalui terminal ataupun dari aplikasi python lainya. dan pembuatan sistem terbagi menjadi dua satu untuk training dan yang satu dibuat untuk memprediksi. Untuk penggambaran singkatnya terdapat pada gambar 5.1



Gambar 5.1 Bagan pemuat arsitektur

Konfigurasi arsitektur akan diekstrak oleh darkflow untuk pembentukan *layer* sesuai dengan konfigurasi yang dimiliki. Untuk *training* diperlukan suatu dataset yang nantinya akan

diproses konvolusi pada *darkflow* yang telah berisikan konfigurasi YOLOv2. Yang nantinya dapat membentuk bobot.

Untuk melakukan prediksi data dan bobot akan dimasukkan kepada *darkflow* yang berisikan konfigurasi YOLOv2. Untuk dilakukan proses prediksi . dan hasil keluaranya berupa hasil prediksi berupa lokasi ukuran dan nilai *confidence level* untuk memprediksi objek

5.4.1 Konstruktor Layer

Untuk mengonstruksi layer dilakukan pada *build.py*. *build.py* bertugas untuk menangkap perintah dari pengguna semisal pengguna menginginkan suatu argumen pada *darkflow*

```
1.   def __init__(self, FLAGS, darknet = None):
2.  self.ntrain = 0
3.  if isinstance(FLAGS, dict):
4.      from ..defaults import argHandler
5.      newFLAGS = argHandler()
6.      newFLAGS.setDefaults()
7.      newFLAGS.update(FLAGS)
8.      FLAGS = newFLAGS
```

Kode 5.6 Mengupdate argumen

Pada Kode 5.6 berfungsi untuk mengupdate apakah ada suatu argumen dari pengguna yang terdapat pada baris 7. jika ada, nilai default pada argumen tersebut di update menggunakan argumen pengguna

```
1.  if darknet is None:
2.      darknet = Darknet(FLAGS)
3.      self.ntrain = len(darknet.layers)
```

Kode 5.7 Potongan Kode untuk melanjutkan pada kelas darknet

Dikarenakan pada kode sebelumnya *darknet* belum ada, maka dengan kode pada Kode 5.6 akan melanjutkanya ke kelas *Darknet* kelas *darknet* yang terdapat pada baris 2 berfungsi untuk pembuatan *layer*.

```
1.  def get_weight_src(self, FLAGS):
```

```
2.      self.src_bin = FLAGS.model + self._EXT
3.      self.src_bin = FLAGS.binary + self.src_bin
4.      self.src_bin = os.path.abspath(self.src_bin)

5.      exist = os.path.isfile(self.src_bin)
6.      if FLAGS.load == str(): FLAGS.load = int()
7.      if type(FLAGS.load) is int:
8.        self.src_cfg = FLAGS.model
9.        if FLAGS.load: self.src_bin = None
10.       elif not exist: self.src_bin = None
11.     else:
12.       assert os.path.isfile(FLAGS.load), \
13.       '{} not found'.format(FLAGS.load)
14.       self.src_bin = FLAGS.load
15.       name = loader.model_name(FLAGS.load)
16.       cfg_path = os.path.join(FLAGS.config, name
    + '.cfg')
17.         if not os.path.isfile(cfg_path):
18.             warnings.warn(
19.                 '{} not found, use {} instead'.fo
    rmat(
20.                 cfg_path, FLAGS.model))
21.             cfg_path = FLAGS.model
22.         self.src_cfg = cfg_path
23.         FLAGS.load = int()
```

Kode 5.8 potongan kode mendapatkan konfigurasi dan bobot

Pertama kali program akan mengambil model dan *binary* dari argumen yang dimasukkan oleh pengguna yang terdapat pada baris 2 dan 3. kemudian mengambil lokasi *file* bobot. Bobot kemudian di cek apakah berbentuk *integer* atau *string* pada baris 6 semisal jika menggunakan integer maka lokasi file yang sebelumnya di definisikan akan dirubah menjadi tidak ada dan konfigurasi arsitektur yang dimasukkan oleh user akan dijadikan sebagai konfigurasi pada *darknet*

```
1.  def parse_cfg(self, model, FLAGS):
2.          args = [model, FLAGS.binary]
3.          cfg_layers = cfg_yielder(*args)
4.          meta = dict(); layers = list()
5.          for i, info in enumerate(cfg_layers):
6.              if i == 0: meta = info; continue
```

```
7.              else: new = create_darkop(*info)
8.              layers.append(new)
9.          return meta, layers
```

Kode 5.9 Potongan kode untuk membuat list layer

Pada kode 5.9 dari konfigurasi aritektur yang didapatkan konfigurasi akan dibentuk pada potongan kode tersebut dengan memanggil dari potongan kode 5.10 pada baris 7 setelah itu nilai nilai yang didapatkan, nilai nya akan dikembalikan pada variabel meta dan layers

```
1.  def _parse(l, i = 1):
2.      return l.split('=')[i].strip()
3.  with open(model, 'rb') as f:
4.      lines = f.readlines()
5.  lines = [line.decode() for line in lines]
6.  meta = dict(); layers = list()
7.  h, w, c = [int()] * 3; layer = dict()
8.  for line in lines:
9.      line = line.strip()
10.     line = line.split('#')[0]
11.     if '[' in line:
12.        if layer != dict():
13.        if layer['type'] == '[net]':
14.              h = layer['height']
15.              w = layer['width']
16.              c = layer['channels']
17.              meta['net'] = layer
18.     else:
19.        try:
20.          i = float(_parse(line))
21.          if i == int(i): i = int(i)
22.             layer[line.split('=')[0].strip()] = i
23.        except:
24.              try:
25.              key = _parse(line, 0)
26.                 val = _parse(line, 1)
27.                 layer[key] = val
28.              except:
29.                 'banana ninja yadayada'
```

Kode 5.10 Pendefinisian jenis layer

Program akan membaca file konfigurasi tersebut tiap barisnya pada baris 4. Hasil pembacaan tiap barisnya dimasukkan pada suatu *list* dimana list tersebut bernama *layer*. Kemudian untuk *layer* yang bersifat parameter akan dimasukkan pada meta dan *layer* yang bersifat dari *layaer convolutional* dan *maxpool* akan dimasukkan pada *layers*.

```
1.  elif d['type'] == '[convolutional]':
2.      n = d.get('filters', 1)
3.      size = d.get('size', 1)
4.      stride = d.get('stride', 1)
5.      pad = d.get('pad', 0)
6.      padding = d.get('padding', 0)
7.      if pad: padding = size // 2
8.      activation = d.get('activation', 'logistic')
9.      batch_norm = d.get('batch_normalize', 0) or co
    nv
10.     yield ['convolutional', i, size, c, n,
11.     stride, padding, batch_norm, activation]
12.     if activation != 'linear': yield [activation,
    i]
13.     w_ = (w + 2 * padding - size) // stride + 1
14.     h_ = (h + 2 * padding - size) // stride + 1
15.     w, h, c = w_, h_, n
16.     l = w * h * c
```

Kode 5.11 Pembuatan layer

Pada kode 5.11 layer pun dibuat, pertama program melihat pada *dictionary layers* jika ada yang bernama layer bertipe *convolutional* pada baris 1 maka nilai dari layer tersebut dimasukkan kepada variabel yang dideklarasikan pada kode tersebut. Ketika sudah mencapai *yield* kode meneruskan fungsi ke Kode 5.9 dimana pada kode tersebut akan meneruskanya pada kelas *Layer*

```
1.  from ..utils import loader
2.  import numpy as np
3.
4.  class Layer(object):
5.      def __init__(self, *args):
6.          self._signature = list(args)
```

```
7.          self.type = list(args)[0]
8.          self.number = list(args)[1]
9.          self.w = dict() # weights
10.         self.h = dict() # placeholders
11.         self.wshape = dict() # weight shape
12.         self.wsize = dict() # weight size
13.         self.setup(*args[2:]) # set attr up
14.         self.present()
15.         for var in self.wshape:
16.             shp = self.wshape[var]
17.             size = np.prod(shp)
18.             self.wsize[var] = size
```

