TUGAS AKHIR – TF 181801

SISTEM DETEKSI PLAT NOMOR OTOMATIS MENGGUNAKAN ALGORITMA YOLO V3 DENGAN FRAMEWORK DARKNET

FARAH QOONITA SYUHAILA
NRP. 02311640000142

Dosen Pembimbing:
Dr.rer.nat.Ir. Aulia M.T Nasution,M.Sc.

Departemen Teknik Fisika
Fakultas Teknologi Industri dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember
Surabaya
2020

TUGAS AKHIR - TF 181801

SISTEM DETEKSI PLAT NOMOR OTOMATIS MENGGUNAKAN ALGORITMA YOLO V3 DENGAN *FRAMEWORK DARKNET*

FARAH QOONITA SYUHAILA
NRP. 0231164.0000120

Dosen Pembimbing:
Dr.rer.nat.Ir. Aulia M.T. Nasution, M.Sc.

Departemen Teknik Fisika
Fakultas Teknologi Industri Dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember
Surabaya
2020

Halaman ini sengaja dikosongkan

FINAL PROJECT- TF 181801

AUTOMATIC LICENSE PLATE DETECTION SYSTEM USING YOLO *V3 ALGORITHM WITH DARKNET FRAMEWORK*

FARAH QOONITA SYUHAILA
NRP. 0231164.0000120

Supervisor:
Dr.rer.nat. Ir. Aulia M.T. Nasution, M.Sc.

Department of Engineering Physics
Faculty of Industrial Technology and System Engineering
Institut Teknologi Sepuluh Nopember
Surabaya
2020

Halaman ini sengaja dikosongkan

PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini.

Nama	: Farah Qoonita Syuhaila
NRP	: 02311640000120
Departemen/ Prodi	: Teknik Fisika/ S1 Teknik Fisika
Fakultas	: Fakultas Teknologi Industri & Rekayasa Sistem (FT-IRS)
Perguruan Tinggi	: Institut Teknologi Sepuluh Nopember

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "***SISTEM DETEKSI PLAT NOMOR OTOMATIS MENGGUNAKAN ALGORITMA YOLO V3 DENGAN FRAMEWORK DARKNET*** "adalah benar karya saya sendiri dan bukan plagiat dari karya orang lain. Apabila di kemudian hari terbukti terdapat plagiat pada Tugas Akhir ini, maka saya bersedia menerima sanksi sesuai ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sebenarnya-benarnya.

Surabaya, 3 Agustus 2020

Yang membuat pernyataan,

Farah Qoonita Syuhaila

NRP. 02311640000120

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

TUGAS AKHIR

SISTEM DETEKSI PLAT NOMOR OTOMATIS MENGGUNAKAN ALGORITMA YOLO V3 DENGAN *FRAMEWORK DARKNET*

Oleh:

Farah Qoonita Syuhaila

NRP. 02311640000120

Surabaya, 26 Agustus 2020

Menyetujui, Pembimbing I

Surabaya: 27/08/2020

Dr. rer. nat. Ir. Aulia M.T Nasution M.Sc.

NIP. 196711171997021001

Mengetahui,

Kepala Departemen

Teknik Fisika FTI RS ITS

Dr. Suyanto, S.T., M.T.

NIP. 19711113199512100

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

SISTEM DETEKSI PLAT NOMOR OTOMATIS MENGGUNAKAN ALGORITMA YOLO V3 DENGAN *FRAMEWORK DARKNET*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Progam Studi S-1 Departemen Teknik Fisika
Fakultas Teknologi Industri & Rekayasa Sistem (FT-IRS)
Institut Teknologi Sepuluh Nopember

Oleh:

Farah Qoonita Syuhaila

NRP. 02311640000120

Disetujui oleh Tim Penguji Tugas Akhir:

1. Dr.rer.nat. Ir. Aulia M.T. Nasution, M.Sc. ……27/08/2020…… (Pembimbing I)
2. Agus Muhamad Hatta, S.T., M.Si., Ph.D. …… (Ketua Penguji)
3. Dr.-Ing Doty Dewi Risanti, S.T., M.T. …… (Penguji I)
4. Detak Yan Pratama, S.T., M.Sc. …… (Penguji II)
5. Iwan Cony Setiadi, S.T., M.T. …… (Penguji III)

SURABAYA

2020

Halaman ini sengaja dikosongkan

SISTEM DETEKSI PLAT NOMOR OTOMATIS MENGGUNAKAN ALGORITMA YOLO V3 DENGAN *FRAMEWORK DARKNET*

Nama	**:**	**Farah Qoonita Syuhaila**
NRP	**:**	**02311640000120**
Departemen	**:**	**Teknik Fisika FT-IRS ITS**
Dosen Pembimbing	**:**	**Dr.rer.nat. Ir Aulia M.T. Nasution, M.Sc.**

ABSTRAK

Otomatisasi sistem dalam pengawasan transportasi memiliki peran penting untuk pendeteksian pelanggaran yang terjadi di jalan raya. Pengawasan kendaraan yang melakukan pelanggaran salah satu diantranya bisa dilakukan dengan cara pengidentifikasian plat nomor dari kendaraan dengan standar SNI. Tahapan dalam pendeteksian plat nomor kendaraan, dimulai dari lokalisasi plat nomor kendaraan lalu dilanjutkan dengan pembacaan karakter pada plat nomor. Pada penelitian ini dikembangkan suatu sistem pendeteksian plat nomor kendaraan secara otomatis menggunakan Algoritma YOLO V3 dengan framework *darknet*. Langkah awal yang dilakukan dalam pembuatan sistem deteksi adalah pendeteksian objek menggunakan Algoritma YOLO V3 dan dilanjutkan dengan pendeteksian karakter menggunakan *Tesseract* OCR. Hasil pengujian menunjukan bahwa sistem pendeteksian plat nomor otomatis mencapai tingkat akurasi 98,52% dengan rata-rata pembacaan pada 3,03 detik dan untuk hasil OCR menggunakan *Software Tesseract* di dapatkan hasil deteksi 20% dimana sistem berhasil untuk mengenali seluruh karakter pada plat mobil yang berupa karakter Alphanumerik sebanyak 6-7 karakter.

***Kata Kunci: CNN*, *Deep learning*, *Object detection*, *Tesseract*, YOLO**

Halaman ini sengaja dikosongkan

AUTOMATIC LICENSE PLATE DETECTION SYSTEM USING YOLO *V3 ALGORITHM WITH DARKNET FRAMEWORK*

Name	**:**	**Farah Qoonita Syuhaila**
NRP	**:**	**02311640000120**
Department	**:**	**Engineering Physics FT-IRS ITS**
Supervisors	**:**	**Dr. rer.nat. Ir. Aulia M.T. Nasution, M.Sc.**

ABSTRACT

Automation of the system in transportation surveillance has an important role for the detection of violations that occur on the highway. Surveillance of vehicles that violate one of them can be done by identifying the license plates of vehicles with SNI standards. Stages in the detection of vehicle license plates, starting from localization of vehicle license plates and then followed by reading the characters on the license plate. In this study a vehicle license plate detection system was developed automatically using the YOLO *V3 algorithm with the darknet framework. The first step taken in making the detection system is object detection using the* YOLO *V3 algorithm and continued with character detection using the Tesseract OCR. The test results show that the automatic number plate detection system reaches an accuracy level of 98.52% with an average reading of 3.03 secon and for the OCR results using the Tesseract Software, a 20% detection result was obtained where the system succeeded in recognizing all characters on the car plate in the form of 6-7 alphanumeric character.*

***Keywords: CNN*, *Deep learning*, *Object detection*, *Tesseract*, YOLO**

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Segala puji dan syukur kehadirat Allah SWT atas berkah, rahmat dan hidayah-Nya yang senantiasa dilimpahkan kepada penulis, sehingga bisa menyelasaikan tugas akhir dengan judul "*Sistem Deteksi Plat Nomor Otomatis Menggunakan Algoritma* YOLO *V3 dengan Framework Darknet*" sebagai syarat untuk menyelesaikan Program Sarjana (S1) pada Program Sarjana Fakultas Teknologi Industri Departemen Teknik Fisika, Institut Teknologi Sepuluh Nopember. Dalam penyusunan tugas akhir ini banyak hambatan serta rintangan yang penulis hadapi namun pada akhirnya dapat melaluinya berkat adanya bimbingan dan bantuan dari berbagai pihak baik secara moral maupun spiritual. Untuk itu pada kesempatan ini penulis menyampaikan ucapan terimakasih kepada:

1. Dr. Suyanto, S.T., M.T. Selaku Kepala Departemen Teknik Fisika Institut Teknologi Sepuluh Nopember.
2. Dr.rer.nat. Ir Aulia M.T. Nasution, M.Sc. Selaku Dosen Pembimbing yang telah bersedia meluangkan waktu untuk memberikan arahan selama penyusunan skripsi.
3. Dr.-Ing. Doty Dewi Risanti, S.T., M.T. dan Andi Rahmadiansah, S.T., M.T. selaku dosen wali yang telah memberikan dukungan pengarahan selama masa perkuliahan.
4. Seluruh jajaran Dosen dan Staff Departemen Teknik Fisika Institut Teknologi Sepuluh Nopember.
5. Kedua Orang tua saya, Ir. Rohyani Gofar, M.T. dan Dr. Heppy Agustiana Vidyastuti yang telah memberikan doa dan dukungan selama proses pembuatan Tugas Akhir.
6. Rida Ayu Arfianti, sebagai seorang sosok yang lebih dari Adik Kelas tetapi sebagai Saudara yang saya sangat sayangi. Yang mulai membantu kelancaran saya dalam berkuliah, semoga semua kebaikan berbalas kepadamu.
7. Mas Wahyu, Mas Kaka, Mas Wira sebagai kakak tingkat yang telah membantu jalannnya tugas akhir ini.
8. Sahabat Krucil 2017 ; Caca, Toplong, Bela, Jovan yang telah menemani pahit manisnya perkuliahan di Teknik Fisika ITS. Terus semangat di tahun terakhirnya

ya nanti, ingat skripsi yang bagus adalah skripsi yang selesai! Dan jangan lupa ukur kapabilitas kemampuan dengan kondisi ya!

9. Ghiffar Abdillah, Vania Huwaida Putri, Hariz Mumtaz Nurfuad, Fathima Noor Alia sebagai adik kelas dari NF yang sangat saya sayangi, terimakasih untuk terus menyemangati saya agar tidak menyerah.
10. Kayi Mahdi Y, Maulana, Selfi Stendafity, Hafiz Salam, Nadhifa Aqillah Husnaa, Murti Marinah, Zyrlirosa, Panjul dan kawan kawan Satu Laboratorium Rekayasa Fotonika yang juga telah mewarnai kehidupan ketika kuliah.

Dan juga terimakasih untuk semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu memberikan dukungan. Penulis mohon maaf atas segala kesalahan yang pernah dilakukan. Semoga Tugas Akhir ini dapat memberikan manfaat untuk mendorong penelitian-penelitian selanjutnya.

Bandung, Juni 2020

Farah Qoonita Syuhaila

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Otomatisasi sistem dalam pengawasan lalu lintas transportasi terhadap pelanggaran dalam berkendara atau pelanggaran pada lampu lalu lintas membutuhkan suatu sistem untuk mengenali plat nomor secara otomatis, dimana sebuah kamera CCTV (*Closed-Circuit Television*) memiliki peran penting untuk memindai jenis pelanggaran yang terjadi. Penelitian penentuan lokasi pemindaian untuk kota kecil dan jalanan kota yang yang tidak terlalu ramai telah dilakukan sebelumnya oleh (Qiu, 2010) dengan menggunakan algoritma *Graphic Minimal Covering*. Namun, untuk memindai pelanggaran lalu lintas di kota besar tentu saja akan dibutuhkan lebih banyak kamera CCTV karena jumlah kendaraan dan jumlah jalan yang lebih kompleks.

Pemindaian kendaraan yang melakukan pelanggaran lalu lintas dilakukan dengan cara mengidentifikasi plat nomor dari kendaraan tersebut. Pembuatan suatu sistem pengenalan plat nomor otomatis membantu mengefisiensikan penggunaan sumber daya manusia untuk mengamati dan merekam setiap plat nomor kendaraan yang terdeteksi melakukan suatu pelanggaran. Langkah penting dari pendeteksian ini adalah melakukan lokalisasi plat nomor kendaraan, kemudian pembacaan karakter pada plat nomor kendaraan dilakukan dengan metode *Optical character recognition* (OCR). Terdapat banyak metode pendeteksian obyek yang telah dikembangkan (Shreyas, 2017). Pada penelitian yang di lakukan oleh Shreyas dkk dibuatlah suatu sistem deteksi plat motor dengan mengirimkan SMS pada ponsel GSM dengan tingkat ketercapaian hingga 80% dan memiliki kekurangan pada pendeteksian karakter pada 10 alphanumerik pada plat nomor.

Deep learning merupakan sub divisi dari *Machine learning* yang memungkinkan komputer untuk belajar dari sistem terdahulu dan pemahaman yang mendalam terkait suatu hal yang bersifat konseptual dan hierarki. Pembelajaran *machine learning* diperoleh dari pembelajaran berkelanjutan dari suatu sistem yang dipelajari, hingga komputer dapat menemukan polanya tersendiri hal ini memungkinkan komputer dapat melakukan suatu pembelajaran tersendiri kedepannya, tanpa bantuan operator manusia. Hierarki konsep pembelajaran berkelanjutan yang dipelajari komputer

memungkinkan komputer untuk mempelajari konsep yang rumit dan mensimplifikasi hal tersebut (Kwanggi, 2016). Seperti pada *deep learning*, pada *deep learning* terdiri dari jumlah lapisan pemrosesan yang lebih tinggi atau lebih dalam, yang memiliki perbedaan dengan model pembelajaran dengan jumlah *layer* yang sedikit. Pergeseran dari pembelajaran yang memiliki jumlah *layer* yang sedikit menuju *deep learning* telah memungkinkan fungsi yang lebih kompleks dan non-linier untuk dipetakan, hal ini dikarenakan komputasi pada *deep learning* tidak dapat dipetakan secara efisien dengan arsitektur penyusun yang sangat sederhana.

Saat ini, ada dua metode *object detection* yang digunakan dalam *deep learning* : yang pertama adalah adalah algoritma pendeteksian target yang menggabungkan *Convolutional Neural Network* dan *candidate region*, diwakili oleh CNN berbasis wilayah yaitu (R-CNN) *Recurrent–Convolutional Neural Network* (Girshick, 2014) dan pengumpulan wilayah spasial piramida (SPP)-net (He, 2015). Lalu pengembangan menggunakan *Single Shot MultiBox Detector* (SSD) (Thakar, 2018) dan model *You Only Look Once* (YOLO) (Redmon J. D., 2016). Hingga saat ini pengembangan YOLO sudah mencapai YOLO V3. Dibandingkan dengan metode lainnya, algoritma YOLO merupakan metode yang sudah lebih sederhana daripada metode lainnya (Redmon J. D., 2016). YOLO diklaim mampu mendeteksi citra lebih cepat dan mempunyai presisi rata-rata dua kali lebih baik daripada sistem *real-time* yang lain karena tidak perlu membagi *region* dalam suatu citra untuk melakukan identifikasi.