Kode 5.12 Kode untuk memasukkan parameter *layer*

Pada kelas ini berfungsi untuk memasukkan nilai yang terdapat pada *layer* ke parameter – parameter dari baris 6 hingga 13 dan juga memproduksi nilai ukuran filter konvolusi dan mendefinisikan termasuk tipe apakah *layer* tersebut kemudian kode dilanjutkan pada kelas *convolution*

```
1.  class convolutional_layer(Layer):
2.      def setup(self, ksize, c, n, stride,
3.                pad, batch_norm, activation):
4.          self.batch_norm = bool(batch_norm)
5.          self.activation = activation
6.          self.stride = stride
7.          self.ksize = ksize
8.          self.pad = pad
9.          self.dnshape = [n, c, ksize, ksize] # dar
    knet shape
10.         self.wshape = dict({
11.             'biases': [n],
12.             'kernel': [ksize, ksize, c, n]
13.         })
14.         if self.batch_norm:
15.             self.wshape.update({
16.                 'moving_variance'  : [n],
17.                 'moving_mean': [n],
18.                 'gamma' : [n]
19.             })
20.             self.h['is_training'] = {
21.                 'feed': True,
22.                 'dfault': False,
```

```
                'shape': ()
            }
```

Kode 5.13 Potongan Kode untuk memasukkan nilai ke layer konvolusi

Dengan kelas ini berfungsi untuk menjabarkan lagi parameter yang telah didapatkan dari kelas sebelumnya dimana hal ini parameter tersebut akan berguna untuk pembentukan layer pada *tensorflow*, Serta untuk melakukan suatu prediksi. Untuk keluaran akhir dari pembentukan layer adalah terdapat pada gambar

Tabel 5.3 Hasil Keluaran konstruktor layer

Variabel	Keluaran
Layer	Convolutional,0,3,3,32,1,1,1,'leaky'
activation	leaky
shape	32,3,3,3
Kernel size	3
Pad	1
class	Darkflow.dark.convolution.convolutional_layer

Kemudian layer yang berhasil terbuat dimasukkan pada *placeholder* pada tensorflow dengan menggunakan kode berikut terutama pada baris 6 untuk memasukkan input

```
1.  def build_forward(self):
2.      verbalise = self.FLAGS.verbalise
3.
4.      # Placeholders
5.      inp_size = [None] + self.meta['inp_size']
6.      self.inp = tf.placeholder(tf.float32, inp_siz
    e, 'input')
7.      self.feed = dict() # other placeholders
8.
9.      # Build the forward pass
10.     state = identity(self.inp)
11.     roof = self.num_layer - self.ntrain
12.     self.say(HEADER, LINE)
13.     for i, layer in enumerate(self.darknet.layers
    ):
```

```
14.         scope = '{}-
    {}'.format(str(i),layer.type)
15.         args = [layer, state, i, roof, self.feed]

16.         state = op_create(*args)
17.         mess = state.verbalise()
18.         self.say(mess)
19.     self.say(LINE)
20.     self.top = state
21.     self.out = tf.identity(state.out, name='outpu
    t')
```

Kode 5.14 Potongan kode untuk menambahkan layer ke placeholder

Langkah awal dari memasukkan layer pada *placeholder*. awal *layer* akan dimasukkan pada *placeholder tensorflow*. Setelah itu membuat suatu *dictionary* bernama *feed()* kemudian setiap *layer* dimasukkan melalui *op_create* ketika setiap layer sudah masuk lalu dilakukan *enumerate* terhadap *layer* untuk memasukkan *layer* berikutnya. Hasil akhir dari pembuatan layer dapat terlihat pada gambar 5.2

```
Building net ...
Source | Train? | Layer description                | Output size
-------+--------+----------------------------------+---------------
       |        | input                            | (?, 416, 416, 3)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 416, 416, 32)
 Load  |  Yep!  | maxp 2x2p0_2                     | (?, 208, 208, 32)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 208, 208, 64)
 Load  |  Yep!  | maxp 2x2p0_2                     | (?, 104, 104, 64)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 104, 104, 128)
 Init  |  Yep!  | conv 1x1p0_1  +bnorm  leaky      | (?, 104, 104, 64)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 104, 104, 128)
 Load  |  Yep!  | maxp 2x2p0_2                     | (?, 52, 52, 128)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 52, 52, 256)
 Init  |  Yep!  | conv 1x1p0_1  +bnorm  leaky      | (?, 52, 52, 128)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 52, 52, 256)
 Load  |  Yep!  | maxp 2x2p0_2                     | (?, 26, 26, 256)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 26, 26, 512)
 Init  |  Yep!  | conv 1x1p0_1  +bnorm  leaky      | (?, 26, 26, 256)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 26, 26, 512)
 Init  |  Yep!  | conv 1x1p0_1  +bnorm  leaky      | (?, 26, 26, 256)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 26, 26, 512)
 Load  |  Yep!  | maxp 2x2p0_2                     | (?, 13, 13, 512)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 13, 13, 1024)
 Init  |  Yep!  | conv 1x1p0_1  +bnorm  leaky      | (?, 13, 13, 512)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 13, 13, 1024)
 Init  |  Yep!  | conv 1x1p0_1  +bnorm  leaky      | (?, 13, 13, 512)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 13, 13, 1024)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 13, 13, 1024)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 13, 13, 1024)
 Load  |  Yep!  | concat [16]                      | (?, 26, 26, 512)
 Init  |  Yep!  | conv 1x1p0_1  +bnorm  leaky      | (?, 26, 26, 64)
 Load  |  Yep!  | local flatten 2x2                | (?, 13, 13, 256)
 Load  |  Yep!  | concat [27, 24]                  | (?, 13, 13, 1280)
 Init  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 13, 13, 1024)
 Init  |  Yep!  | conv 1x1p0_1    linear           | (?, 13, 13, 45)
-------+--------+----------------------------------+---------------
```

Gambar 5.2 Hasil Pembuatan Layer

5.4.2 Training

Untuk menjalankan *training* pertama adalah dengan memasukan argument untuk jumlah epoch, batch, model, dataset, anotasi. Kemudian untuk pelaksanaan training dilakukan dengan kode berikut

```
1.  def train(self):
2.      loss_ph = self.framework.placeholders
3.      loss_mva = None; profile = list()
4.      batches = self.framework.shuffle()
5.      loss_op = self.framework.loss
6.
7.      for i, (x_batch, datum) in enumerate(batches)
    :
8.          if not i: self.say(train_stats.format(
9.              self.FLAGS.lr, self.FLAGS.batch,
10.             self.FLAGS.epoch, self.FLAGS.save
11.         ))
12.         feed_dict = {
13.             loss_ph[key]: datum[key]
```

```
14.                 for key in loss_ph }
15.         feed_dict[self.inp] = x_batch
16.         feed_dict.update(self.feed)
17.
18.         fetches = [self.train_op, loss_op]
```

Kode 5.15 Potongan kode untuk melakukan training

Pertama variabel dari *loss_ph* mengambil *placeholder layer* sebelumnya yang telah dibuat kemudian pada variabel batches memanggil *shuffle()*. Untuk memanggil data anotasi dan di inputkan secara acak pada *placeholder* dan *input loss* yang terdapat pada baris 4 dan 5. Pada *shuffle()* akan menghasilkan 2 varabel pertama adalah variabel *x_batch*, Yaitu variabel yang digunakan pada *placeholder input layer*, dan yang kedua adalah veriabel *datum* variabel yang digunakan pada *placeholder loss layer*. *Datum* berisi mengenai probabilitas, koordinat, area, posisi, kepercayaan, dari setiap kotak prediksi. Untuk input berisi mengenai informasi gambar yang diubah menjadi suatu *array* angka dimana sudah ukur ulang sesuai dengan *input* pada *input layer*.