YOLO melihat keseluruhan citra selama proses pelatihan dan pengujian, dengan demikian YOLO membuat kurang dari setengah jumlah kesalahan pada latar gambar. YOLO mempelajari representasi objek yang dapat digeneralisasikan. Ketika dilatih dengan gambar alami dan diuji pada karya seni, YOLO mengungguli metode deteksi seperti *Deformable Part Models* (DPM) dan R-CNN (*Recurrent-Convolutional Neural network*) dengan margin pendeteksian yang lebar. YOLO masih mempunyai kelemahan, yaitu dalam hal akurasi. Meskipun YOLO mampu mengidentifikasi obyek dengan cepat, namun YOLO masih kesulitan dalam melokalisasi obyek, terutama obyek yang berukuran kecil.

Pada penelitian yang dilakukan oleh Shen dkk dilakukan pendeteksian plat nomor dengan Algoritma YOLO V3 termodifikasi. Modifikasi dengan penambahan

convolutional layer ini mampu meningkatkan presisi pendeteksian plat nomor kendaraan hingga 97,77%. Namun demikian, metode ini masih kurang mampu mendeteksi untuk kondisi citra kabur dan terlalu terang (Shen, 2020). YOLO V3 juga terbukti mampu mendeteksi plat nomor kendaraan dengan akurasi deteksi sebesar 99,31% (Sun, 2019). Sebagaimana disebutkan di atas bahwa YOLO mempunyai kelemahan dalam mendeteksi obyek berukuran kecil, modifikasi YOLO dengan menggunakan *sliding window* telah dilakukan (Hendry, 2019). Metode ini mampu mencapai 98,22% akurasi untuk mendeteksi plat nomor kendaraan dan 78% akurasi untuk pengenalan plat nomor kendaraan.

Penelitian lain yang dilakukan oleh Jamstho dkk menggunakan YOLO V2 *Darknet* mampu menghasilkan presisi rata-rata sebesar 98,6% dalam pembacaan plat nomor kendaraan (Jamstho, 2020). *Darknet* sendiri adalah sebuah *framework neural network* yang bersifat *open source* digunakan membangun, melatih dan menjalankan *neural network* (Alexey A. , 2016). Perkembangan pengenalan gambar telah mencapai melalui tiga tahap: pengenalan karakter, pemrosesan gambar digital dan pengenalan objek. OCR adalah satu pengenalan karakter, yang menggunakan metode optik untuk mengubah teks dalam kertas menjadi gambar matriks dan mengubah gambar ke dalam teks oleh perangkat lunak untuk pengeditan dan pemrosesan lebih lanjut. *Tesseract* merupakan suatu mesin OCR *open source,* yang mendukung pengenalan ribuan karakter bahasa (Sun P. Q., 2019). Berdasarkan penelitian yang telah dilakukan sebelumnya, maka akan dilakukan perancangan sistem deteksi plat nomor otomatis menggunakan algoritma YOLO V3 dengan *framework Darknet*. Dengan adanya metode pendeteksian obyek menggunakan YOLO ini diharapkan dapat mengenali plat nomor yang berada di Indonesia.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, maka rumusan masalah dalam penelitian tugas akhir ini adalah sebagai berikut:

a) Bagaimana menerapkan pengenalan plat nomor standar SNI secara otomatis menggunakan Algoritma YOLO V3 dan keterkenalan huruf menggunakan *Tesseract*-OCR?

b) Bagaimana cara menguji tingkat akurasi pengenalan plat nomor dan karakter pada plat nomor dengan metode yang digunakan?

1.3 Tujuan

Adapun tujuan yang ingin dicapai pada penelitian tugas akhir ini yaitu:

a) Menerapkan pengenalan plat nomor standar SNI secara otomatis menggunakan Algoritma YOLO V3 dan keterkenalan huruf menggunakan *Tesseract*-OCR.
b) Menguji tingkat akurasi pengenalan plat nomor dan karakter pada plat nomor dengan metode yang digunakan.

1.4 Batasan Masalah

Bersadarkan identifikasi masalah maka batasan masalah pada penelitian tugas akhir ini adalah sebagai berikut:

a) Ukuran plat nomor yang dilokalisasi adalah ukuran plat nomor kendaraan bermotor yang telah ditetapkan oleh kepolisian Indonesia dengan ukuran 395 x 135 mm.
b) Citra yang digunakan hanya citra kendaraan mobil dengan plat nomor Standar Negara Indonesia (SNI).
c) Citra kendaraan bermotor yang ditangkap dalam keadaan diam
d) Atribut yang digunakan dalam deteksi Tanda Nomor Kendaraan Bermotor pada media *streaming* dikhususkan untuk citra maupun video yang pada kendaraan bermotor.
e) Data gambar yang digunakan adalah plat nomor kendaraan bermotor kendaraan roda empat atau roda dua.
f) Algoritma yang digunakan adalah YOLO V3.
g) Alat bantu yang digunakan adalah Python dengan *framework Darknet.*
h) Pengambilan citra data dilakukan pada rentang waktu 09.00-15.00 WIB.
i) Karakter yang dikenali adalah angka (0 sampai 9) dan huruf alphabet kapital (A sampai Z).
j) *Software* yang digunakan untuk mengenali karakter pada plat nomor kendaraan bermotor adalah *Tesseract* 4.11

1.5 Sistematika Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut :

a) BAB I PENDAHULUAN

Pada BAB I ini terdiri dari latar belakang, perumusan masalah, tujuan, lingkup kerja dan sistematika laporan.

b) BAB II TINJAUAN PUSTAKA

Pada BAB II ini dibahas mengenai teori-teori yang berkaitan dengan penelitian yang akan dilakukan, yaitu Kendaraan Bermotor, Tanda Nomor Kendaraan Bermotor, *Computer Vision, Image Processing, Object Detection, Neural Network, Convolutional Neural Network, Deep Learning,* Algoritma YOLO, Python, *Optical Character Recognition* dan Akurasi.

c) BAB III METODOLOGI PENELITIAN

Pada BAB III ini berisi mengenai rancangan dari penelitian yang dilakukan, metode dan langkah-langkah dalam penelitian.

d) BAB IV ANALISA DATA DAN PEMBAHASAN

Pada BAB IV analisis hasil penelitian dari proses deteksi dan identifikasi Tanda Nomor Kendaraan Bermotor sesuai standar SNI yang berupa hasil akurasi dari pemindaian sistem untuk keterkenalan plat nomor dan Alphanumerik plat nomor.

e) BAB V KESIMPULAN DAN SARAN

Pada bab V ini diberikan kesimpulan tentang tugas akhir yang telah dilakukan berdasarkan hasil yang diperoleh, serta diberikan saran sebagai penunjang maupun pengembangan tugas akhir selanjutnya.

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Kendaraan Bermotor

Tahun 2012 Tanda Nomor Kendaraan Bermotor yang selanjutnya disingkat TNKB adalah tanda registrasi dan identifikasi kendaraan bermotor yang berfungsi sebagai bukti legitimasi pengoperasian kendaraan bermotor berupa pelat atau berbahan lain dengan spesifikasi tertentu yang diterbitkan oleh Polisi Republik Indonesia dan berisikan kode wilayah, nomor registrasi, serta masa berlaku dan dipasang pada kendaraan bermotor. Menurut Undang-Undang Republik Indonesia Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan Pasal 68 pada ayat (1) : Setiap Kendaraan Bermotor yang dioperasikan di Jalan wajib dilengkapi dengan Surat Tanda Nomor Kendaraan Bermotor dan Tanda Nomor Kendaraan Bermotor. Pada ayat (2) : Surat Tanda Nomor Kendaraan Bermotor sebagaimana dimaksud pada ayat (1) memuat data Kendaraan Bermotor, identitas pemilik, nomor registrasi Kendaraan Bermotor, dan masa berlaku. Pada ayat (3) : Tanda Nomor Kendaraan Bermotor sebagaimana dimaksud pada ayat (1) memuat kode wilayah, nomor registrasi, dan masa berlaku. Pada Ayat (4) : Tanda Nomor Kendaraan Bermotor harus memenuhi syarat bentuk, ukuran, bahan, warna, dan cara pemasangan

Tanda Nomor Kendaraan Bermotor (TNKB), atau seringkali disebut plat nomor adalah plat aluminium tanda kendaraan bermotor di Indonesia yang telah didaftarkan pada Samsat (Sistem Administrasi Manunggal Satu Atap) setempat. (Undang Undang Republik Indonesia, 2009)

2.2 Spesifikasi Teknis

Tanda Nomor Kendaraan Bermotor berbentuk plat aluminium dengan cetakan tulisan dua baris. Baris pertama menunjukkan: kode wilayah (huruf), nomor polisi (angka), dan kode atau seri akhir wilayah (huruf) dan baris kedua menunjukkan bulan dan tahun masa berlaku. Bahan baku TNKB adalah aluminium dengan ketebalan 1 mm. Ukuran TNKB untuk kendaraan bermotor roda 2 dan roda 3 adalah 250x105 mm, sedangkan untuk kendaraan bermotor roda 4 atau lebih adalah 395x135 mm.Terdapat cetakan garis lurus pembatas lebar 5 mm diantara ruang

nomor polisi dengan ruang angka masa berlaku. Pada sudut kanan atas dan sudut kiri bawah terdapat tanda khusus (*security mark)* cetakan lambang Polisi.

Lalu Lintas; sedangkan pada sisi sebelah kanan dan sisi sebelah kiri ada tanda khusus cetakan "DITLANTAS POLRI" yang merupakan hak paten pembuatan TNKB oleh Polri (Undang Undang Republik Indonesia, 2009)

2.2.1 Plat Nomor Kendaraan Bermotor di Indonesia

Tanda Nomor Kendaraan Bermotor (TNKB), atau seringkali disebut plat nomor, adalah plat aluminium tanda kendaraan bermotor di Indonesia yang telah didaftarkan pada Samsat setempat.

Gambar 2.1 Jenis Plat Nomor Kendaraan Bermotor di Indonesia (Lukas, 2013)

2.2.2 Spesifikasi Teknis Baru

Korps Lantas Mabes Polri terhitung mulai April 2011 mangganti desain plat nomor kendaraan. Ukurannya lebih panjang 5 sentimeter daripada plat nomor sebelumnya. Perubahan ukuran plat dilakukan karena ada penambahan menenjadi tiga huruf di belakang nomor (Contoh B 2634 **SFJ**), sementara sebelumnya hanya dua huruf (Contoh B 1090 **CA)**. Perubahan ini membuat angka dan huruf pada plat nomor berdesakan, sehingga sulit dibaca. Dengan diperpanjangnya plat tersebut, jarak antara nomor dan huruf pada plat lebih luas sehingga mudah terbaca. ukuran TNKB untuk kendaraan roda 2 dan 3 sekarang menjadi 275 mm dengan lebar 110 mm, sedangkan untuk kendaraan roda 4 atan lebih adalah panjang 430 mm dengan lebar 135 mm. Sementara ini, plat resmi yang lama masih berlaku. Selain itu pada

spesifikasi teknis baru ini plat nomor menggunakan rupa huruf (*font)* yang sama (Peraturan Kepala Kepolisian Republik Indonesia, 2012)

2.2.3 Warna Tanda Nomor Kendaraan Bermotor

Warna pada Tanda Nomor Kendaraan Bermotor ditetapkan sebagai berikut:

1) Kendaraan bermotor bukan umum dan kendaraan bermotor sewa: Warna dasar hitam dengan tulisan berwarna putih
2) Kendaraan bermotor umum : Warna dasar kuning dengan tulisan berwarna hitam
3) Kendaraan bermotor milik Pemerintah : Warna dasar merah dengan tulisan berwarna Putih
4) Kendaraan bemotor Korps Diplomatik Negara Asing : Warna dasar merah dengan tulisan berwarna hitam

2.3 *Computer Vision*

Pada hakikatnya*, computer vision* meniru cara kerja sistem visual manusia. Dalam proses penglihatan manusia*,* manusia melihat objek dengan menggunakan indera penglihatan yang berupa mata*,* lalu citra objek diteruskan ke otak untuk diinterpretasikan sehingga manusia mengerti objek yang tampak. Hasil interpretasi ini kemudian digunakan untuk pengambilan keputusan.

Tabel 2.1 Garis besar *human vision* dan *computer vision*

Human Vision	Computer Vision
Menggunakan mata dan visual cortex di dalam otak.	Menggunakan kamera-kamera yang terhubung pada sistem komputer.
Menemukan dari gambar objek apa yang ada dalam penglihatan, dimana posisinya, bagaimana bergerak, dana apa bentuknya	Secara otomatis menginterpretasi gambar-gambar dan mencoba untuk mengerti isinya seperti pada human vision.

Secara garis besar, *computer vision* adalah sebuah teknologi mesin yang mampu mengenali objek yang diamati. Kemampuan untuk mengenali ini merupakan kombinasi dari pengolahan citra dan pengenalan pola. Baik atau lebih mudah diinterpretasikan, sedangkan pengenalan pola adalah proses identifikasi objek pada citra. Proses-proses dalam *computer vision* secara garis besar dapat dibagi menjadi (Febriandita, 2016):

1. Proses mengakuisisi citra digital (*Image Acquisition*)
2. Proses pengolahan citra (*Image processing*)
3. Proses analisis data citra (*Image Analysis*)
4. Proses pemahaman data citra (*Image Understanding*)

Computer vision merupakan kombinasi antara *image processing* dan *pattern recognition*. *Computer vision* adalah pembangunan deskripsi objek fisik yang eksplisit dan gamblang dari sebuah gambar. *Output* dari *computer vision* adalah deskripsi atau interpretasi atau beberapa pengukuran kuantitatif struktur dalam adegan 3D (Febriandita, 2016)

2.4 *Image Processing*

Menurut Murni (Murni, 2016) pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Proses ini mempunyai ciri data masukan dan informasi keluaran berbentuk citra. Teknik pengolahan citra menggunakan komputer untuk mendigitasi pola bayangan dan warna dari gambar yang sudah tersedia. Informasi yang telah terdijitasi ini kemudian dipindah ke layar dari monitor video. Pengolahan citra banyak digunakan dalam dunia fotografi misalnya mengubah intensitas cahaya sebuah foto, dunia perfilman misalnya animasi, dunia kedokteran misalnya untuk membuat analisa medis, dan dunia game

2.5 *Object Detection*

Object detection atau deteksi objek merupakan keilmuan yang menggabungkan antara *image processing* dan *pattern recognition*. *Object detection* merupakan suatu keilmuan untuk menentukan keberadaan suatu objek dan atau ruang lingkupnya, dan lokasi pada gambar. Hal ini dapat diberlakukan sebagai pengenalan objek kelas dua, dimana satu kelas mewakili kelas objek dan kelas lainnya mewakili kelas non-objek.