Loss_op merupakan keluaran dari hasil penghitungan *loss* pada model. Penghitungan *loss function* terdapat 8 tahap diantaranya adalah pengambilan nilai *meta* arsitektur

```
1.  m = self.meta
2.      sprob = float(m['class_scale'])
3.      sconf = float(m['object_scale'])
4.      snoob = float(m['noobject_scale'])
5.      scoor = float(m['coord_scale'])
6.      H, W, _ = m['out_size']
7.      B, C = m['num'], m['classes']
8.      HW = H * W # number of grid cells
9.      anchors = m['anchors']
10.     print('{} loss hyper-
    parameters:'.format(m['model']))
11.     print('\tH       = {}'.format(H))
12.     print('\tW       = {}'.format(W))
13.     print('\tbox     = {}'.format(m['num']))
14.     print('\tclasses = {}'.format(m['classes']))
```

```
15.     print('\tscales  = {}'.format([sprob, sconf,
    snoob, scoor]))
16.
17.     size1 = [None, HW, B, C]
18.     size2 = [None, HW, B]
19. ...
```

Kode 5.16 deklarasi variabel untuk menghitung loss

Nilai pada meta dimasukkan pada variabel yang di deklarasikan pada fungsi tersebut kemudian dimasukkan pada tiap tiap placeholder di *tensorflow*

```
1.  # return the below placeholders
2.   _probs = tf.placeholder(tf.float32, size1)
3.      _confs = tf.placeholder(tf.float32, size2)
4.      _coord = tf.placeholder(tf.float32, size2 + [
    4])
5.      # weights term for L2 loss
6.      _proid = tf.placeholder(tf.float32, size1)
7.      # material calculating IOU
8.      _areas = tf.placeholder(tf.float32, size2)
9.      _upleft = tf.placeholder(tf.float32, size2 +
    [2])
10.     _botright = tf.placeholder(tf.float32, size2
    + [2])
11. ...
```

Kode 5.17 mengembalikan nilai - nilai yang terdapat ada layer

Kemudian berikutnya baru dilakukan penghitungan sesuai dengan penghitungan loss pada YOLO. Hasil keluaran loss tersebut dimasukkan pada *fetches*,*fetches* berisi *train optimizer* dan keluaran dari hasil penghitungan loss, untuk masukkan dan *placeholder* akan dimasukkan pada *feed_dict*

```
1.  if self.FLAGS.summary:
2.     fetches.append(self.summary_op)
3.
4.  fetched = self.sess.run(fetches, feed_dict)
5.  loss = fetched[1]
6.  if loss_mva is None: loss_mva = loss
7.  loss_mva = .9 * loss_mva + .1 * loss
```

```
8.  step_now = self.FLAGS.load + i + 1
9.
10. if self.FLAGS.summary:
11.     self.writer.add_summary(fetched[2], step_now)

12.
13. form = 'step {} - loss {} - moving ave loss {}'
14. self.say(form.format(step_now, loss, loss_mva))
15. profile += [(loss, loss_mva)]
16. ckpt = (i+1) % (self.FLAGS.save // self.FLAGS.bat
    ch)
17. args = [step_now, profile]
18. if not ckpt: _save_ckpt(self, *args)
19. if ckpt: _save_ckpt(self, *args)
```

Kode 5.18 Potongan kode untuk mendapatkan loss

Lalu kemudian *fetches* dan *feed_dict* dimasukkan pada fetched dan dijalankan menggunakan *self.sess.run()* dari menjalankan tersebut nilai *loss* akan masuk pada *fetched list* pertama

5.4.3 Pembutan prediksi

Untuk memanggil prediksi pertamanya dengan membuat program mengenai konfigurasi dan untuk mengembalikan hasil prediksi contoh pada kode dibawah

```
1.  import cv2
2.  from darkflow.net.build import TFNet
3.
4.  option = {
5.      'model': 'cfg/yolo-custom.cfg',
6.      'labels':'darkflow/labels.txt',
7.      'load': 51800,
8.      'threshold': 0.6,
9.      'gpu': 0.8
10. }
11. Capture = cv2.VideoCapture(xxxx)
12. Frame = capture.read()
13. results = tfnet.return_predict(frame)
```

Kode 5.19 Potongan kode untuk memanggil prediksi

Program tersebut memanggil fungsi mengembalikan prediksi serta memasukan argumen argumen yang diperlukan untuk melakukan prediksi

```
1.  def return_predict(self, im):
2.      assert isinstance(im, np.ndarray), \
3.                      'Image is not a np.ndarray'
4.      h, w, _ = im.shape
5.      im = self.framework.resize_input(im)
6.      this_inp = np.expand_dims(im, 0)
7.      feed_dict = {self.inp : this_inp}
8.
9.      out = self.sess.run(self.out, feed_dict)[0]
10.     boxes = self.framework.findboxes(out)
11.     threshold = self.FLAGS.threshold
12.     boxesInfo = list()
13.     for box in boxes:
14.         tmpBox = self.framework.process_box(box,
    h, w, threshold)
15.         if tmpBox is None:
16.             continue
17.         boxesInfo.append({
18.             "label": tmpBox[4],
19.             "confidence": tmpBox[6],
20.             "topleft": {
21.                 "x": tmpBox[0],
22.                 "y": tmpBox[2]},
23.             "bottomright": {
24.                 "x": tmpBox[1],
25.                 "y": tmpBox[3]}
26.         })
27.     return boxesInfo
```

Kode 5.20 Potongan kode untuk menghasilkan Prediksi

Input harus dalam bentuk *ndArray* jika tidak maka input pun ditolak. Kemudian ukuran dari gambar di atur ulang hingga sesuai dengan pada masukkan arsitektur yaitu 416x416 pada bais 5. Input tersebut diperluas agar dapat menjadi parameter yang sesuai pada *feed_dict.* Kemudian dijalankan menggunakan *sess.run()* untuk mendapatkan keluaran yang berguna untuk memprediksi objek. Kemudian hasil keluaran di lanjutkan ke *findboxes* untuk mencari objek yang dibutuhkan.

```
1.  def findboxes(self, net_out):
2.        # meta
3.        meta = self.meta
4.        boxes = list()
5.        boxes=box_constructor(meta,net_out)
6.        return boxes
```

Kode 5.21 Kode untuk melanjutkan ke darknet

Keluaran tersbut dimasukan lagi pada *box_constructor* dimana *box_consructor* melanjutkanya lagi ke *darknet* pada baris 5 dengan menggunakan *cython* untuk mencari objek pada gambar tersebut kemudian hasil dari pencarian akan dimasukkan pada *list boxes.*

Tabel 5.4 Hasil prediksi

Variabel	Hasil
x	0.54
y	0.44
w	0.37
h	0.12
probs	0.90
c	0.95

Dalam *box_constructor* menghasilkan prediksi berupa x,y,w,h yang merupakan posisi dan ukuran objek pada gambar dan c adalah nilai *confidence* dan probs adalah nilai probabilitas kelas .Hasil prediksi tersebut di proses lagi pada *process_box* guna untuk memproses hasil yang didapatkan dari *box_constructor* tersebut.

```
1.  def process_box(self, b, h, w, threshold):
2.      max_indx = np.argmax(b.probs)
3.      max_prob = b.probs[max_indx]
4.      label = self.meta['labels'][max_indx]
5.      if max_prob > threshold:
6.          left  = int ((b.x - b.w/2.) * w)
7.          right = int ((b.x + b.w/2.) * w)
8.          top   = int ((b.y - b.h/2.) * h)
9.          bot   = int ((b.y + b.h/2.) * h)
```

```
10.         if left  < 0    :  left = 0
11.         if right > w - 1: right = w - 1
12.         if top   < 0    :   top = 0
13.         if bot   > h - 1:   bot = h - 1
14.         mess = '{}'.format(label)
15.         return (left, right, top, bot, mess, max_
    indx, max_prob)
16.     return None
```

Kode 5.22 Kode untuk memroses box

Pada *process_box* memroses hasil prediksi yang didapatkan dari *box_constructor*. Hasil dari prediksi tersebut dicari mana angka index maksimal dan angka probabilitas maksimal indeks menunjukan kelas yang di prediksi kemudian indeks tersebut diberikan label sesuai dengan indeks tersebut. Kemudian dipilih objek yang mana saja yang memiliki angka yang probabilitas yang lebih besar dari *threshold* pada baris 5 kemudian dihitung nilai bawah kiri kanan belakang dari nilai w dan h yang didapatkan dikalikan dengan ukuran input layer pada baris 6 hingga 8. Hal ini dilakukan untuk mengetahui posisi objek pada gambar sebenarnya. Kemudian nantinya akan menghasilkan keluaran yang dapat dilihat pada Tabel 5.5

Tabel 5.5 Hasil prediksi yang telah diproses

Variabel	Hasil
Confidence	0.90
Topleft	X=755;y=410
Bottomright	X=809;y=535

5.5 Pembuatan sistem pengklasifikasian dan penghitungan kendaraan

Sistem pengklasifikasian ini akan mengklasifikasikan kendaraan dengan menggambarkan kotak ke kendaraan tersebut dan pemberian label pada objek yang diklasifikasikan. Sistem ini juga berfungsi untuk input video dan memutar video tersebut dalam frame per frame

```
from darkflow.net.build import TFNet
import numpy as np
import time
import matplotlib.pyplot as plt
import datetime as dt

option = {
    'model': 'cfg/yolo-custom.cfg',
    'labels':'darkflow/labels.txt',
    'load': 51800,
    'threshold': 0.6,
    'gpu': 0.8
}
tfnet = TFNet(option)
capture = cv2.VideoCapture('sample_video/video5.MP4')
colors = [tuple(255 * np.random.rand(3)) for i in range(255)]

plt.show()

loop = 1
loop_arr = []
total_mobil_arr = []
total_motor_arr = []
total_truk_arr = []
total_bus_arr = []
```

Kode 5.23 Kode untuk menginisiasi konfigurasi sistem

Pada potongan kode 5.23 pertama adalah menginputkan argumen argumen apa saja yang akan dipakai untuk melakukan prediksi berikutnya adalah memasukkan argumen tersebut pada *darkflow* untuk dijalankan. Berikutnya sistem mengambil gambar ataupun video dengan menggunakan cv2.

Kemudian untuk penghitungan dibuat terlebih dahulu suatu wadah array untuk di inputkan jumlah setiap satuan waktu

```
while (capture.isOpened()):
    stime = time.time()
    ret, frame = capture.read()
    if ret:
        results = tfnet.return_predict(frame)
```

```
6.          total_mobil = 0
7.          total_motor = 0
8.          total_truk = 0
9.          total_bus = 0
10.         waktu = dt.datetime(2019, 2, 15, 10, 30,
    20)
11.         for color, result in zip(colors, results)
    :
12.             tl = (result['topleft']['x'], result[
    'topleft']['y'])
13.             br = (result['bottomright']['x'], res
    ult['bottomright']['y'])
14.             label = result['label']
15.             confidence = result['confidence']
16.             text = '{}: {:.0f}%'.format(label, co
    nfidence * 100)
17.             frame = cv2.rectangle(frame, tl, br,
    color, 7)
18.             frame = cv2.putText(frame, text, tl,
    cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 0), 2)
19.             if label == 'mobil':
20.                 total_mobil += 1
21.             if label == 'motor':
22.                 total_motor += 1
23.             if label == 'truk':
24.                 total_truk += 1
25.             if label == 'bus':
26.                 total_bus +=1
```

Kode 5.24 Kode untuk menbikin box

Selama pengambilan gambar dijalankan setiap gambar nya di *decode*. Hasil *decode* tersebut dilakukan prediksi oleh *darkflow* dimana hasil prediksi akan disimpan pada variabel *results* yang terspar baris 5. Kemudian pada baris berikutnya untuk jumlah kendaraan dan waktu di inisiasi dan waktu di isikan dengan waktu pada saat pengambilan gambar.

Kemudian sistem membuat suatu kotak untuk objek yang terdeteksi dengan masukkan dari hasil *results* yang terdapat pada baris 11 hingga 18. Kotak dibuat dengan menginputkan posisi posisi atas kiri dan bawah kanan pada objek yang terdeteksi bedasarkan *results*. Setiap objek yang terdeteksi juga

diberikan suatu label yang berisikan hasil klasifikasi objek tersebut.

Setiap kendaraan yang terdeteksi hasil klasifikasi tersebut akan menambahkan angka 1 pada variabel yang terinisiasi pada setiap jenis kendaraanya

```
1.  if loop % 30 == 0:
2.              waktu_loop = waktu + dt.timedelta(sec
    onds= loop/30)
3.              loop_arr.append(waktu_loop.strftime('
    %H:%M:%S'))
4.              total_mobil_arr.append(total_mobil)
5.              total_motor_arr.append(total_motor)
6.              total_truk_arr.append(total_truk)
7.              total_bus_arr.append(total_bus)
8.
9.              loop_arr = loop_arr[-20:]
10.             total_mobil_arr = total_mobil_arr[-
    20:]
11.             total_motor_arr = total_motor_arr[-
    20:]
12.             total_truk_arr = total_truk_arr[-
    20:]
13.             total_bus_arr = total_bus_arr[-20:]
```

Kode 5.25 Potongan kode untuk menghitung jumlah kendaraan

Dari jumlah yang didapatkan dari tiap kelasnya berikutnya ditambahkan ke tiap tiap masing *list* pada baris 4 hingga 7. Dan juga dibataskan jumlahnya menjadi 20 list agar tak tertumpuk pada pembuatan grafiknya hal tersebut terdapat pada baris 9 hingga 13. Untuk penambahan waktu dari waktu inisiasi digunakan *timedelta* untuk penambahan waktu dengan variable dari jumlah frame dibagi dengan 30

```
1.      a = total_mobil_arr[-1]
2.      b = total_truk_arr[-1]
3.      c = total_bus_arr[-1]
4.      d = total_motor_arr[-1]
5.
6.      plt.cla()
7.      plt.xticks(rotation=45, ha='right')
```

```
8.      plt.subplots_adjust(bottom=0.30)
9.      plt.plot(loop_arr, total_mobil_arr,label='mob
    il = '+str(a))
10.     plt.plot(loop_arr, total_motor_arr, label='mo
    tor = '+str(d))
11.     plt.plot(loop_arr, total_bus_arr, label='bus
    = '+str(c))
12.     plt.plot(loop_arr, total_truk_arr, label='tru
    k = '+str(b))
13.     plt.title('Jumlah kendaraan Jl. Bungurasih \n
    jumlah kendaraan = '+str(a + b + c + d)+ '\n tang
    gal '+ waktu_loop.strftime('%d-%m-%Y'))
14.     plt.legend()
15.     plt.pause(0.0001)
16.
17. loop += 1
```

Kode 5.26 Potongan kode memasukkan array pada grafik

Dari *list* yang dibuat sebelumnya dibuat dimasukan pada plot(x,y) pada baris 9 hiingga 12 untuk dibentuk kurva grafik dan juga menambahkan judul, jumlah, legenda pada kurva grafik. Untuk plot x adalah waktu pada video dan plot y adalah jumlah kendaraan yang terdapat pada video

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini akan menjelaskan mengenai hasil dan pembahasan dari implementasi sistem, uji training, dan tes validasi terhadap penggunaan arsitektur YOLOv2.

6.1 Hasil Pelabelan

Untuk hasil dari pelabelan diambil 300 gambar dari masing masing tiap sisi dan untuk jumlah objek yang dihasilkan untuk setiap kelasnya terdapat pada tabel 6.1

Tabel 6.1 Jumlah objek hasil pelabelan gambar

Nama	Jumlah
Mobil	8936
Bus	462
Truk	1681
Motor	10119
TOTAL	21198

Dengan jumlah hasil pengklasifikasian tersebut jumlah dataset memiliki tidak keseimbangan dimana jumlah dari truk dan bus lebih sedikit dari pada mobil dan motor

6.2 Hasil Training Model YOLOv2

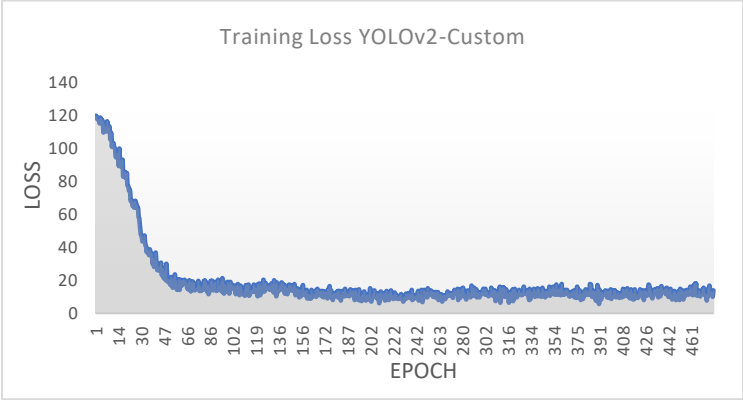
Sebelum melakukan training pertama adalah perlu dilakukanya konfigurasi terhadap berapa persen pemanfaatan GPU komputer dan berapa ukuran *batch* yang diinginkan berikut merupakan konfigurasi yang digunakan