Deteksi objek dapat dibagi lagi menjadi *soft detection* dan *hard detection*. *Soft detection* hanya mendeteksi adanya objek sedangkan *hard detection* mendeteksi keberadaan dan lokasi objek. Bidang deteksi objek biasanya dilakukan dengan mencari setiap bagian gambar dan melokalisasi bagian tertentu, yang sifat fotometrik atau geometriknya sesuai dengan objek target dalam *database* pelatihan. Hal ini dapat dilakukan dengan memindai *template* objek di gambar di lokasi, skala, dan rotasi yang berbeda, dan deteksi dideklarasikan jika kemiripan antara *template* dan gambar cukup tinggi. Kesamaan antara *template* dan area gambar dapat diukur dengan korelasi. Selama beberapa tahun terakhir telah ditunjukkan bahwa detektor objek berbasis gambar sensitif terhadap data pelatihan (Jalled., 2016)

2.6 Data *Pre-Processing*

Data *Pre-processing* mengacu pada serangkaian kegiatan yang dilakukan untuk membuat data mentah diproses lebih lanjut. Pengolahan data ini dibagi menjadi beberapa tahap yaitu penambangan data (*Data mining*). Proses ini digunakan untuk mengumpulkan informasi terkait data yang akan digunakan dalam penelitian. Pembersihan data *(Data cleaning*) proses ini merupakan proses penyaringan data. Proses penyaringan data ini diklasifikasi berdasarkan data yang sesuai dengan kebutuhan penelitian ataupun tidak (Xiang-wei, 2012).

2.6.1 Teknik *Pre-Processing Data*

2.6.1.1 *Image* Scaling

Dalam grafik komputer, penskalaan gambar adalah proses mengubah ukuran gambar digital. Dalam kasus perbesaran gambar metode yang digunakan terlebih dahulu dengan mengubah ukuran gambar *padding* dengan nol, diikuti oleh konvolusi dengan *filter* linear dua dimensi, invarian spasial. Dalam proses *scaling* akan berhadapan dengan matriks yang mewakili diskrit nilai piksel, oleh karena itu diperlukan untuk menggunakan *filter* dalam domain diskrit. Karenanya setiap *filter* dapat diwakili oleh fungsi impuls h, yang disebut *filter*. Untuk pengecilan gambar dilakukan metode seperti peningkatan frekuensi tinggi pada gambar yang akan menyebabkan beberapa piksel untuk dipecah menjadi satu. Oleh karena itu diperlukan untuk penerapan *filter* pemerataan (*smoothing filter*) untuk mereduksi permasalahan pada target gambar. (Prasantha, 2013)

2.7 *Neural Network*

Jaringan saraf tiruan adalah sistem pemrosesan informasi yang memiliki karakteristik kinerja tertentu yang sama dengan jaringan saraf biologis. Jaringan saraf tiruan telah dikembangkan sebagai generalisasi model matematika dari pemahaman manusia atau saraf biologi, berdasarkan asumsi bahwa :

1. Pengolahan informasi terjadi pada banyak elemen sederhana yang disebut *neuron*.
2. Sinyal dilewatkan antara *neuron* melalui jalur yang terhubung.
3. Setiap jalur yang berhubungan memiliki bobot.
4. Setiap *neuron* berlaku fungsi aktivasi (biasanya non linier) untuk menentukan sinyal *output*.

Neural network telah diaplikasikan di berbagai bidang. Hal ini dikarenakan *Neural network* memiliki kelebihan-kelebihan (Yahya, 2016):

1. Dapat memecahkan problema non-linear yang umum dijumpai,
2. Kemampuan memberikan jawaban terhadap *pattern* yang belum pernah dipelajari *(generalization)*
3. Dapat secara otomatis mempelajari data numerik yang diajarkan pada jaringan tersebut.

Neural network mampu melakukan berbagai hal berikut (Yahya, 2016):

1. Klasifikasi : memilih suatu *input* data ke dalam kategori tertentu yang sudah ditetapkan
2. Asosiasi : menggambarkan suatu obyek secara keseluruhan hanya dengan bagian dari obyek lain,
3. *Self organizing* : kemampuan mengolah data-data *input* tanpa harus mempunyai target
4. Optimasi : menemukan suatu jawaban terbaik sehingga mampu untuk meminimalisasi biaya.

2.8 *Convolutional Neural Network*

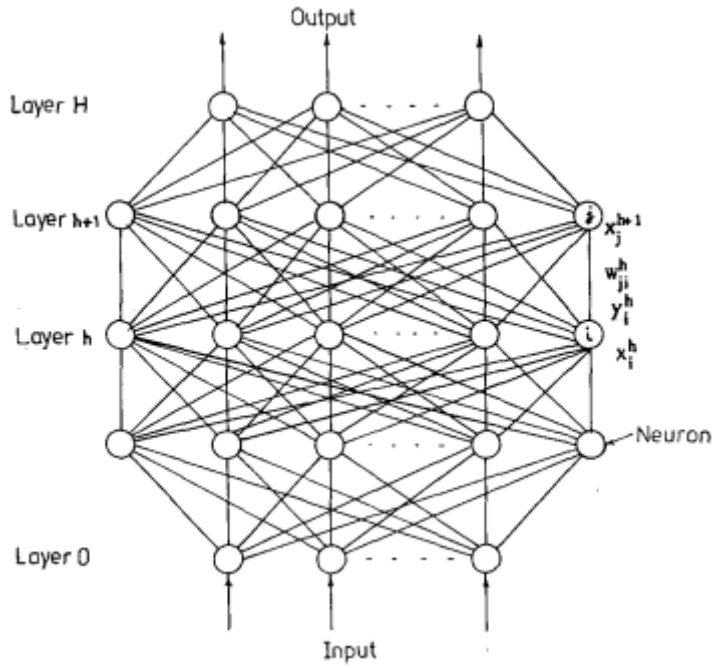
Convolutional neural network (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk

dalam jenis *Deep Neural network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik. CNN pertama kali dikembangkan dengan nama NeoCognitron oleh Kunihiko Fukushima (Fukushima, 1980), seorang peneliti dari *NHK Broadcasting Science Research Laboratories*, Kinuta, Setagaya, Tokyo, Jepang 4 CEK Konsep tersebut kemudian dimatangkan oleh Yann LeCunn, seorang peneliti dari AT&T *Bell Laboratories* di Holmdel, New Jersey, USA.

Model CNN dengan nama LeNet berhasil diterapkan oleh LeCunn pada penelitiannya mengenai pengenalan angka dan tulisan tangan (LeCunn, 1990) Pada tahun 2012, (Krizhevsky, 2012) dengan penerapan CNN miliknya berhasil menjuarai kompetisi *ImageNet Large Scale Visual Recognition Challenge* 2012. Dengan membawa nilai *error* dalam pengenalan hingga 15,3% (Du, 2018). Kemudian, ZF-net, GoogLeNet dan VGGNet membawa tingkat kesalahan (*error*) yang lebih rendah dibandingkan sebelumnya. Pada akhir tahun 2015, ResNet telah memperoleh tingkat kesalahan untuk pengembangan CNN sebesar 3,6% yang bahkan lebih rendah dari mata manusia 5,1%. Prestasi tersebut menjadi momen pembuktian bahwa metode *deep learning*, khususnya CNN. Metode CNN terbukti berhasil mengungguli metode *machine learning* lainnya seperti *Support Vector Machine* pada kasus klasifikasi objek citra dan juga mampu mengungguli kemampuan manusia

2.8.1 Konsep CNN

Cara kerja CNN memiliki kesamaan pada MLP, namun dalam CNN setiap *neuron* dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap *neuron* hanya berukuran satu dimensi.



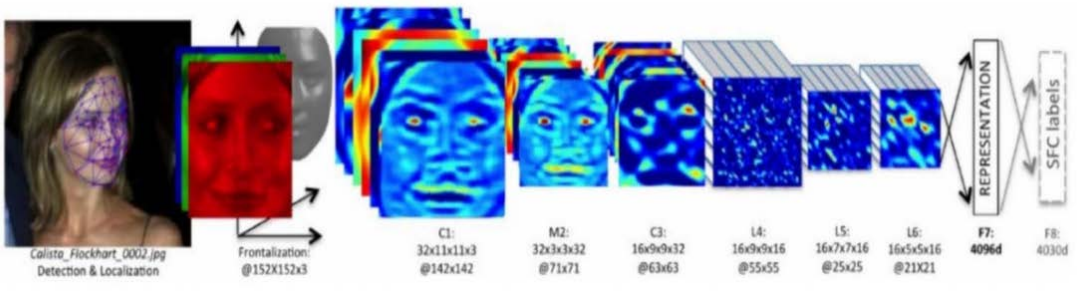
Gambar 2.2 Arsitektur MLP Sederhana (Pal, 1992)

Sebuah MLP seperti pada Gambar 2.2 memiliki *layer* 0, h, h+1,dan H dengan masing-masing *layer* berisi *neuron* (lingkaran putih). MLP menerima *input* data satu dimensi dan mempropagasikan data tersebut pada jaringan hingga menghasilkan *output*. Setiap hubungan antar *neuron* pada dua *layer* yang bersebelahan memiliki parameter bobot satu dimensi yang menentukan kualitas mode. Di setiap data *input* pada *layer* dilakukan operasi linear dengan nilai bobot yang ada, kemudian hasil komputasi akan ditransformasi menggunakan operasi non linear yang disebut sebagai *activation function*. Pada CNN, data yang dipropagasikan pada jaringan adalah data dua dimensi, sehingga operasi linear dan parameter bobot pada CNN berbeda. Pada CNN operasi linear menggunakan operasi konvolusi, sedangkan bobot tidak lagi satu dimensi saja, namun berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi seperti pada Gambar.2.2. Dimensi bobot pada CNN adalah:

$$neuron\ input\ x\ neuron\ output\ x\ tinggi\ x\ lebar \tag{2.1}$$

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

Pada Gambar 2.3 dijelaskan terkait proses yang terjadi pada gambar pada proses operasi konvolusi.



Gambar 2.3. Proses Konvolusi pada CNN (Albawi, Mohammed, & Al-Zawi, 2017)

2.8.2 Arsitektur CNN

Artificial neural network terdiri dari berbagai *layer* dan beberapa *neuron* pada masing-masing *layer*. Kedua hal tersebut tidak dapat ditentukan menggunakan aturan yang pasti dan berlaku berbeda-beda pada data yang berbeda (Stathakis, 2008). Pada kasus MLP, sebuah jaringan tanpa *hidden layer* dapat memetakan persamaan linear apapun, sedangkan jaringan dengan satu atau dua *hidden layer* dapat memetakan sebagian besar persamaan pada data sederhana. Namun pada data yang lebih kompleks, MLP memiliki keterbatasan.

Pada permasalahan jumlah *hidden layer* dibawah tiga *layer*, terdapat pendekatan untuk menentukan jumlah *neuron* pada masing-masing *layer* untuk mendekati hasil optimal. Penggunaan *layer* diatas dua pada umumnya tidak direkomendasikan dikarenakan akan menyebabkan *overfitting* serta kekuatan *backpropagation* berkurang secara signifikan.

Dengan berkembangnya *deep learning*, ditemukan bahwa untuk mengatasi kekurangan MLP dalam menangani data kompleks, diperlukan fungsi untuk mentransformasi data *input* menjadi bentuk yang lebih mudah dimengerti oleh MLP.

Hal tersebut memicu berkembangnya *deep learning* dimana dalam satu model diberi beberapa *layer* untuk melakukan transformasi data sebelum data diolah menggunakan metode klasifikasi. Hal tersebut memicu berkembangnya model *neural network* dengan jumlah *layer* diatas tiga. Namun dikarenakan fungsi *layer* awal sebagai metode *Feature extraction*, maka jumlah *layer* dalam sebuah *deep neural*

network tidak memiliki aturan universal dan berlaku berbeda-beda tergantung *dataset* yang digunakan. Dikarenakan hal tersebut, jumlah *layer* pada jaringan serta jumlah *neuron* pada masing-masing *layer* dianggap sebagai *hyperparameters* dan dioptimasi menggunakan pendekatan *searching.*

2.8.3 Lapisan pada CNN

Kerangka lapisan jaringan dari *neural network* ke CNN kurang lebih berfungsi sama yang membedakan hanya fungsi dan pembentukan jaringan. Dalam CNN terdapat beberapa lapisan, dimana tiap lapisan memiliki suatu fungsi tersendiri, lapisan CNN ditunjukkan sebagai berikut :

1. *Input Layer*

Tugas utama adalah menginisialisasi data gambar *input* untuk membuat semua dimensi data *i* berpusat nol. Kemudian, itu menormalkan skala semua *input* data di dalamnya [0*,* 1] untuk mempercepat kecepatan konvergen, dan memutihkan data untuk mengurangi redundansi. Itu memanfaatkan *Principal Components Analysis* (PCA) untuk menurunkan dan mengkorelasi informasi terkait dimensi data yang akan dibuat. Pada mereka fokus hanya pada beberapa faktor dan tidak menyeluruh (Krizhevsky, 2012)

2. *Convolutional Layer*

Lapisan konvolusional merupakan inti dari CNN. Lapisan konvolusional menggunakan kernel CONV sebagai *filter* untuk meluncur di gambar asli. Nilai setiap piksel data berkorelasi lokal di dalam *filter* akan dikalikan dan ditambahkan sebagai hasil konvolusional. Jenis aturan ini disebut konvolusi yang memungkinkan fitur gambar diekstraksi dengan kernel CONV yang dirancang. Selain itu, metode menyaring bagian gambar yang berbeda dengan kernel CONV yang sama disebut *shared weight*, yang memungkinkan bahwa sel-sel netral dengan fitur yang sama dapat dikenali dan diklasifikasikan ke dalam jenis objek yang sama.

Parameter meliputi: ukuran kernel, kedalaman, langkah, zero-*padding*, dan jumlah *filter*. Penghitungan algoritma ukuran *output* ditunjukan pada persamaan 2.2 dan persamaaan 2.3: (Du, 2018).

$$H_{out} = 1 + \frac{H_{in} + (2\ x\ pad) - K_{height}}{s} \quad (2.2)$$

$$W_{out} = 1 + \frac{W_{in} + (2\ x\ pad) - K_{width}}{s} \quad (2.3)$$

3. Active Layer

Untuk membuat hasil dari lapisan CONV nonlinear*, active layer* diatur untuk menyelesaikan lenyapnya masalah gradien yang disebabkan oleh *underfitting*. Ada banyak fungsi untuk itu: Sigmoid, Tanh, *Rectified Linear Unit* (ReLU), LeLU Leaky, *Exponential Linear Unit* (ELU) dan *Maxout*. Baru-baru ini, Leaky ReLU adalah yang paling populer karena konvergensi lebih cepat daripada Sigmoid dan Tanh, menghitung cara yang lebih sederhana dan lebih efisien, dan tidak mengandung area mati. (Du, 2018)

4. *Pooling Layer*

Lapisan *pooling* digunakan untuk mengurangi dimensi hasil dari lapisan CONV, terletak di antara dua lapisan CONV. Terdapat tiga jenis penyatuan: *General Pooling*, *Overlapping Pooling* dan *Spatial Pyramid Pooling* (SPP). Lebar *General Pooling* sering sama dengan *stride*. Dengan menggunakan pengelompokan maksimum dan pengelompokan rata-rata *Overlapping Pooling* biasanya memiliki waktu lebih lama lebar dari *stride*. SPP dapat mengubah fitur konvolusional gambar dengan ukuran apa pun, menjadi jumlah dimensi yang sama. Keunggulan SPP ini memungkinkan CNN tidak hanya dapat menangani gambar yang multivariabel tetapi juga menghindari kehilangan informasi yang disebabkan oleh *cropping* dan *warping*, yang membuatnya inti dari SPP-Net. (Du, 2018)

5. Fully Connected Layer

fully connected layer seringkali merupakan lapisan terakhir di ujung CNN. Mereka mentransmisikan data ke *output*, serta menyederhanakan dan mempercepat perhitungan data.

6. Others layer

Beberapa model CNN masih memiliki lapisan putus sekolah dan lapisan regresi. Dimana lapisan sebelumnya berfungsi untuk mengatasi *overfitting*. Untuk menghindari bobot yang terlalu subyektif, biasanya lapisan ini memperbarui bobot sel saraf dengan probabilitas tertentu diputuskan oleh kebijakan stokastik. Lapisan regresi

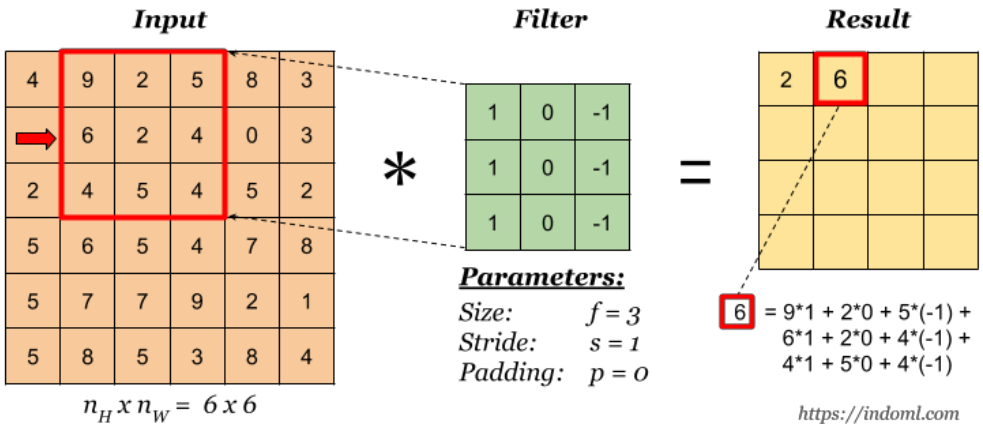
adalah untuk mengklasifikasikan fitur. Ada banyak metode: *Logistic Regression* (LR), *Gaussian Process for Regression* (GPR), *Bayesian Linear Regression* (BLR). Regresi memberikan probabilitas semua kemungkinan jenis objek, yang merupakan dasar penilaian akhir dari proses CNN (Du, 2018)

7. *Loss Function*

Loss function merupakan suatu fungsi yang bertujuan untuk mengukur tingkat performansi yang dihasilkan oleh model dalam melakukan prediksi terhadap target. *Loss function* bekerja ketika model pembelajaran memberikan kesalahan yang perlu diperhatikan. *Loss function* yang baik adalah fungsi yang menghasilkan *error* yang diharapkan paling rendah (Du, 2018)

2.8.4 Operasi Konvolusi

Operasi konvolusi ini memiliki tujuan utama untuk mengekstrak fitur dari gambar masukan. Operasi konvolusi ini bekerja dengan mempelajari fitur gambar menggunakan kotak-kotak kecil untuk mengetahui piksel pada data yang dimasukkan (Krizhevsky, 2012). Cara kerja *convolution* dapat dilihat pada Gambar 2.4 berikut ini



Gambar 2.4 Operasi Konvolusi (Honglak Lee, 2009)

Pada Gambar 2.4 matrik 3 x 3 menghitung elemen multiplikasi dan menambahkan hasil perkalian untuk mendapatkan bilangan bulat yang berbentuk satu elemen (Suartika, 2016). Dalam metode CNN matrik 3 x 3 tersebut disebut juga dengan " *Filter* " atau "kernel" atau "*feature detector* " dan matrik yang dibentuk dengan menggeser

filter diatas gambar yang dapat menghitung titik-titik *filter* disebut dengan "*convolve feature*" atau "*activation map*" atau "*feature map*".

Secara matematis, teknik konvolusi adalah teknik yang mempresentasikan jumlah lingkaran dari sudut fungsi F yang digeser atas fungsi G dan pada akhirnya menghasilkan fungsi h. Berikut adalah persamaan 2.4 yang digunakan dalam operasi konvolusi (Gazali, 2012)

$$\mathrm{h(x,y)} = \mathrm{F(x,y)} * \mathrm{G(x,y)} \tag{2.4}$$

Operasi konvolusi merupakan operasi dua fungsi argumen bernilai nyata x operasi ini menerapkan fungsi *output* sebagai *feature map* dari *input* citra. *Input* dan *output* ini dapat dilihat sebagai dua argumen bernilai riil. Secara formal operasi konvolusi ditulis pada persamaan 2.5

$$s(t) = (x * w)(t) \tag{2.5}$$

Fungsi s(t) memberikan *output* tunggal berupa *feature map*, argumen pertama adalah *input* yang merupakan x dan argumen kedua w sebagai kernel atau *filter*. Jika *input* dinyatakan dalam dua dimensi, maka t dinyatakan sebagai piksel dan menggantinya dengan i dan J. Pada persamaan 2.6 dinyatakan operasi untuk konvolusi ke *input* dengan nilai lebih dari satu dimensi.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.6}$$

Persamaan 2.6 merupakan perhitungan dasar dalam operasi konvolusi dimana i dan j adalah piksel dari citra. Perhitungannya bersifat komutatif dan muncul saat K sebagai kernel, I sebagai *input* dan kernel yang dapat dibalik relatif terhadap *input*. Sebagai alternatif operasi konvolusi dapat dilihat sebagai perkalian matriks antara citra *input* dan *filter* dimana keluarannya dapat dihitung dengan *dot product.*

Penentuan volume *output* dari masing-masing lapisan ditentukan dengan *hyperparameter*s. *Hyperparameter* yang digunakan pada persamaan 2.7 akan menghitung berapa banyak neuron aktivasi dalam sekali *output*.

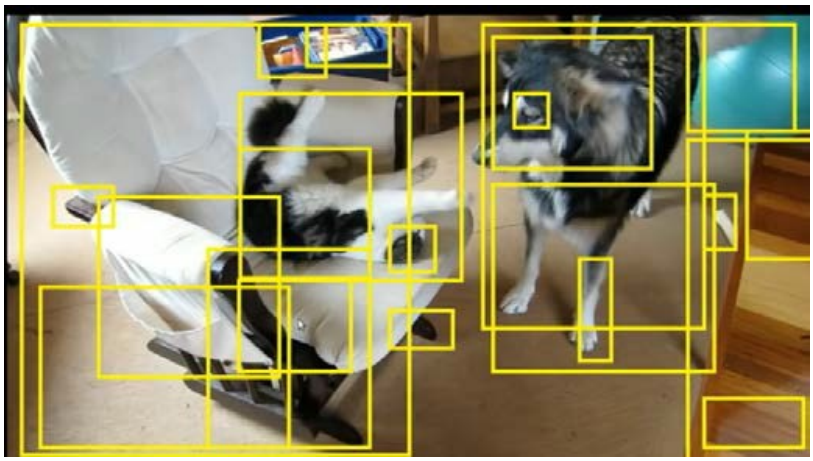
$$\frac{(W-F+2P)}{S} + 1 \tag{2.7}$$

Berdasarkan persamaan 2.7 ukuran spasial dari volume *output* dapat dihitung. Dimana *hyperparameter* yang dipakai adalah ukuran volume (W), *filter* (F), *stride* yang akan diterapkan (S) dan jumlah padding yang digunakan (P). *Stride* adalah nilai yang digunakan untuk menggeser *filter* melalui *input* citra dan *zero padding* adalah nilai untuk menempatkan angka nol di sekitar batas gambar (Goodfellow, 2016)

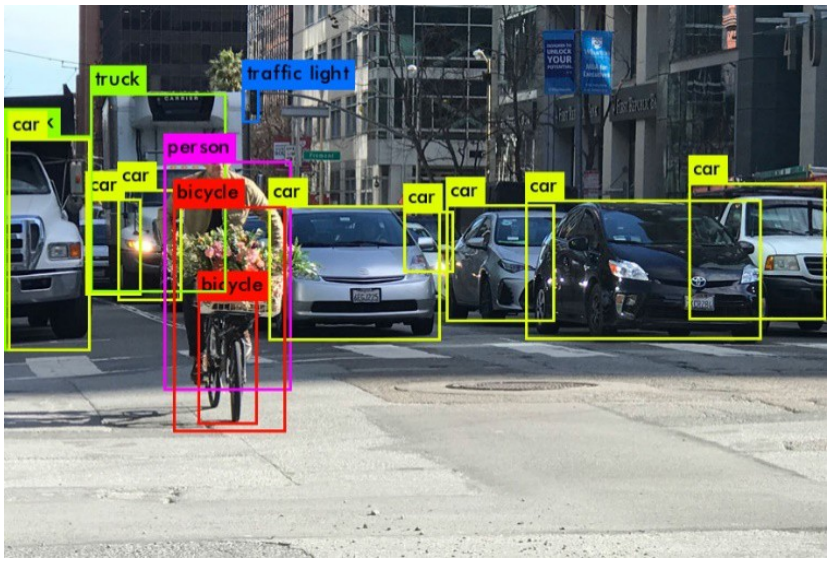
2.9 Algoritma YOLO

You Only Look Once (YOLO) adalah sebuah algoritma yang dikembangkan oleh Joseph Redmon, Ali Farhadi dkk untuk mendeteksi sebuah objek secara *real-time*. Sistem pendeteksian yang dilakukan adalah dengan menggunakan *repurpose classifier* atau *localizer* untuk melakukan deteksi. Sebuah model diterapkan pada sebuah citra di beberapa lokasi dan skala. Daerah dengan citra yang diberi *score* paling tinggi akan dianggap sebagai sebuah pendeteksian (Redmon J, 2016). Algoritma YOLO membingkai ulang objek yang ingin dideteksi sebagai pemetaan permasalahan regresi tunggal, piksel pada gambar yang telah dipecah akan di integrasikan dengan kelas gambar dan nilai probabilitas (*confidence value)* . YOLO menggunakan pendekatan *artificial neural network* untuk mendeteksi objek pada sebuah citra. Jaringan ini membagi citra menjadi beberapa wilayah dan memprediksi setiap kotak pembatas dan probabilitas untuk setiap wilayah. Kotak-kotak pembatas ini kemudian dibandingkan dengan setiap probabilitas yang diprediksi. YOLO memiliki beberapa kelebihan dibandingkan dengan sistem yang berorientasi pada *classifier*, terlihat dari seluruh citra pada saat dilakukan *test* dengan prediksi yang diinformasikan secara global pada citra.

prediksi yang dilakukan oleh YOLO tidak seperti pada sistem *Region-Convolutional Neural Network* (R-CNN) (Redmon J, 2016) yang membutuhkan ribuan untuk sebuah citra, hal ini di perlihatkan pada Gambar 2.5 sehingga membuat YOLO, lebih cepat hingga beberapa kali daripada R-CNN visualisasi penggunaan YOLO sebagai metode *object detection* di perlihatkan pada Gambar 2.6 (Du, 2018).



Gambar 2.5. Sistem Pendeteksian Objek menggunakan R-CNN



Gambar 2.6. Sistem Pendeteksian Objek menggunakan YOLO

2.9.1 Tahapan Pendeteksian Objek menggunakan YOLO

Ada 2 tahapan besar dalam proses mendeteksi objek usulan YOLO, tahapan tersebut adalah (Redmon J, 2016):

a. Memperbesar gambar masukan menjadi 448 x 448.

b. Membagi gambar seperti kotak-kotak berukuran S x S, kotak ini dinamakan *grid cell*. Setiap kotak dijadikan sebagai titik tengah dan diberikan tiga bentuk *Bounding Box*es *(BBoxes)*. *BBox*es ini digabungkan dengan *confidence*, penggabungan ini mempunyai kerja masing-masing yaitu memprediksikan apakah ada objek atau tidak. Jika diprediksi adanya objek maka *BBox*es akan

diberikan nilai satu, jika tidak diberikan nilai nol. *BBox*es yang bernilai satu akan menghasilkan *bounding box*. Setiap *bounding box* memiliki lima prediksi : x, y, w, h dan *confidence*. (x, y) sebagai titik koordinat tengah *grid cell*. (w, h) *width* dan *height* sebagai batas ukuran keseluruhan gambar (objek). *Confidence* adalah bentuk nilai keyakinan dari prediksi dimana S = 7, B = 3 dan C = 20. Terdapat dua proses dalam menemukan *bounding box*, yaitu :

1. *Bbox* dan *Confidence*

   Untuk menentukan nilai *confidence* dapat dilihat dari persamaan 2.8 berikut:

   $$Confidence = Probability * IoU \frac{Ground\ Truth}{Ground\ Prediction} \quad (2.8)$$

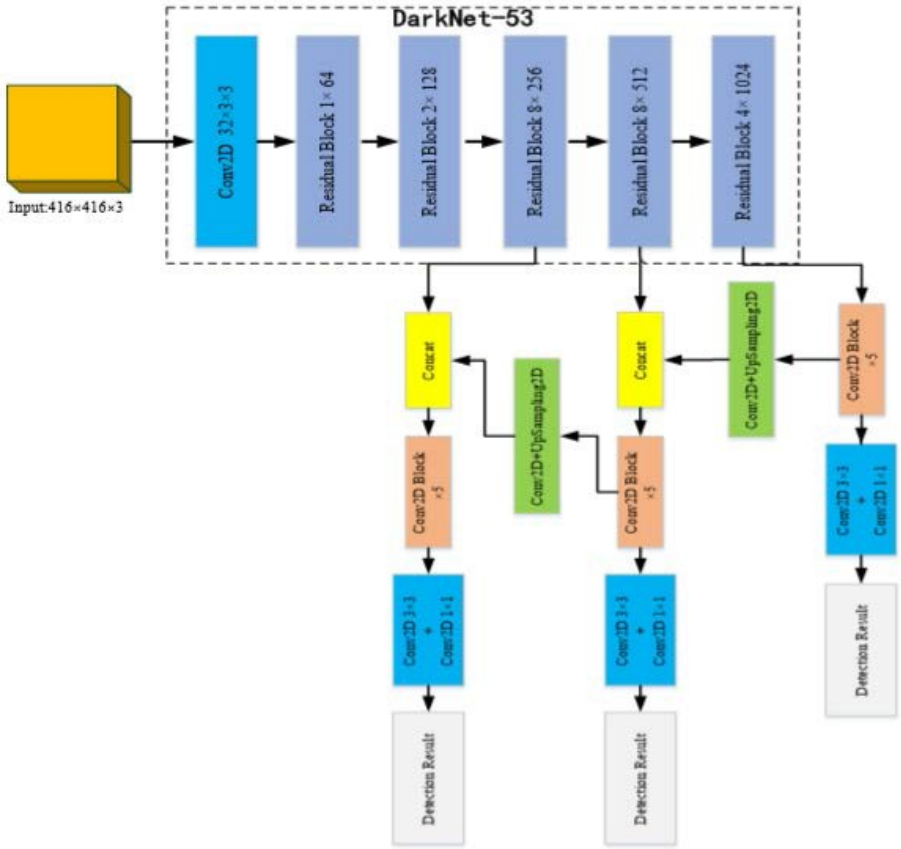
2. *Class Probability*

   Untuk mendapatkan *class probability* dijalankan operasi konvolusi. Setelah dilakukan operasi konvolusi didapatkan *BBox*es dan *confidence value,* maka dijalankan proses CNN pada setiap *BBox*es. Pada lapisan konvolusi pertama digunakan matrik 7 x 7 *stride* 2, selanjutnya matrik 3 x 3 *stride* 2, pada proses *pooling layer* digunakan matrik 2 x 2 *stride*, terakhir diikuti oleh *fully connected layer.* Setelah operasi konvolusi yang dilakukan selesai, dan didapatkan *Bbox, confidence* dan *class probability* yang dijadikan satu dalam satu nilai pembobotan atau weight.