Tabel 6.2 tabel Konfigurasi training

Konfigurasi	Jumlah
Persentasi pemakaian GPU	80%
Ukuran Batch	10
Jumlah Epoch	500

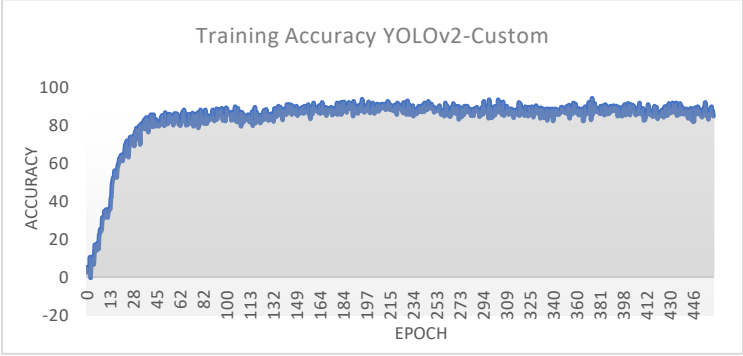
Dengan konfigurasi terssebut training dilakukan dalam waktu kurang lebih 10 jam. Dan dalam prosesnya menghasilkan *loss*

pada setiap epochnya dimana dalam setiap *epoch*, *loss* akan berkurang. Untuk perkembangan berkurangnya terlihat pada



Gambar 6.1 Loss Function Model

Seiring dengan berkurangnya *Loss*, Akurasi akan juga bertambah pada setiap *epoch*-nya berikut adalah grafik pertambahan akurasi pada Gambar 6.2



Gambar 6.2 Akurasi model

Dari setiap iterasi tersebut menghasilkan hasil akhir loss dan akurasi sebagai berikut

Tabel 6.3 hasil loss dan akurasi

Loss	15.1105
Akurasi	84.8895

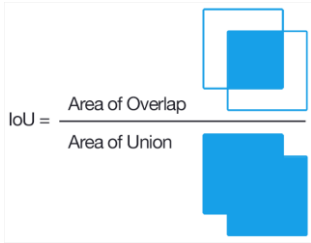
6.3 Hasil tes validasi

Pengujian dilakukan pada jumlah dataset tiap masing masing kelas dengan jumlah sebagai berikut

Tabel 6.4 Jumlah objek pengetesan

Nama	Jumlah
Mobil	2089
Bus	123
Truk	350
Motor	1987

Hasil keluaran dari training yang berupa bobot kemudian diuji lagi. Pengujian menggunakan satuan mAP, mAP didapatkan dari penghitungan IoU antara *ground truth* dengan yang dipredikasi dimana dijelaskan pada



Gambar 6.3 Gambaran rumus IoU

Pada gambar 6.3 dijelaskan IoU didapatkan dari area diantara area kepercayaan dan prediksi dibagi dengan union area. Sebuah objek akan bernilai true positive jika suatu objek antara yang diprediksi dengan kepercayaan memiliki label yang sama dan IoU memiliki nilai lebih dari sama dengan 0.5.

Untuk penelitian kali ini sebelum melakukan pengujian diperlukan beberapa konfigurasi diantaranya adalah sebagai berikut

Tabel 6.5 Konfigurasi Pengujian

Konfigurasi	Jumlah
Penggunaan GPU	80%
threshold	0.5

Pada tabel 6.5 Untuk konfigurasi yang digunakan menggunakan presentase penggunaan GPU pada 80 %. Dan *threshold* yang digunakan adalah 0.5. *threshold* adalah Batasan terhadap nilai *Confidence Level* agar prediksi yang dikeluarkan oleh model dapat dikeluarkan atau tidak.

Selain hal yang telah disebutkan sebelumnya, model dilatih dengan tingkat pembelajaran 0,001 dan momentum 0,9. dan *Leaky Rectified Linear Unit* (Leaky RELU) juga digunakan sebagai fungsi aktivasi. Selain itu, algoritma *Stochastic Gradient Descent (SGD)* yang terkenal juga diterapkan untuk memperbarui *learning paremeter* hingga konvergen.

Berikut adalah salah satu contoh gambar yang memiliki nilai *true positive dan false positive*

Gambar 6.4 Contoh true positive

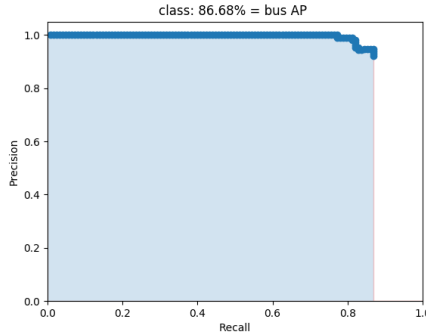
Pada gambar terdapat bus sebagai objek. Dan terdapat kotak yang dimana hijau menandakan prediksi dan biru menandakan kotak kepercayaan dan prediksi tersebut adalah menghasilkan prediksi true positif dikarenakan model dapat memprediksi bus tersebut dengan menghasilkan IoU 87.85% dimana keika IoU terbilang positif adalah diatas 50%. Sementara untuk yang false positive adalah sebagai berikut

Gambar 6.5 Contoh false positive

Pada gambar tesebut terdapat kotak berwarna merah yang menandakan bahwa model kurang tepat dalam memprediksi posisi objek sehingga dapat menghasilkan IoU dengan nilai yang kurang IoU pada objek tersebut bernilai hanya 40% sehingga bernilai false positive. Hasil dari total yang memiliki *false positive* dan *true positive* adalah sebagai berikut

Gambar 6.6 Hasil deteksi model

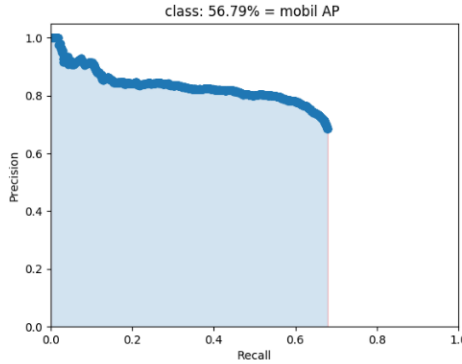
Dari data *false positive* dan true positive tersebut dapat dihasilkan precision dan recall. Berikut adalah hasil dari precision recall dari setiap kelasnya



Gambar 6.7 Precision recall kelas bus

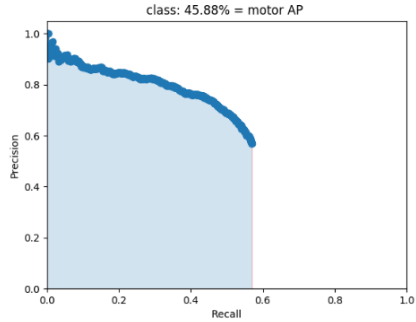
Pada Gambar 6.7 *Precission* dan *recall* pada kelas bus terlihat stabil pada angka satu tetapi ketika *recall* mencapai jumlah 0.8

Precission mulai menurun untuk *Average Precission (AP)* dihitung dari luasan warna biru dibandingkan dengan warna putih sehingga AP dari bus mencapai 86.68%



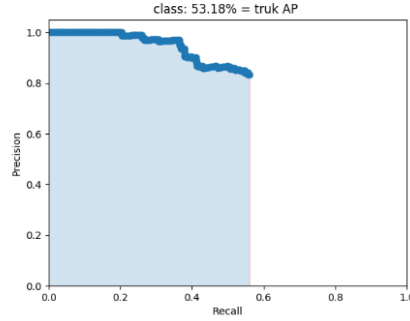
Gambar 6.8 Precision recall kelas mobil

Untuk kelas mobil pada Gambar 6.8 mencapai AP 56.79%. pada *recall* dibawah 0.2 terdapat penurunan yang signifikan ketika sudah melewati 0.2 mulai seimbang



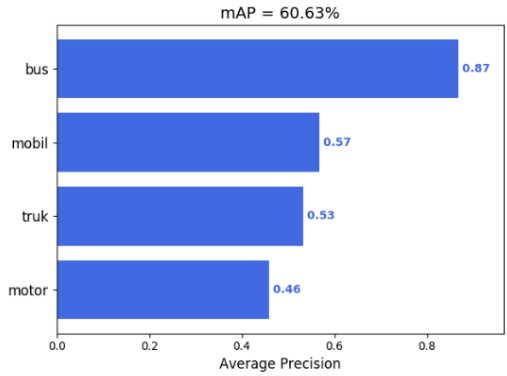
Gambar 6.9 Precision recall kelas motor

Untuk motor makin tinggi *recall* menunjukan penurunan yang signifikan *recall* mencapai nilai tidak mencapai 0.6 dan recall