2.9.2 YOLO V3

YOLO V3 merupakan versi pengembangan terbaru dari algoritma YOLO. YOLO V3 menggunakan jaringan *Darknet*-53 di sejumlah besar lapisan konvolusional berukuran 1 × 1 dan 3 × 3 dan memiliki total 106 lapisan konvolusi secara keseluruhan. Lapisan konvolusi ini berguna untuk menghubungkan interaksi fitur lokal. Algoritma YOLO V3 merupakan peningkatan pada YOLO V1 dan YOLO V2 karena memiliki kelebihan akurasi deteksi tinggi, penentuan posisi yang akurat, dan kecepatan cepat. Terutama ketika *multiscale detection* ketika diperkenalkan, dimana penggunaan YOLO V3 dapat mencapai deteksi target kecil dan memiliki

ketahanan yang baik untuk berbagai kondisi lingkungan, dikarenakan hal tersebut Algoritma YOLO V3 ini menjadi acuan bagi para akademisi untuk melakukan penelitian di pendeteksian objek.



Gambar 2.7. Struktur Jaringan YOLO V3 (Yi-Qi, 2020)

Struktur jaringan dari algoritma YOLO V3 ditunjukkan pada Gambar 2.7 Jaringan residual terutama digunakan untuk memutakhirkan jaringan *feature extraction*, dan jaringan *backbone* dasar diperbarui dari *Darknet*-19 ke *Darknet*-53 untuk mengekstraksi fitur dan mendapatkan lapisan penyempurnaan dan penambahan koneksi pintas. Operasi ini memungkinkan YOLO untuk mendapatkan informasi semantik yang lebih bermakna dari fitur-fitur sampel dan informasi yang lebih halus dari pemetaan fitur sebelumnya. Jaringan *feature extraction* ini memiliki 53 lapisan konvolusional, yaitu disebut *Darknet* -53, dan strukturnya ditunjukkan dibagi an atas pada Gambar 2.7 dalam *Darknet*-53, kernel konvolusi bergantian 1 × 1 dan 3 × 3 digunakan, dan setelah setiap lapisan konvolusi, lapisan *Batch Normalization* (BN) digunakan untuk normalisasi. Fungsi Leaky Relu digunakan sebagai *activation*

function, lapisan *pooling* dibuang, dan ukuran langkah kernel konvolusi diperbesar untuk mengurangi ukuran peta fitur. Karena struktur jaringan lebih dalam, kemampuannya untuk mengekstrak fitur juga ditingkatkan. Fungsi dari lapisan *sampling* (*upsample*) adalah untuk menghasilkan gambar ukuran kecil dengan interpolasi peta fitur ukuran kecil dan metode lainnya. Ketika koneksi pendek diatur antara beberapa lapisan untuk menghubungkan fitur tingkat rendah dengan yang tingkat tinggi, informasi berbutir halus dari fitur abstrak tingkat tinggi ditingkatkan, dan mereka dapat digunakan untuk prediksi kelas dan regresi kotak pembatas. Proses prediksi jaringan YOLO V3 tercantum di bawah ini:

1. Pertama, gambar berukuran 416 × 416 dimasukkan ke dalam jaringan *Darknet*-53. Setelah melakukan banyak konvolusi, sebuah peta fitur ukuran 13 × 13 diperoleh, dan kemudian 7 kali dengan konvolusi 1 × 1 dan 3 × 3 kernel diproses untuk mewujudkan kelas pertama dan prediksi kotak pembatas regresi.
2. Peta fitur dengan ukuran 13 × 13 diproses 5 kali oleh kernel konvolusi 1 × 1 dan 3 × 3, kali lapisan *upsampling*, dan jahitan dengan ukuran pada peta fitur 26 × 26. Fitur baru ini memetakan ukuran 26 × 26 yang diproses 7 kali menggunakan 1 × 1 dan 3 × 3 konvolusi kernel untuk mencapai kategori kedua dan prediksi kotak regresi yang terikat.
3. Peta fitur baru memiliki ukuran 26 × 26. Pertama, akan digunakan kernel konvolusi 1 × 1 dan 3 × 3 untuk memproses 5 kali, melakukan operasi *upsmpling* ganda, dan menjahitnya pada peta fitur size 52 × 52. Kemudian, yang fitur peta yang diproses 7 kali menggunakan 1 × 1 dan 3 × 3 konvolusi kernel untuk mencapai *Bounding box* prediksi YOLO V3.

Hal ini dapat dilihat dari hasil di atas, bahwa YOLO V3 dapat *output* tiga fitur peta dari ketiga ukuran yang berbeda pada yang sama waktu, 13 × 13, 26 × 26, dan 52 × 52.

Dalam hal ini fitur peta dari ukuran yang berbeda dioptimalkan untuk mendeteksi target kecil. Setiap peta fitur memprediksi tiga kotak pembatas regresi pada setiap posisi, masing-masing kotak pembatas berisi nilai keyakinan target, empat

nilai koordinat, dan kemungkinan tepi yang berbeda. Ada (52 × 52 + 26 × 26 + 13 × 13) × 3 = 10647 kotak terikat regresi (Yi-Qi, 2020)

2.9.2.1 YOLO V3 *Loss Function*

Pada persamaan 2.9 akan ditunjukan persamaan *loss function* yang digunakan di YOLO V3.

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
\tag{2.9}
$$

2.10 Framework Darknet

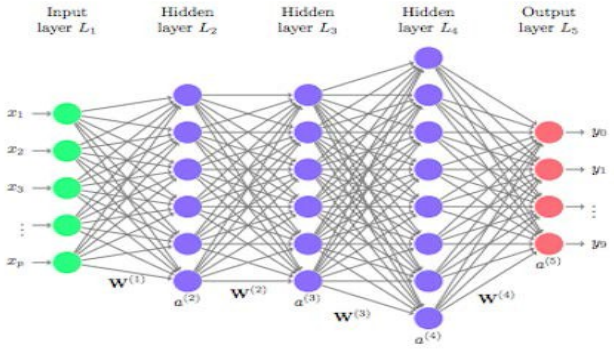
Darknet merupakan *framework neural network open source* yang ditulis dalam bahasa C dan CUDA. *Darknet* memiliki kemampuan komputasi yang cepat, mudah dipasang, dan mendukung komputasi CPU dan GPU. (Redmon J, 2016).

2.11 *Deep Learning*

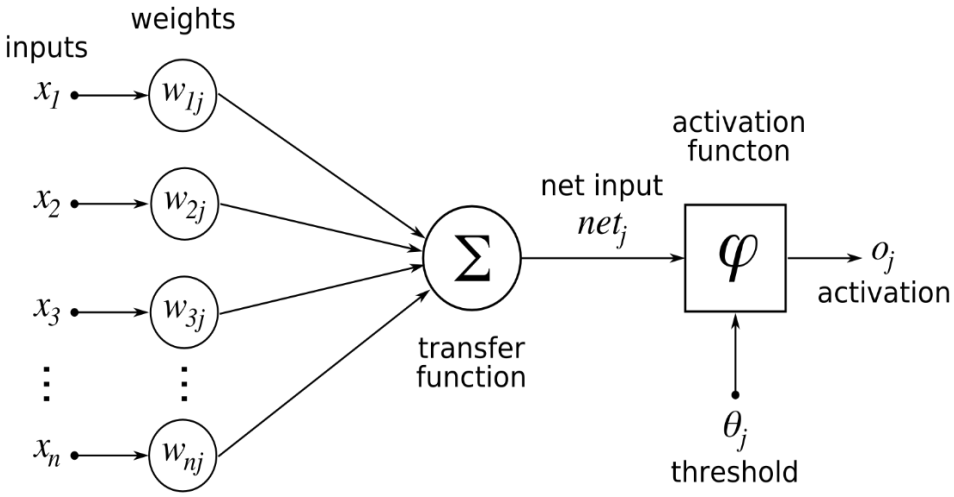
Machine learning merupakan salah satu subset dari *artificial intellegence* yang mampu mengekstraksi informasi dari suatu *dataset* yang tidak dapat diklasifikasi dengan peraturan yang jelas. *Deep learning* merupakan salah satu metode dari *machine learning*. *Deep learning* mempelajari representasi data dengan beberapa tingkat representasi yang didapatkan dari beberapa modul non-linear yang akan mentransformasi setiap nilai pada setiap level menjadi representasi yang memiliki tingak keabstrakan yang lebih tinggi (Yann LeCun, 2015)

Pada dasarnya arsitektur *deep learning* terdiri dari *input layer*, *hidden layer* dan *output layer* yang dapat dilihat pada Gambar 2.8. *Input layer* merupakan *layer* dengan

node yang terdiri dari data *input*. *Hidden layer* merupakan bagian dimana pengolahan data dilakukan, *hidden layer* dapat terdiri dari beberapa *layer*. *Output layer* merupakan *layer* dimana noda menunjukan nilai *output*. *Deep learning* terdiri dari kumpulan *neuron*, diagram *neuron* dapat dilihat pada Gambar 2.8 *Neuron* menerima beberapa *input* yang telah diberikan pembebanan yang kemudian dijumlah, diagram *neuron* dapat dilihat pada Gambar 2.9. *Neuron* akan memberikan batas dimana bila hasil jumlah dibawah batas maka *neuron* akan memberikan nilai 0 (Nilsson, 1998).



Gambar 2.8. Arsitektur *Deep learning*



Gambar 2.9. Diagram *Neuron*

2.12 *Optical Character Recognition*

Optical Character Recognition (OCR) adalah sebuah sistem komputer yang dapat membaca huruf, baik yang berasal dari sebuah pencetak (printer atau mesin ketik) maupun yang berasal dari tulisan tangan. OCR adalah aplikasi yang

menerjemahkan gambar karakter (*image character*) menjadi bentuk teks dengan cara menyesuaikan pola karakter per-baris dengan pola yang telah tersimpan dalam database aplikasi. Hasil dari proses OCR adalah berupa teks sesuai dengan gambar *output* scanner dimana tingkat keakuratan penerjemahan karakter tergantung dari tingkat kejelasan gambar dan metode yang digunakan.

Terdapat dua buah metode pada OCR, metode pertama yakni *template matching*. Pada dasarnya *template matching* adalah proses yang sederhana. Suatu citra masukan yang mengandung *template* tertentu dibandingkan dengan *template* pada basis data. *template* ditempatkan pada pusat bagian citra yang akan dibandingkan dan dihitung seberapa banyak titik yang paling sesuai dengan *template*. Langkah ini diulangi terhadap keseluruhan citra masukan yang akan dibandingkan. Nilai kesesuaian titik yang paling besar antara citra masukan dan citra *template* menandakan bahwa *template* tersebut merupakan citra *template* yang paling sesuai dengan citra masukan.

Metode yang kedua dari OCR adalah *feature extraction*. *Feature extraction* merupakan salah satu cara untuk mengenali suatu objek dengan melihat ciri-ciri khusus yang dimiliki objek tersebut. Tujuan dari *feature extraction* adalah melakukan perhitungan dan perbandingan yang bisa digunakan untuk mengklasifikasikan ciri-ciri yang dimiliki oleh suatu citra (Raden Sofian Bahri, 2012)

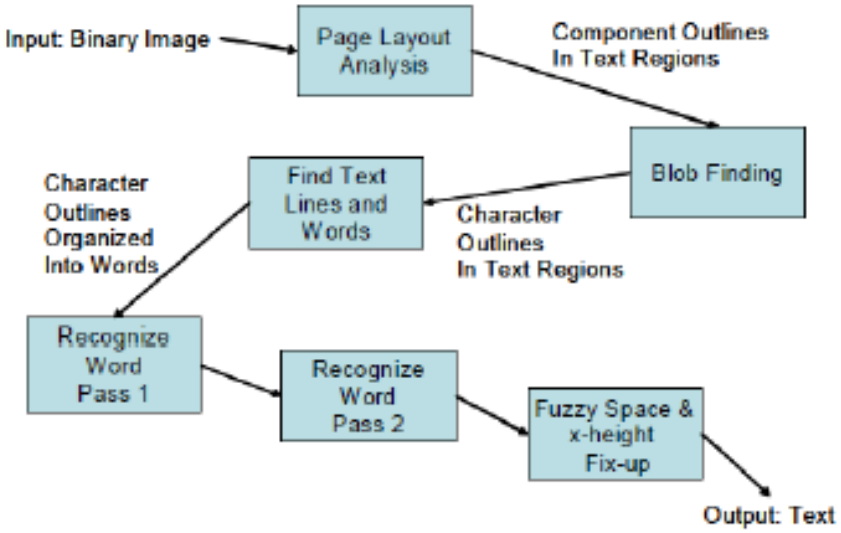
2.13 *Tesseract*

Tesseract merupakan *library optical character recognition* (OCR) yang bersifat *open source* untuk berbagai sistem operasi. *Tesseract* dikembangkan pertama kali oleh Hewlett Packard (HP) pada tahun 1985 hingga 1995. Setelah tahun 2006, *Tesseract* dilepas oleh HP untuk digunakan secara bebas. Sejak saat itu, *Tesseract* dikembangkan secara luas oleh Google dan dirilis di bawah lisensi Apache 2.0 (Smith, 2007). Pada saat ini *Tesseract* merupakan suatu *software* yang bersifat *open source* dan memiliki kapabilitas untuk mengenali lebh dari 100 bahasa di dunia. Hingga saat ini versi terbaru dari *Tesseract* ialah versi 4.11 (Google, 2019). Pada penelitian ini metode OCR yang akan digunakan adalah *template matching*.

2.13.1 Arsitektur *Tesseract*

Pada Gambar 2.10 dijelaskan terkait arsitektur dari *Tesseract*. *Tesseract* OCR mengasumsikan *input* yang diterima berupa sebuah binary *image*. Pertama, analisis

dilakukan pada komponen terhubung (*connected component*) (CC) untuk menemukan dimana *outline* komponen disimpan. Pada tahap ini *outline* dikumpulkan bersama menjadi *blob*. *Blob* disusun menjadi baris teks, sedangkan garis dan region dianalisis untuk *pitch* tetap dan teks proporsional. Baris teks dipecah menjadi kata-kata berbeda menurut jenis spasi karakter. Teks dengan *pitch* tetap dibagi menjadi sel-sel karakter. Teks proporsional dipecah menjadi kata-kata dengan menggunakan ruang pasti dan ruang *fuzzy*. Pengenalan kata pada *image* dilakukan pada dua tahap proses yang disebut *pass-two*



Gambar 2.10. Arsitektur *Tesseract* (Smith, 2007)

Pada *pass* pertama dilakukan untuk mengenali masing-masing kata pada gilirannya. Kata-kata yang sukses pada *pass* pertama yaitu kata-kata yang terdapat di kamus dan tidak ambigu kemudian diteruskan ke *adaptive classifier* sebagai data pelatihan. Begitu *adaptive classifier* memiliki sampel yang cukup, *adaptive classifier* ini dapat memberikan hasil klasifikasi bahkan pada *pass* pertama. Proses *pass* kedua dilakukan untuk mengenali kata-kata yang mungkin saja kurang dikenali atau terlewatkan pada *pass* pertama, pada tahap ini *adaptive classifier* telah memperoleh informasi lebih dari *pass* pertama. Tahap terakhir menyelesaikan ruang *fuzzy* dan memeriksa hipotesis alternatif pada ketinggian-x untuk mencari teks dengan *smallcap.*

Tesseract dirancang untuk mengenali teks putih di atas latar hitam dan teks hitam di atas latar putih. Hal ini menyebabkan rancangan mengarah pada analisis komponen terhubung *connected component* (CC) dan operasi pada *outline* komponen. Langkah pertama setelah analisis CC adalah menemukan *blob* pada region teks.

Sebuah *blob* merupakan unit putatif yang dapat diklasifikasikan, yang mana bisa satu atau lebih komponen-komponen yang saling tumpang tindih secara horizontal. Ada beberapa langkah yang dilakukan oleh *Tesseract* untuk pengenalan karakter adalah sebagai berikut:

2.13.2 Pencarian Teks-Line Dan Kata

Algoritma *line finding* dirancang supaya halaman yang miring dapat dikenali tanpa harus *de-skew* (proses untuk mengubah halaman yang miring menjadi tegak hurus) sehingga tidak menurunkan kualitas gambar Kunci bagian proses ini adalah *blob filtering* dan *line constuction*. (Smith, 2007).

Filtered blob lebih cenderung cocok dengan model *non-overlapping*, parallel, tetapi berupa garis-garis miring (*sloping line)*. Pemrosesan *blob* oleh koordinat x memungkinkan untuk menetapkan *blob* ke sebuah baris teks yang unik. Sementara penelusuran kemiringan di seluruh halaman, dengan banyak mengurangi bahaya penugasan ke baris teks yang salah dengan adanya kemiringan (*skew*). Setelah *blob* tersaring ditetapkan ke garis, sebuah median terkecil dari kotak-kotak yang cocok digunakan untuk memperkirakan *baseline*, dan *blob* yang sudah akan diberi *filter* dengan baik dipasang kembali ke garis yang sesuai. (Smith, 2007)

Langkah terakhir dari proses pembuatan garis *(line creation*) adalah menggabungkan *blob* yang *overlapping*, menempatkan *diacritical marks* dengan dasar yang tepat, dan menghubungkan bagian-bagian dari beberapa karakter yang rusak secara benar (Smith, 2007).

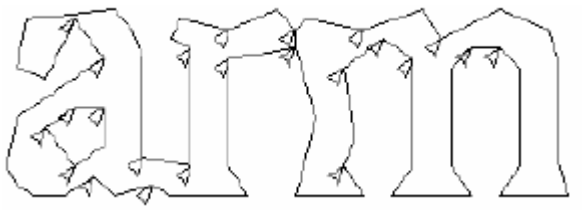
2.13.3 *Chopping*

Tesseract menguji garis teks (*text line*) untuk menentukan apakah mereka merupakan *fixed pitch*. Bila ditemukan *fixed pitch text, Tesseract* memotong kata-kata menjadi karakter-karakter. (Smith,2007). Pemotongan karakter pada *software Tesseract* ditunjukan pada Gambar 2.11

Gambar 2.11. Pemotongan Karakter (Smith, 2007)

2.13.4 Pemisahan Karakter Terhubung

Apabila hasil dari pengenalan kata tidak memuaskan, *Tesseract* berusaha untuk memperbaiki hasil dengan memisahkan *blob* dengan keyakinan terburuk dan pengklasifikasian (*classifier*) karakter. Kandidat untuk titk-titik pemisahan diternukan dari simpul cekung dari pendekatan poligonal *outline* dan mungkin saja terdapat titik cekung berlawanan lainnya atau segmen garis. Titik pemotongan berguna untuk memisahkan karakter yang terhubung dan set ASCII.



Gambar 2.12.. Kandidat titik Potong (Smith, 2007)

Gambar 2.12 di atas menunjukan satu set calon titik potong (*chop points*) dengan tanda panah dan potongan terpilih sebagai sebuah garis melintasi kerangka dimana huruf ‘r’ bersentuhan dengan huruf ‘m’ (Smith, 2007)

2.13.5 Asosiasi Karakter Patah

Ketika potongan yang potensial tidak ada lagi, ketika kata tersebut masih belum cukup baik untuk dikenali, Hal ini diberikan kepada *associator*. *Associator* membuat pencaran. A* (*best first search*) dari segmentasi grafik yang mungkin kombinasi dari *blob* yang dipotong secara maksimal ke dalam kandidat karakter. Ketika A * *segmentation* digunakan untuk diimplementasikan pertama kali pada tahun 1989, akurasi *Tesseract* terhadap karakter yang rusak cukup baik yang menjadikan *Tesseract* mesin komersial pada saat itu. Pada Gambar 2.13 dijelaskan tentang gambar yang sudah rusak dan tidak bisa dikenali

Gambar 2.13.. Gambar yang sudah rusak dan tidak bisa dikenali (Smith, 2007)

2.14 Evaluation Metrics

2.13.1 Akurasi

Akurasi merupakan metode pengujian berdasarkan tingkat kedekatan antara nilai prediksi dengan nilai aktual. Dengan mengetahui jumlah data yang diklasifikasikan secara benar maka dapat diketahui akurasi hasil prediksi. Persamaan akurasi seperti pada persamaan berikut (Dewantoro, 2020). :

$$\text{Akurasi Deteksi } (\%) = \frac{jumlah\ total\ objek\ yang\ berhasil\ di\ deteksi}{jumlah\ keseluruhan\ objek} \text{ x } 100$$

2.13.2 *Precision* dan *Recall*

Precision dan *Recall digunakan untuk mengukur* kinerja sistem. *Precision* adalah kecocokan antara bagian data yang diambil dengan informasi yang dibutuhkan. *Recall* merupakan tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. (Villman, 2014) nilai dari *Precision* dan *recall* dapat dihitung menggunakan *confusion matrix*. Tabel *confusion matrix* dapat dilihat pada tabel 2.2

Tabel 2.2 Tabel *Confusion matrix*

	True	False
True (Positive)	*TP* *(True Positive)* *Correct result*	*FP* *(False Positive)* *Unexpected result*
False (Negative)	*FN* *(False Negative)* *Missing Result*	*TN* *(True Negative)* *Correct absence of result*

$$Precision = \frac{TP}{TP+FP} \qquad (2.7)$$

$$Recall \quad = \frac{TP}{TP+FN} \qquad (2.8)$$

Keterangan :

TP = Banyak data dengan nilai sebenarnya positif dan nilai prediksi positif

FP = Banyak data dengan nilai sebenarnya negatif dan nilai prediksi positif

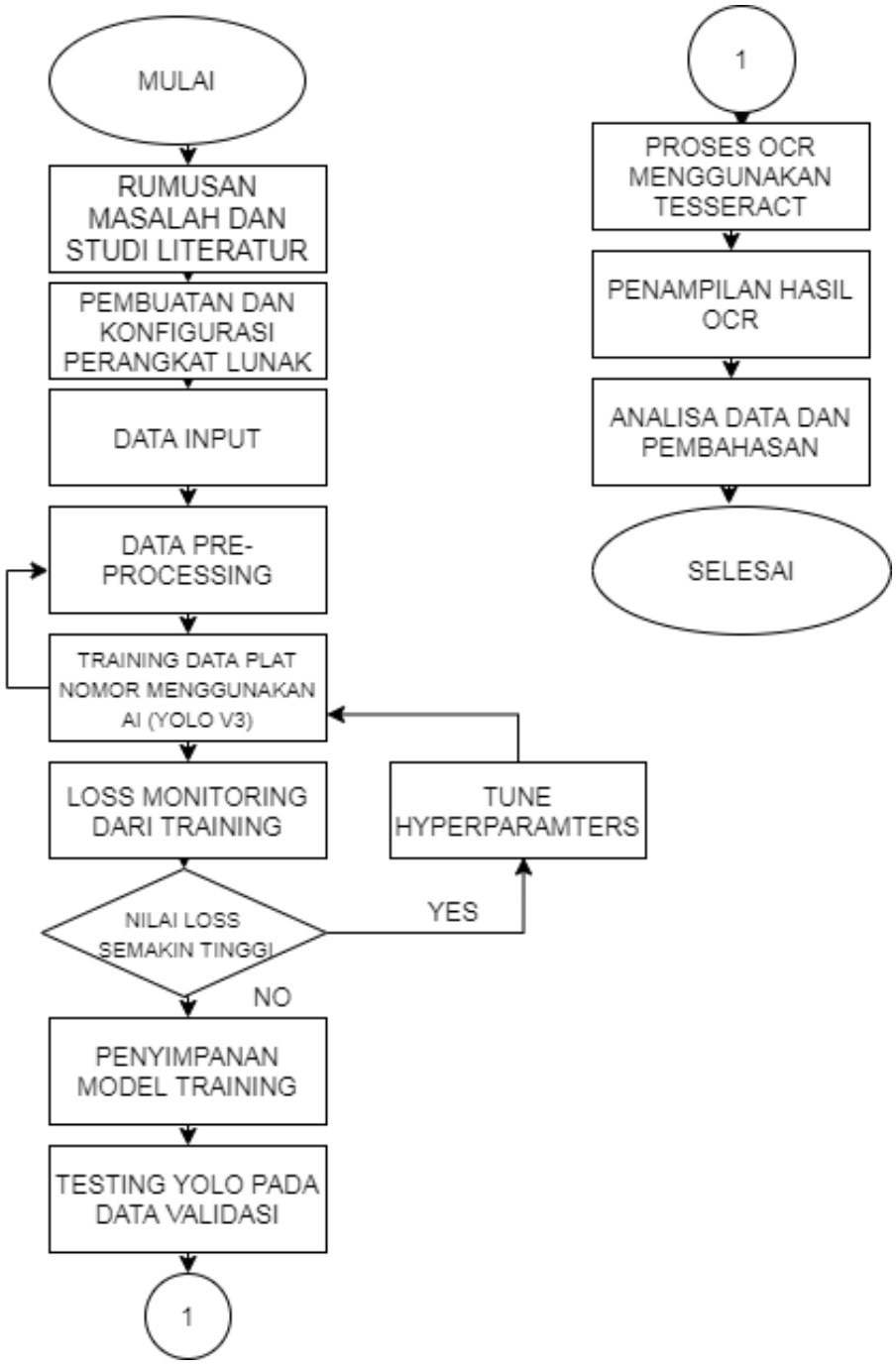
FN = Banyak data dengan nilai sebenarnya positif dan nilai prediksi negatif

TN= Banyak data dengan nilai sebenarnya negatif dan nilai prediksi negatif

Halaman ini sengaja dikosongkan

BAB III
METODOLOGI PENELITIAN

Tahapan yang dilakukan dalam penelitian tugas akhir ini dapat ditampilkan dalam sebuah diagram alir pada Gambar 3.1.



Gambar 3.1 Diagram Alir Pengerjaan Tugas Akhir

Penjelasan secara detail dari diagram alir pada Gambar 3.1 dijelaskan pada sub bab berikut ini.

3.1 Perumusan Masalah

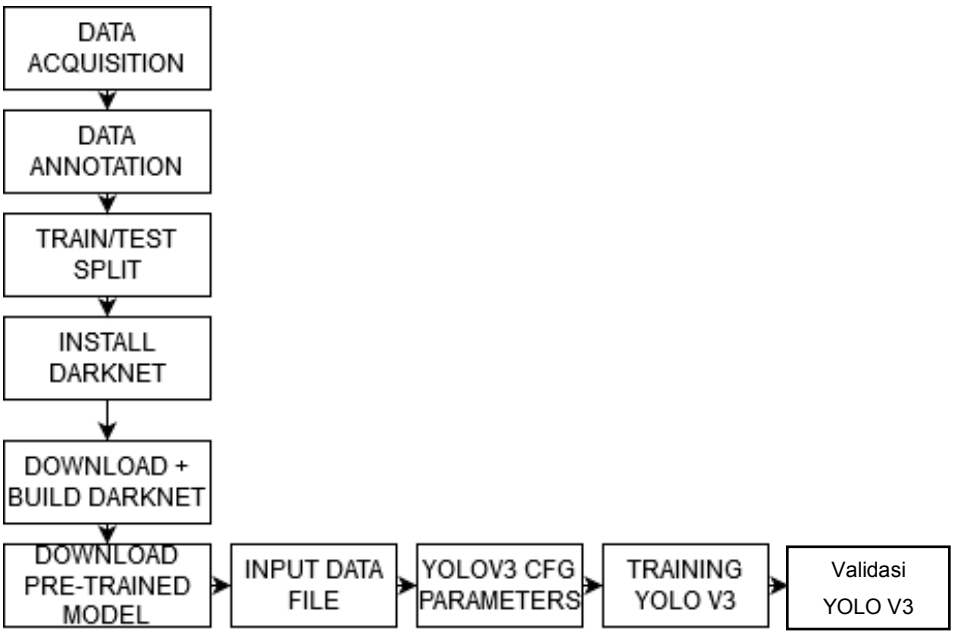
Langkah pertama untuk melakukan penelitian ini adalah tahapan perumusan masalah yaitu menentukan dan mengidentifikasi setiap masalah yang ada dalam metode yang telah dilakukan untuk deteksi plat nomor di penelitian sebelumnya dan berdasarkan masalah-masalah tersebut akan dirumuskan suatu rumusan masalah yang akan menjadi tujuan dan penentuan metode yang akan digunakan dalam pengerjaan tugas akhir ini.

3.2 Studi Literatur

Penelitian ini dimulai dengan melakukan studi literatur terkait pemrosesan citra secara pertimbangan metode analisis menggunakan *convolutional neural network*, hasil pengolahan data menggunakan YOLO V3 dan referensi pembuatan sistem pendeteksi menggunakan algoritma CNN. Dilanjutkan dengan studi literatur lebih jauh terkait *optical character recognition* yang akan digunakan untuk bisa memindai alphanumerik dalam Tanda Nomor Kendaraan Bermotor.