Gambar 6.10 Precision recall kelas truk

Dari hasil precision recall tersebut dapat diketahui *Average Precision (AP)* dari tiap tiap kelasnya. Sehingga dapat juga diketahui *Mean Average Precision (mAP)*. Untuk nilai dari mAP dapat dilihat pada gambar



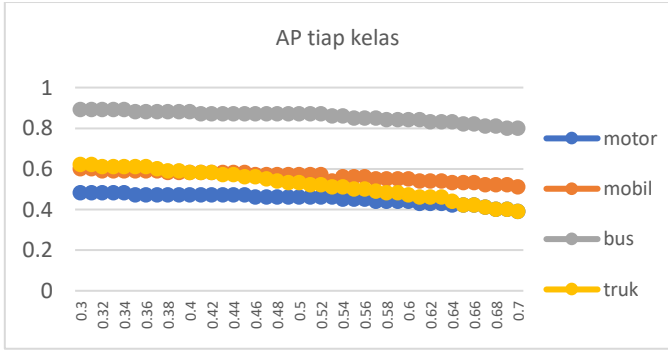
Gambar 6.11 Hasil mAP model

Pada pengujian konfigurasi *threshold* diatur pada 0.5 menghasilkan mAP sebesar 60.63%. pada grafik tersebut terlihat bahwa bus memiliki AP yang tinggi dibanding yang lain yaitu memiliki AP pada 0.87. sedangkan motor memiliki AP yang kecil dibanding dengan kelas lain. Kemudian pengujian dilakukan pada kondisi trhreshold yang berbeda hasil pengujian tersebut terlihat pada tabel 6.6

Tabel 6.6 Perbandingan mAP tiap threshold

threshold	AP				mAP
	motor	mobil	Bus	truk	
0.3	0.48	0.60	0.89	0.62	64.85%
0.4	0.47	0.58	0.88	0.58	62.95%
0.5	0.46	0.57	0.87	0.53	60.63%
0.6	0.44	0.55	0.84	0.47	57.59%
0.7	0.39	0.51	0.80	0.40	52.50%

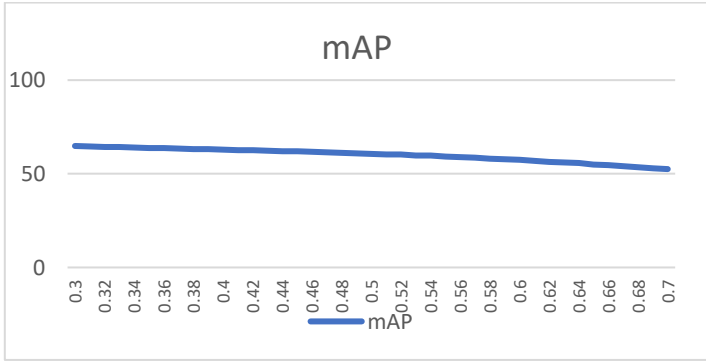
Pada tabel 6.5 terlihat bahwa semakin rendah threshold semakin tinggi angka mAP namun dengan rendahnya threshold menurunkan angka *true positive* pada gambar yang diprediksi. Dapat terlihat pada gambar



Gambar 6.12 *Average Precission* tiap kelas

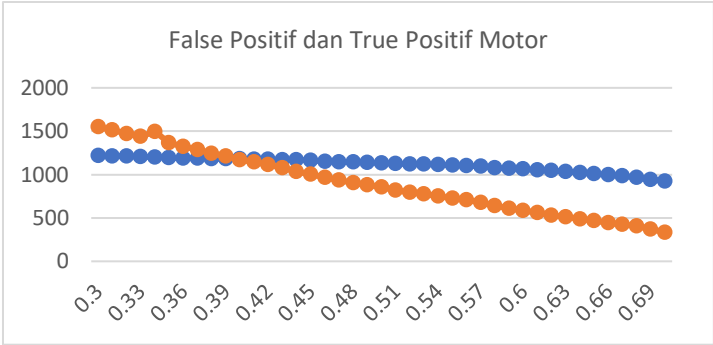
Untuk AP tiap kelas bis memiliki AP yang mengungguli dari kelas lain sedangkan motor yang memiliki AP yang rendah dibandingkan kelas lain. untuk mobil dan motor pada jumlah *threshold* awal 0.3 memiliki AP yang hampir sama ketika mencapai *threshold* yang besar truk mulai berkurang sehingga pada akhir truk memiliki nilai dibawah mobil dan juga AP truk memiliki penurunan yang signifikan dalam bertambahnya AP.

Dari data tersebut Bus merupakan objek yang mudah dikenali oleh YOLOv2. Terlihat dengan nilai AP yang mengungguli semua objek yang dideteksi. Hal ini dimungkinkan karena karakteristik objek pada Bus yang besar dan memiliki lekuk tubuh yang cenderung sama.



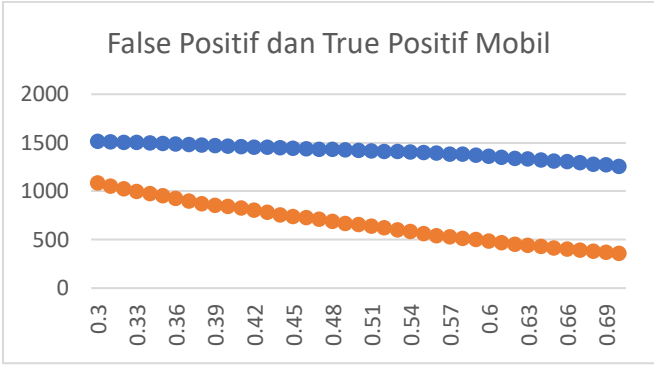
Gambar 6.13 grafik mAP

Dapat terlihat pada grafik bahwa semakin bertambahnya *threshold* mAP semakin menunjukan penurunan. Hal ini dikarenakan *true positive* bertambah namun hal ini juga berdampak pada *false positive*



Gambar 6.14 False positive dan True Positive Motor

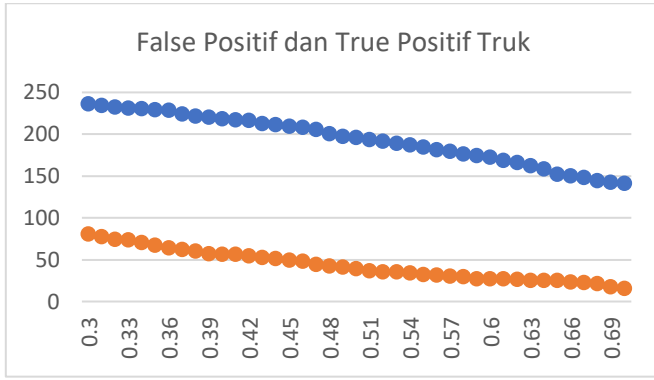
Untuk *true positive* pada motor cenderung seimbang namun jumlahnya pada *threshold* yang lebih kecil memiliki angka yang lebih kecil



Gambar 6.15 False Positive dan True Positive Mobil

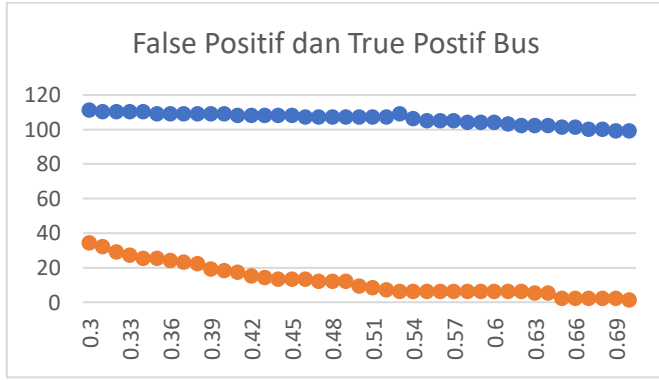
True postive pada mobil cenderung seimbang dan tidak mengalami penurunan yang signifikan tetapi untuk *false positive* memiliki penurunan signifikan pada *threshold* yang tinggi sehingga untuk batas aman untuk mendeteksi mobil adalah sekitar pada 0.5 dikarenakan jumlah *true positive* tidak

mengalami penurunan signifikan pada nilai tersebut namun *false positive* memiliki jumlah lebih sedikit pada angka *threshold* tersebut