3.3 Pembuatan dan Konfigurasi Perangkat Lunak

Perangkat lunak yang akan digunakan dalam membuat algoritma pembacaan plat nomor otomatis adalah Python. Setelah program yang akan digunakan ditentukan maka akan dibuat algoritma pembuatan program seperti pada Gambar 3.2



Gambar 3.2 Alur Pembuatan Perangkat Lunak

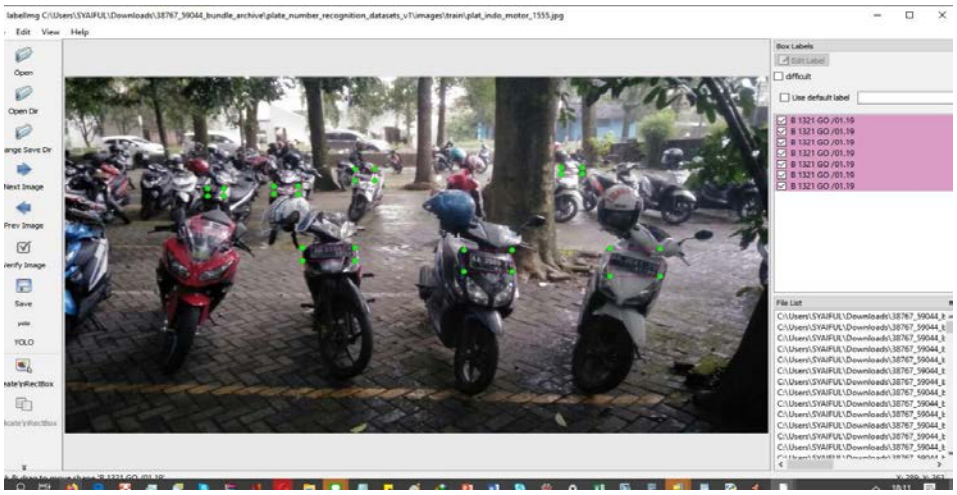
Gambar 3.2 menjelaskan proses yang dilakukan dalam pembuatan perangkat lunak untuk pembuatan model AI. Proses pembuatan perangkat lunak ini diawali dengan *data acquisition* atau proses perolehan gambar yang akan menjadi data *input* pada *training model*. Plat nomor kendaraan bermotor sesuai SNI difoto dan disimpan, kemudian dilanjutkan dengan proses anotasi, atau pelabelan pada bagian dari objek yang ingin dideteksi. Setelah itu data hasil perolehan dan pelabelan akan dibagi menjadi data *training* dan data *testing*. Dilanjutkan dengan pengunduhan *darknet* sebagai *framework* yang akan dipakai dalam *training* dan penentuan konfigurasi dari *hyperparameters* yang ingin digunakan dalam proses *training*. Dalam proses ini dilakukan *transfer learning* dari model yang sudah ada, hal ini diperlukan agar model bisa lebih cepat belajar dengan cara memanfaatkan pembelajaran dari model yang sudah dilatih sebelumnya. Metode *transfer learning* merupakan suatu metode yang memanfaatkan model yang sudah dilatih terhadap suatu *dataset* untuk menyelesaikan permasalahan lain yang serupa dengan cara menggunakannya sebagai *starting point*, memodifikasi dan memperbaharui parameternya sehingga sesuai dengan *dataset* yang baru.

Dengan adanya suatu *transfer learning* ini diharapkan model yang digunakan untuk pendeteksian plat nomor bisa mencapai tingkat akurasi yang tinggi meski

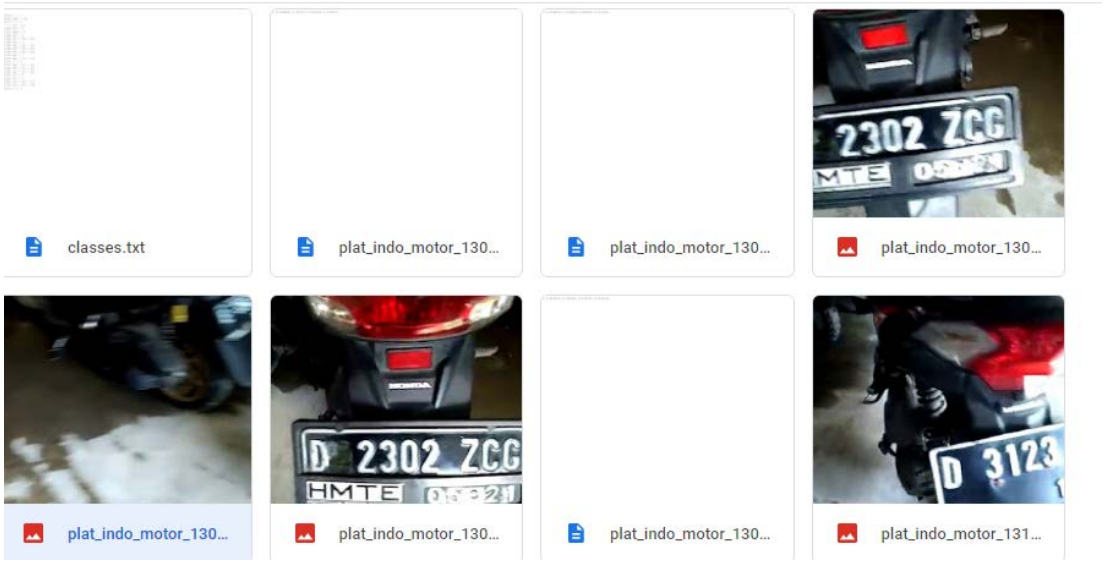
dengan *dataset* yang sedikit. dilanjutkan dengan pemasangan konfigurasi dari parameter dari *hyperparameters* yang akan digunakan untuk *training dataset* yang telah disiapkan. Pemasangan konfigurasi parameter ini dilakukan agar lebih mudah untuk mengatur jalannya *training* data. Dalam konfigurasi *hyperparameters* yang berada di dalam lampiran terdapat proses *pre-processing* yang dilakukan kepada data yaitu *rescaling*, atau perubahan ukuran dari gambar. Hal ini diperlukan agar bisa mendapatkan hasil yang lebih akurat dari proses *training* dan memimimalisir waktu untuk komputasi.

3.4 Proses *Input* Data

Persiapan dilakukan dengan mengumpulkan data (*data acquisition*) yang berupa gambar atau foto plat nomor tanda kendaraan bermotor sesuai SNI yang berada di seluruh Wilayah Negara Republik Indonesia dan diberikan suatu label yang menunjukan lokasi plat nomor (*data annotation*) menggunakan *software* LabelImg, proses pelabelan dalam gambar ditunjukan pada Gambar 3.3. Pelabelan ini dilakukan untuk melatih mesin agar bisa mengenali suatu objek spesifik dalam sebuah gambar dengan label tertentu. Pada penelitian ini jenis label yang digunakan ialah label plat nomor. Setelah dilakukan pelabelan, data akan diklasifikasi (*data classification*) berdasarkan label XML, TXT, dan IMG. Lalu dari jumlah keseluruhan data, akan dilakukan pemisahan antara *data training* dan *data testing* (*train-test split*) untuk penelitian ini dilakukan *train-test split* sebesar 80:20 (Yi-Qi, 2020). Setelah dilakukan proses *data acquisition, data annotation* dan *data classification* akan dilanjutkan proses *training* terhadap data *input*. Gambar yang sudah di *input* pada *google collab* ditunjukan pada Gambar 3.4, pada Gambar 3.4 ditunjukan terkait komposisi data yang akan digunakan di dalam *training,* yaitu data dengan format TXT dan JPG. File yang berwarna putih merupakan file TXT sedangkan yang memiliki gambar plat nomor merupakan file JPG.



Gambar 3.3 Proses *Labelling* (Anotasi) menggunakan *Software Label Img*



Gambar 3.4 Proses *Input* data

3.5 Implementasi Sistem

Implementasi sistem dilakukan pada computer dengan spesifikasi RAM: 12.0 GB DDR3, GPU: Nvidia GeForce GTX 820M, HDD: 1 TB, CPU: Core i5, dan OS Windows 10 Professional. Implementasi pengkodean sistem untuk memvalidasi hasil *training* YOLO, menggunakan bahasa pemrograman Python dengan versi 3.7.0, Anaconda versi 3.0, *Framework* yang digunakan untuk sistem ini adalah *Darknet*

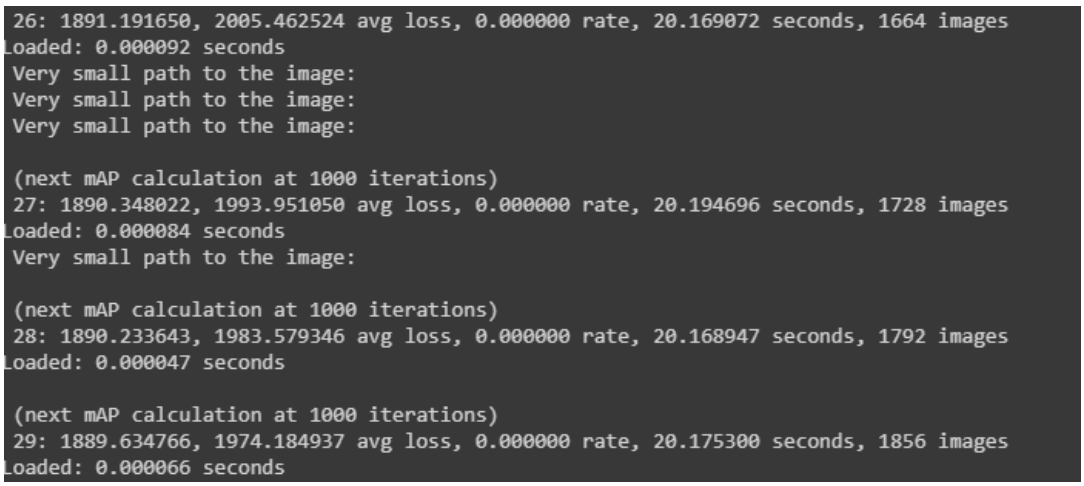
adapun beberapa *library* yang mendukung sistem ini adalah *PyTesseract, Numpy* dan *OpenCV2*.

3.6 *Training Neural network*

Pada model *training neural network* semua file yang sudah dilabeli *bounding box* (*annotated data*) sebelumnya di *upload* pada piranti google colab. Dalam proses *training*, data akan dilakukan *pre-processing* terlebih dahulu, dimana ukuran dari gambar akan diseragamkan pada ukuran gambar tertentu (*rescaling*). Hal ini dibutuhkan agar diperoleh hasil *training* yang ideal dan waktu komputasi yang lebih minimum. *Training* ini akan dilakukan secara berulang hingga *training* berhenti, yang akan menyatakan nilai maksimum iterasi telah dicapai. Setelah dilakukan *training*, *software* model AI akan menjadi semakin pintar dan untuk mengujinya dilakukan validasi data. Validasi data akan dijelaskan lebih lanjut pada poin 3.8.

Sebelum dilakukan *training* ditetapkan *hyperparameters* tertentu untuk melatih *data training*. Setiap hasil *weight* dari *training* akan disimpan per 1.000 iterasi. Dimana hasil *weight* tersebut akan menjadi bahan uji untuk identifikasi plat nomor pada *data testing*. Gambar akan dilatih sebanyak iterasi tertentu, hingga *training* berhenti sendiri. Semua *weight* pada data pelatihan diinisialisasi menggunakan random *weight* tertentu mengikuti *normal distribution*. Proses *training* ini dibantu dengan *backpropagation neural network* untuk bisa mencapai tingkat akurasi yang di kehendaki. Dimana dari hasil keseluruhan *weight* yang didapatkan akan diuji model terbaik nya. Dalam proses *training* dilakukan proses konvolusi pada tiap gambar yang akan diuji. Dalam proses konvolusi data *input* akan dilatih mengikuti konfigurasi dari *filter* dan *hyperparameters* yang sudah ditetapkan sebelumnya, proses konvolusi ini akan melewati beberapa lapisan pada CNN. *Network* terakhir pada CNN merupakan *network* yang digunakan untuk mengklasifikasi dan regresi untuk menentukan lokasi atau *region* yang mengandung objek pada sebuah gambar dan *class* dari objek. Setelah mendapatkan *region* yang mengandung objek, *region-region* tersebutkan melalui tahap yang dinamakan ROI *Pooling*. ROI *Pooling* berfungsi untuk menyamakan dimensi *width* dan *height* dari *region-region* yang telah didapatkan oleh *Region Proposal Network*. Hal ini dilakukan karena CNN hanya dapat menerima *input* dengan dimensi yang sama. Pada Gambar 3.5 ditunjukan terkait proses *training*

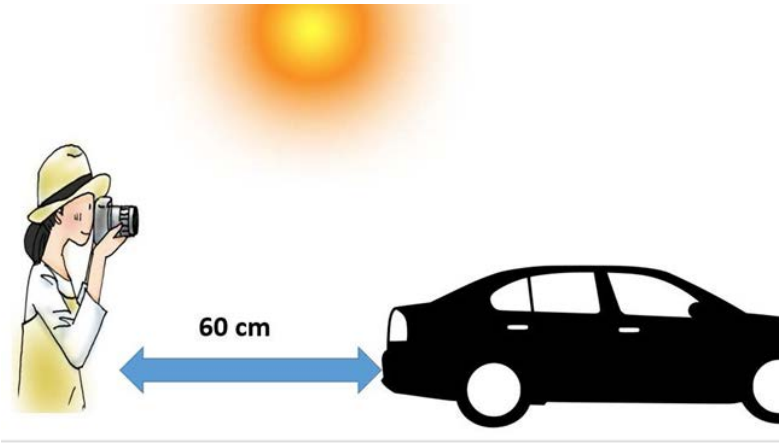
yang dilakukan pada google colab. Hasil yang didapatkan dari proses *training* pada iterasi tertentu ini adalah *weight*. Hasil *weight* inilah yang akan menjadi data untuk melakukan proses validasi, seperti tercantum pada poin 3.8.



Gambar 3.5 Proses ketika dilakukan *training* pada Google colab

3.7 Pengambilan Data Untuk Uji Validasi

Proses pengambilan data untuk uji validasi dijelaskan pada Gambar 3.6 Pada Gambar 3.6 dijelaskan terkait proses pengambilan data untuk uji validasi dimana hasil gambar yang akan di foto menggunakan kamera ponsel Samsung Galaxy A9 Pro dengan nomor model SM-A910F/DS dan nomor Seri RR8HC03L6EK pada rentang waktu antara 09.00-15.00. Nilai Lumen pada waktu pengambilan akan dihitung menggunakan Aplikasi Lux Light Meter Free yang dibuat oleh Doggo Apps pada versi 018.2019.07.24 dengan *update* terakhir pada *software* ini dilakukan oleh *developer* pada 27 Juli 2019. Rentang jarak pengambilan adalah ± 60 cm.



Gambar 3.6 Skema Pengambilan Gambar

3.8 Validasi YOLO pada Python

Setelah mendapatkan hasil *weight* yang didapatkan pada poin 3.6 tiap *weight* hasil iterasi akan diunduh dan akan menjadi suatu rujukan untuk diuji pada Python. Jika berhasil dikenali dalam sistem, maka akan ditampilkan *confidence value* yang menyatakan probabilitas benda tersebut berada di gambar. Hasil pembacaan YOLO dinyatakan benar jika terdapat kotak yang menunjukan jenis objek, angka yang menunjukan probabilitas keberadaan benda yang berhasil dipindai oleh YOLO dan jenis objek yang ingin dikenali. Contoh gambar hasil pembacaan pendeteksian objek oleh YOLO ditunjukan pada Gambar 3.7.