Gambar 6.16 *False Positive dan True Positive* truk

Truk memiliki nilai *True positive* yang tinggi pada *threshold* yang rendah namun memiliki nilai False Positive yang tinggi juga pada *threshold* tersebut. Nilai *True Positive* dan *False Positive* memiliki selisih yang besar terdapat pada *threshold 0.3* hingga 0.4



Gambar 6.17 False Positive dan True Positive bus

Bus memiliki nilai *true positive* yang cenderung seimbang pada tiap *threshold*-nya dan *false positive* mengalami penurunan tiap

peningkatan angka *threshold*-nya. Angka *threshold* terbaik yang digunakan adalah 0.5 hingga 0.55

Besar kecil nya angka *false positive* dan *true positive* disebabkan oleh besar dan kecilnya ukuran objek terhadap gambar. Pada gambar 6.18 adalah contoh hasil deteksi dari YOLOv2

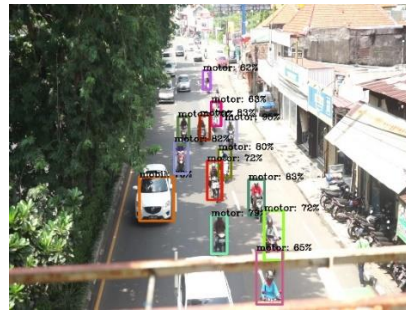
Gambar 6.18 hasil prediksi YOLOv2

Pada gambar 6.18 terlihat bahwa pada objek yang jauh, YOLOv2 sudah tidak dapat untuk melakukan deteksi secara akurat. dan pada objek yang dekat YOLOv2 masih memiliki akurasi yang baik

6.4 Visualisasi penghitungan kendaraan

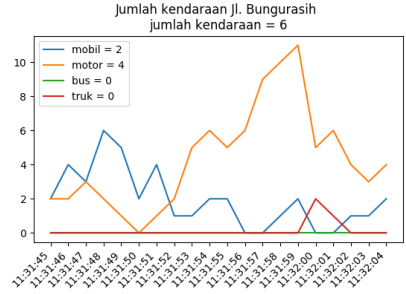
Setelah dilakukan training dan pengujian pada model kemudian dibuat untuk menampilkan hasil dari melakukan klasifikasi. YOLOv2 dalam pengujian ini dapat mendapatkan diantara 20 hingga 35 *Frame Per Seconds (FPS)*. Dimana dengan hasil

tersebut YOLOv2 dapat melakukan klasifikasi dengan waktu yang hampir menyamai waktu nyata.



Gambar 6.19 Contoh gambar hasil klasifikasi

Pada gambar 6.19 merupakan bagaimana hasil klasifikasi dilakukan dengan menggunakan video, kemudian hasil penghitungan dilakukan secara bersamaan dengan video



Gambar 6.20 Visualisasi penghitungan Kendaraan

Kemudian hasil video klasifikasi juga dapat diliat secara bersamaan dalam waktu nyata tetapi dalam *window* yang berbeda yang terlihat pada gambar 6.20

Gambar 6.21 Hasil akhir penghitungan dan pengklasifikasian kendaraan

Pada gambar 6.21 merupakan hasil dari klasifikasi dan penghitungan kendaraan dapat dilakukan penghitungan dan pengklasifikasian dapat dijalankan secara bersamaan secara realtime dengan FPS yang cepat dimana diantara 20 hingga 35 FPS

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan dibahas mengenai kesimpulan dan saran dari semua proses yang dilakukan pada penelitian tugas akhir kali ini yang dapat diberikan untuk pengembangan yang lebih baik

7.1 Kesimpulan

Kesimpulan yang didapatkan dari proses pengerjaan tugas akhir kali ini antara lain :

1. Semakin kecil *confidence level* yang digunakan semakin besar nilai mAP yang didapatkan, tetapi memperbanyak jumlah objek yang memiliki *false positive.* Hal itu disebabkan terdapat objek yang berukuran kecil yang menyebabkan YOLOv2 kurang mampu untuk mendeteksi objek tersebut
2. YOLOv2 memiliki pendeteksian yang lebih baik untuk mendeteksi Bus karena memiiki nilai AP yang lebih besar pada kelas lain
3. Untuk motor memiliki angka yang baik melakukan deteksi pada *confidence level* minimal 0.40 hingga 0.50, mobil adalah 0.5, bus 0.5 hingga 0.55, truk 0.3 hingga 0.4
4. YOLOv2 memiliki kemampuan deteksi dengan performa dari 20 FPS hingga 35 FPS. Dengan angka tersebut YOLOv2 mampu mengeluarkan hasil deteksi menyamai waktu nyata

7.2 Saran

Dari pengerjaan tugas akhir ini, adapun beberapa saran untuk pengembangan penelitian ke depan

1. Untuk Pengerjaan YOLOv2 dapat menggunakan *darkflow* sebagai *tools* untuk pembuatan bobot dan mengeluarkan hasil prediksi

2. Pengambilan objek pada dataset lebih baik menggunakan objek yang lebih besar untuk lebih memvalidasi dari arsitektur YOLOv2
3. *Training* data dapat dilakukan lebih lama lagi dari waktu 10 jam untuk mendapatkan nilai mAP yang lebih baik
4. Memparbanyak jumlah dataset agar menambah tingkat akurasi
5. Mengambil data mentah pada lokasi yang berbeda agar dapat program dapat menyesuaikan dalam berbagai lingkungan

DAFTAR PUSTAKA

[1] Badan Pusat Statistik, "Perkembangan Jumlah Kendaraan Bermotor Menurut Jenis, 1949-2016." [Online]. Available: https://www.bps.go.id/linkTableDinamis/view/id/1133. [Accessed: 14-Jan-2019].

[2] R. Pratama, "Jumlah Kendaraan Kian Banyak, Membanggakan Tapi bikin Nggak Sehat," 2018. [Online]. Available: https://oto.detik.com/mobil/d-4242211/jumlah-kendaraan-kian-banyak-membanggakan-tapi-bikin-nggak-sehat. [Accessed: 16-Jan-2019].

[3] S. Febrina Laucereno, "OJK Akan Izinkan Uang Muka Kredit Motor dan Mobil 0%." [Online]. Available: https://finance.detik.com/moneter/d-4175644/ojk-akan-izinkan-uang-muka-kredit-motor-dan-mobil-0. [Accessed: 14-Jan-2019].

[4] P. Maulidya, "Kemacetan di Surabaya Bisa Dipantau Lewat Aplikasi SITS - Surya," 2018. [Online]. Available: http://surabaya.tribunnews.com/2018/10/11/kemacetan-di-surabaya-bisa-dipantau-lewat-aplikasi-sits. [Accessed: 15-Jan-2019].

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015.

[6] Susanto, E. Rudiawan, R. Analia, P. D. Sutopo, and H. Soebakti, "The deep learning development for real-time ball and goal detection of barelang-FC," *Proc. IES-ETA 2017 - Int. Electron. Symp. Eng. Technol. Appl.*, vol. 2017–Decem, pp. 146–151, 2017.

[7] M. Wang, M. Liu, F. Zhang, G. Lei, J. Guo, and L. Wang, "Fast Classification and Detection of Fish Images

with YOLOv2," in *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*, 2018, pp. 1–4.

[8] S. Kul, S. Eken, and A. Sayar, "A concise review on vehicle detection and classification," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, vol. 2018–Janua, pp. 1–4.

[9] "PERATURAN PEMERINTAH REPUBLIK INDONESIA NOMOR 55 TAHUN 2012 TENTANG KENDARAAN," 2012.

[10] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] Mathworks, "Introducing Deep Learning with MATLAB," in *Introducing Deep Learning with MATLAB*, 2017, p. 15.

[12] K. Fukushima, "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biol. Cybernatics*, vol. 36, no. 5349, pp. 193–202, 1980.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] MathWorks, "Introduction to Deep Learning: What Are Convolutional Neural Networks? Video - MATLAB." [Online]. Available: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html. [Accessed: 14-Feb-2019].

[15] Mathworks, "Object detection in computer vision." [Online]. Available: https://www.mathworks.com/discovery/object-detection.html?s_tid=srchtitle. [Accessed: 28-Jan-2019].

[16] Z.-Q. Zhao, P. Zheng, S. Xu, and X. Wu, “Object Detection with Deep Learning: A Review,” *Econ. Educ. Rev.*, vol. 3, no. 3, pp. 231–245, Jul. 2018.

[17] F. Li, A. Karpathy, and J. Johnson, “Lecture 8 : Spatial Localization and Detection,” *Stanford 231 Comput. Sci. Course Mater.*, pp. 1–90, 2016.

[18] M. A. Fischler and R. A. Elschlager, “The Representation and Matching of Pictorial Structures,” *IEEE Trans. Comput.*, vol. C-22, no. 1, pp. 67–92, Jan. 1973.

[19] E. Osuna, R. Freund, and G. Federico, “Training Support Vector Machines: an Application to Face Detection,” in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, vol. 27, no. 2, pp. 104–112.

[20] H. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 23–28, 1977.

[21] P. Viola and M. Jones, “Robust Real-Time Face Detection,” *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[22] C. H. Lampert, M. B. Blaschko, and T. Hofmann, “Efficient subwindow search: A branch and bound framework for object localization,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2129–2142, 2009.

[23] C. Garcia and M. Delakis, “A neural architecture for fast and robust face detection,” *Object Recognit. Support. by user Interact. Serv. Robot.*, vol. 2, no. 11, pp. 44–47, 2004.

[24] “LabelImg,” 2017. [Online]. Available: https://devhub.io/repos/tzutalin-labelImg. [Accessed: 14-Feb-2019].

[25] FFmpeg, “About FFmpeg.” [Online]. Available: https://www.ffmpeg.org/about.html. [Accessed: 15-Feb-2019].

[26] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017–Janua, pp. 6517–6525, Dec. 2016.

BIODATA PENULIS

Penulis lahir di Surabaya pada tanggal 23 September 1997. Merupakan anak keduadari 2 bersaudara. Penulis telah menempuh pendidikan formal yaitu : SD Muhammadiyah 4 Surabaya, SMPN 17 Surabaya, SMAN 15 Surabaya.

Pada tahun 2015 pasca kelulusan SMA. Penulis melanjutkan Pendidikan dengan mengikuti jalur SBMPTN di Departemen Sistem Informasi FTIK – Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 05211540000079. Selama mahasiswa penulis mengikuti berbagai kegiatan kampus seperti Departemen Organization Social Responsibility (OSR) di BEM FTIK selain itu juga penulis pernah mengikuti kegiatan kepanitiaan pada ISE 2016, dan 2017 dan pada 2017 diamanahkan menjadi coordinator keamanan dan perijinan.

Pada tahun keempat, karena memiliki ketertarikan di bidang pengolahan data, maka penulis mengambil bidang minat Akuisisi Data dan Diseminasi Informasi (ADDI). Penulis dapat dihubungi melalui email di yoga.g54g@gmail.com

Halaman ini sengaja dikosongkan

LAMPIRAN A. mAP tiap threshold

threshold	AP				mAP
	motor	mobil	bus	truk	
0.3	0.48	0.6	0.89	0.62	64.85
0.31	0.48	0.6	0.89	0.62	64.55
0.32	0.48	0.59	0.89	0.61	64.37
0.33	0.48	0.59	0.89	0.61	64.27
0.34	0.48	0.59	0.89	0.61	64.16
0.35	0.47	0.59	0.88	0.61	63.87
0.36	0.47	0.59	0.88	0.61	63.74
0.37	0.47	0.59	0.88	0.6	63.48
0.38	0.47	0.58	0.88	0.59	63.22
0.39	0.47	0.58	0.88	0.59	63.12
0.4	0.47	0.58	0.88	0.58	62.95
0.41	0.47	0.58	0.87	0.58	62.66
0.42	0.47	0.58	0.87	0.58	62.55
0.43	0.47	0.58	0.87	0.57	62.28
0.44	0.47	0.58	0.87	0.57	62.17
0.45	0.47	0.58	0.87	0.56	61.97
0.46	0.46	0.57	0.87	0.56	61.65
0.47	0.46	0.57	0.87	0.55	61.38
0.48	0.46	0.57	0.87	0.54	61.05
0.49	0.46	0.57	0.87	0.53	60.8
0.5	0.46	0.57	0.87	0.53	60.63
0.51	0.46	0.57	0.87	0.52	60.4
0.52	0.46	0.57	0.87	0.52	60.2
0.53	0.46	0.54	0.86	0.51	59.85
0.54	0.45	0.56	0.86	0.51	59.65
0.55	0.45	0.56	0.85	0.5	59.16
0.56	0.45	0.56	0.85	0.5	58.87
0.57	0.44	0.55	0.85	0.49	58.64

threshold	AP				mAP
	motor	mobil	bus	truk	
0.58	0.44	0.55	0.84	0.48	58.1
0.59	0.44	0.55	0.84	0.48	57.86
0.6	0.44	0.55	0.84	0.47	57.59
0.61	0.43	0.54	0.84	0.46	56.92
0.62	0.43	0.54	0.83	0.46	56.45
0.63	0.43	0.54	0.83	0.46	56.04
0.64	0.42	0.53	0.83	0.44	55.66
0.65	0.42	0.53	0.82	0.42	54.87
0.66	0.42	0.53	0.82	0.42	54.55
0.67	0.41	0.52	0.81	0.41	54.01
0.68	0.4	0.52	0.81	0.4	53.46
0.69	0.4	0.52	0.8	0.4	52.85
0.7	0.39	0.51	0.8	0.39	52.5

LAMPIRAN B. False Positive dan True Positive tiap threshold

threshold	Motor		Mobil		Bus		Truk	
	TP	FP	TP	FP	TP	FP	TP	FP
0.3	1217	1549	1513	1085	111	34	236	80
0.31	1213	1517	1511	1053	110	32	234	77
0.32	1210	1469	1504	1022	110	29	232	74
0.33	1204	1438	1502	996	110	27	231	73
0.34	1199	1498	1498	974	110	25	230	70
0.35	1193	1364	1494	952	109	25	229	67
0.36	1189	1323	1486	924	109	24	228	64
0.37	1187	1284	1483	897	109	23	224	62
0.38	1185	1246	1473	872	109	22	221	60
0.39	1181	1215	1471	854	109	19	220	57
0.4	1179	1169	1465	841	109	18	218	56
0.41	1176	1145	1460	824	108	17	217	56
0.42	1174	1113	1455	806	108	15	216	54
0.43	1170	1075	1452	784	108	14	212	52
0.44	1168	1035	1447	753	108	13	211	51

threshold	Motor		Mobil		Bus		Truk	
	TP	FP	TP	FP	TP	FP	TP	FP
0.45	1163	1005	1443	739	108	13	209	49
0.46	1153	966	1439	726	107	13	208	48
0.47	1146	939	1434	708	107	12	205	44
0.48	1143	906	1432	688	107	12	200	42
0.49	1137	883	1428	669	107	12	197	41
0.5	1132	855	1419	653	107	9	196	39
0.51	1126	821	1416	640	107	8	193	36
0.52	1122	797	1412	623	107	7	191	35
0.53	1121	774	1409	600	109	6	189	35
0.54	1116	751	1404	582	106	6	187	34
0.55	1106	728	1399	561	105	6	184	32
0.56	1099	708	1393	539	105	6	181	31
0.57	1093	675	1383	531	105	6	179	30
0.58	1079	639	1380	510	104	6	176	29
0.59	1073	611	1372	501	104	6	174	27
0.6	1065	584	1362	484	104	6	172	27
0.61	1054	559	1347	468	103	6	168	27

threshold	Motor		Mobil		Bus		Truk	
	TP	FP	TP	FP	TP	FP	TP	FP
0.62	1045	531	1337	452	102	6	166	26
0.63	1033	510	1331	441	102	5	162	25
0.64	1025	485	1324	429	102	5	158	25
0.65	1011	469	1312	415	101	2	152	25
0.66	997	447	1304	404	101	2	150	23
0.67	984	424	1292	393	100	2	148	22
0.68	964	404	1279	381	100	2	144	21
0.69	941	368	1271	371	99	2	142	17
0.7	926	336	1254	357	99	1	141	15