Gambar 3.7 Hasil prediksi Plat menggunakan YOLO V3

3.9 Proses OCR

Seteah dilakukan proses validasi YOLO menggunakan Python, dilakukan *testing* data pengenalan alphanumerik pada plat nomor mobil menggunakan *software Tesseract*. Pada proses ini gambar keluaran dari YOLO akan diproses oleh *software Tesseract* yang di operasikan di Python, dimana gambar *input Tesseract* akan menyesuaikan dengan *library* yang dimiliki oleh *Tesseract* (*template matching*). Jika berhasil, hasil pendeteksian alphanumerik pada plat nomor akan ditampilkan pada layar komputer yang berisi pengenalan karakter data *input* oleh *Tesseract.*

3.10 Analisa Data Dan Penyusunan Laporan

Pengujian sistem dilakukan dengan cara mengevaluasi performa daripada sistem *training* dan sistem *testing* yang telah dibuat. Pencatatan pengujian akan dibedakan berdasarkan jenis plat yang berhasil diidentifikasi dengan ditandai, adanya *bounding box, class* akan menjadi parameter keberhasilan dari pemindaian plat nomor oleh sistem. Dilakukan suatu analisa yang membahas pada sistem pemindai plat nomor untuk mampu membaca keterkenalan objek pada iterasi yang telah diinisialisasi. Lalu akan diambil hasil terbaik dimana tingkat akurasi maksimal. Dilanjutkan dengan proses *optical character recognition* yang akan membaca pengenalan karakter alphanumerik pada plat nomor di gambar yang telah diharapkan. Penyusunan laporan Tugas Akhir merupakan pembukuan serta dokumentasi yang menghasilkan sebuah tulisan ilmiah sebagai tahap akhir dari penelitian ini.menghasilkan sebuah tulisan ilmiah sebagai tahap akhir dari penelitian ini.

Halaman ini sengaja dikosongkan

BAB IV
HASIL DAN PEMBAHASAN

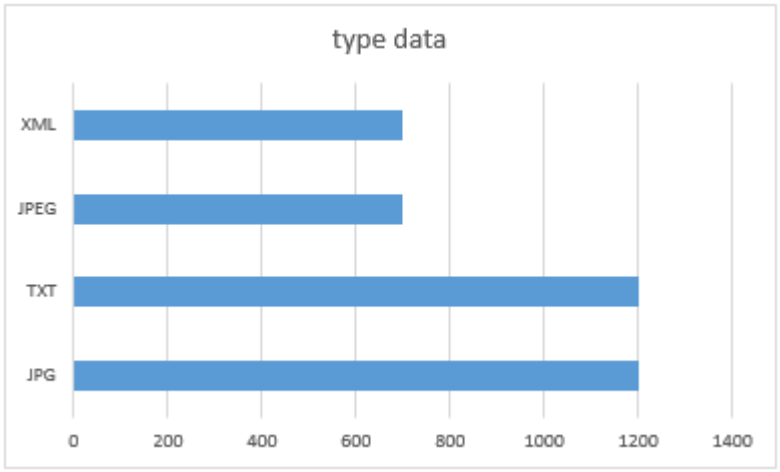
Untuk mendapatkan model *Artificial intellegence* dilakukan beberapa tahap yang dapat dilihat pada Gambar 3.1. *Dataset* yang telah disiapkan akan dilabeli (Anotasi) terlebih dahulu menggunakan *software* Labelimg lalu seluruh data yang dikumpulkan akan dibagi menjadi data *training, testing* dan data Validasi. Dengan perbandingan 80:20 untuk data *training* dan *testing*. Data *training* digunakan untuk melatih model *Artificial intellegence* yang telah di buat (AI), data *testing* digunakan untuk menguji model AI dengan dengan *dataset* yang sama dan data validasi digunakan untuk menguji performansi model terhadap gambar baru. Pada tahap *training* model *artificial intellegence* (AI) dan *dataset* akan di *training* menggunakan *hyperparameters* pada tabel 4.1 hingga menemukan *weight* yang sesuai untuk dilakukan pengujian validasi model AI. Tahap selanjutnya dilakukan dengan menguji model dengan gambar yang baru, lalu dihitung performansi model dengan *accuracy rate* dan *confusion matrix*. Berdasarkan hasil perolehan data yang didapatkan maka akan dilakukan suatu analisa. Tahap selanjutnya adalah proses *optical character recognition* (OCR) menggunakan *Tesseract*. Seluruh kode Python yang digunakan pada penelitian ini dapat dilihat lampiran.

Tabel 4.1 *Hyperparameters* yang digunakan untuk *training*

Hyperparameters	*description*
Batch	64
Subdi*vision*s	16
Width	416
Height	416
Channels	3
Momentum	0, 9
Decay	0, 0005
Angle	0
Saturation	1.5
Exposure	1.5
Hue	1
Learning rate	0,001
Burn_in	1.000
Max Batches	500200
policy=stepssteps	4.00000, 45.0000
Scales	1, 1

4.1 Pengumpulan *Dataset*

Pengumpulan *dataset* serta proses anotasi, didapatkan dari data primer, yang diperoleh dari lapangan. Didapatkan total *dataset* sebanyak 3.800 plat nomor dengan format JPG, JPEG, XML dan TXT. Pada penelitian ini akan dilakukan *training* dengan file ekstensi JPG dan hasil anotasi berupa TXT. Berdasarkan hasil pengumpulan *dataset* didapatkan sebanyak 2.404 plat nomor. *Dataset* yang telah dikumpulkan dan tidak terpakai akan dimasukan ke Kaggle.com, sebagai bentuk sumbangsih *dataset* plat nomor yang dapat dilakukan untuk penelitian dikemudian hari. Hasil pengumpulan *dataset* inilah yang akan dijadikan data *training* dan data *test*. Pada Gambar 4.1-4.3 akan ditunjukan terkait statistikal *dataset* yang sudah diperoleh



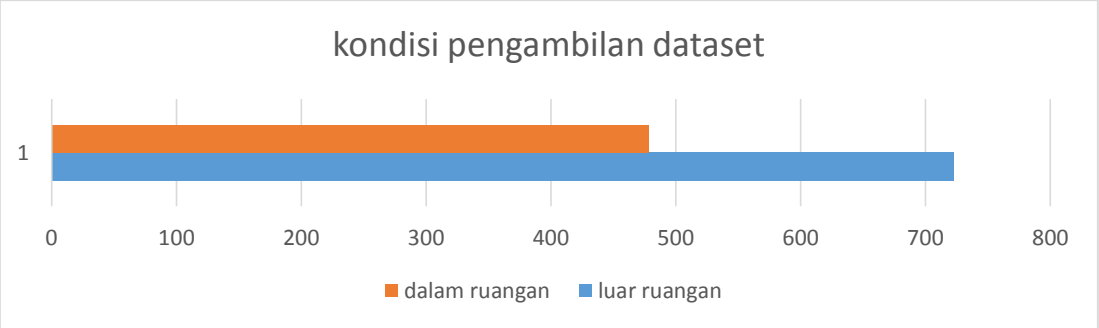
Gambar 4.1 Tipe data yang diperoleh

Pada Gambar 4.1 terlihat bahwa jumlah data secara keseluruhan ialah 1.900 data yang terdiri dari :

- Tipe data JPG : 1.202 data
- Tipe data TXT : 1.202 data
- Tipe data JPEG: 698 data
- Tipe data XML: 698 data

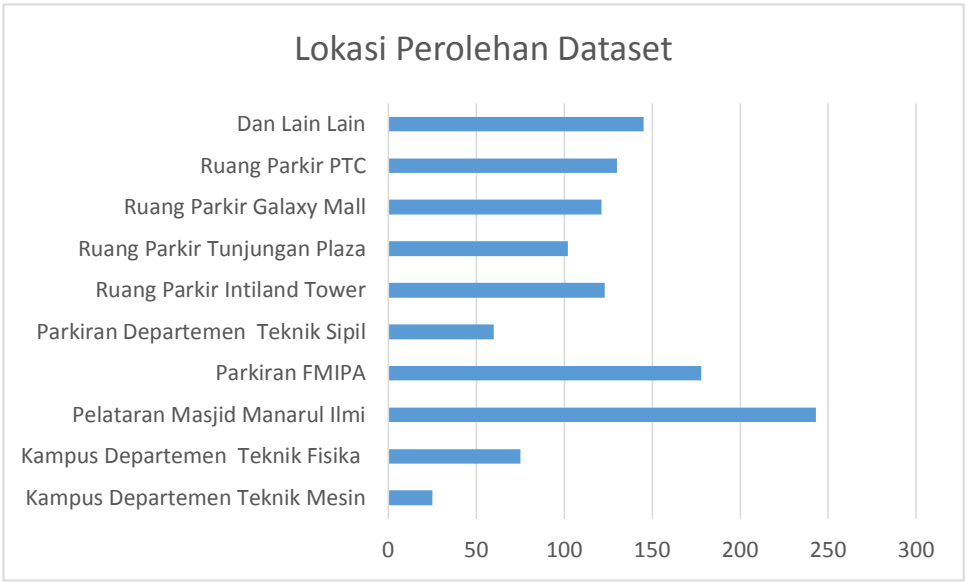
Data yang digunakan dalam penelitian ini adalah tipe data. TXT dan. JPG, sedangkan data yang lainya tidak diperlukan.

Kondisi pengambilan gambar dilakukan pada siang hari dalam rentang waktu jam 09.00 – 15.00, dan dilakukan pengambilan gambar di dalam ruangan dan di luar ruangan. Gambaran kondisi pengambilan *dataset* dapat dilihat pada Gambar 4.3.



Gambar 4.2 Kondisi Pengambilan *Dataset*

Berdasarkan Gambar 4.3 jumlah gambar yang diambil di dalam ruangan memiliki jumlah 479 gambar dan gambar yang diambil di luar ruangan berada 723 gambar. Pengambilan *dataset* yang berada pada dua kondisi ini diharapkan Ketika diuji validasi model AI, model dapat mengenali plat nomor yang berada di kondisi tertentu.



Gambar 4.3 Lokasi Perolehan *Dataset*

Gambar 4.3 menjelaskan terkait lokasi pengambilan *dataset* yang akan digunakan oleh model. Perolehan *dataset* ini diambil pada kota Surabaya, dimana lokasi pengambilan *dataset* adalah sebagai berikut :

- Kampus Departemen Teknik Mesin sebanyak 25 gambar,
- Kampus Departemen Teknik Fisika sebanyak 75 gambar,
- Plataran Masjid Manarul Ilmi sebanyak 243 gambar,
- Parkiran Fakultas Matematika dan Ilmu Pasti Alam (FMIPA) sebanyak 178 gambar,
- Parkiran Departemen Teknik Sipil sebanyak 60 gambar
- Ruang Parkir Intiland Tower sebanyak 128 gambar,
- Ruang Parkir Tunjungan Plaza sebanyak 102 gambar,
- Ruang Parkir Galaxy Mall sebanyak 121 gambar,
- ruang parkir Pakuwon Trade Center sebanyak 121 gambar dan
- tempat lain sebanyak 149 gambar.

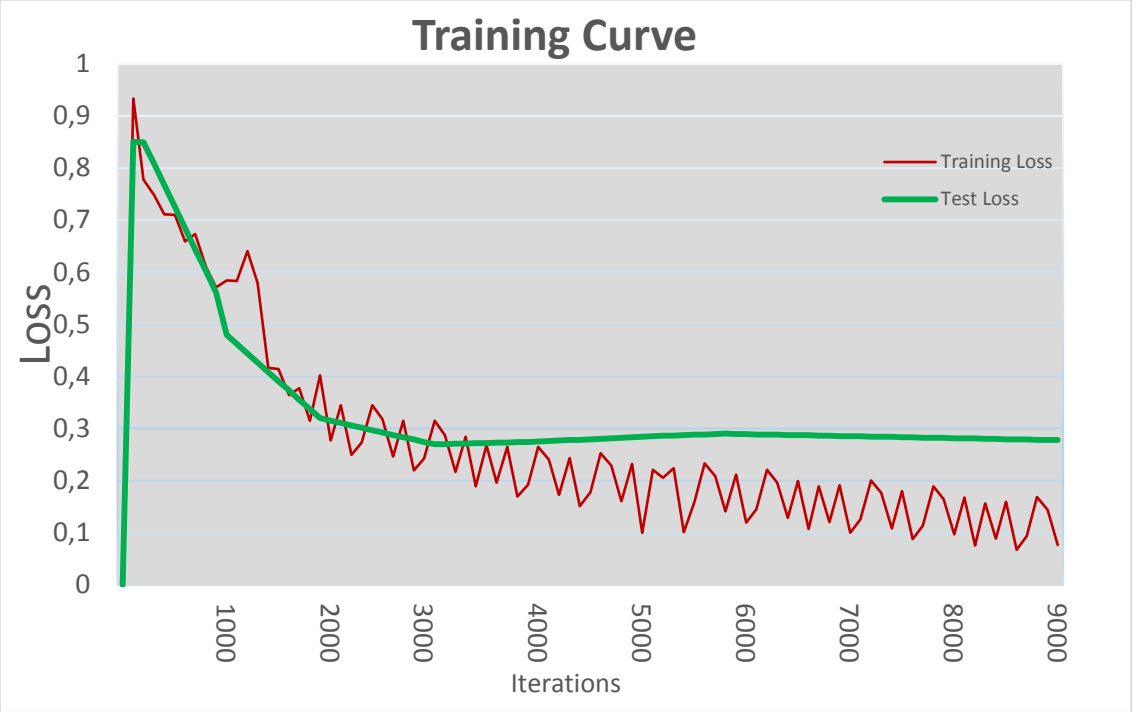
Data-data yang diperoleh dari hasil perolehan gambar akan digunakan sebagai data *input* yang akan digunakan dalam training model AI.

4.2 *Training* model AI

Sebelum dilakukan training pada model AI, sistem pendeteksi plat nomor dibuat terlebih dahulu di Python dan konfigurasi YOLO disiapkan. *Training* CNN dilakukan pada 8 direktori dan, 1202 data *training* dan test di google colab. Sebelum dilakukan *training,* dilakukan *pre-processing* pada *dataset train-test split* di google colab mengikuti konfigurasi YOLO V3 pada lampiran, *dataset* ini akan dirubah ukurannya menjadi 416 x 416 piksel, dan perubahan nilai HSV sebagai berikut :

- Nilai Hue 0,1
- Saturation 1,5
- Exposure 1,5

berdasarkan data tersebut diperoleh hasil kurva *training* sebagaimana ditunjukan pada Gambar 4.4



Gambar 4.4 Grafik Hasil *Training*

Berdasarkan Gambar 4.4 menjelaskan terkait hasil grafik dari proses *training.* Diuji nilai *loss* terhadap iterasi untuk data *train* dan data *test.* Pada rentang nilai 0-1.000 nilai *loss* pada *training* diperoleh masih tinggi yaitu sebesar 0,98. Namun seiring dengan pertambahan proses pembelajaran nilai *loss* menurun, sehubungan dengan hal berikut sebagaimana ditunjukan pada Gambar 3.1, maka tidak di perlukan *tune hyperparameters* dan bisa dilanjutkan pada *step* selanjutnya yaitu penyimpanan *weight* hasil *training,* berdasarkan Gambar 4.4 perolehan hasil *training* hingga pada iterasi ke 9.000 memiliki nilai *loss* yang semakin menurun hingga pada pada rentang 0,1. Namun terdapat *gap* pada kurva *training loss* dan *test loss* pada rentang iterasi 5.000 hingga 9.000. Iterasi yang optimal diperoleh dalam rentang antara iterasi ke 3.000 sampai dengan iterasi ke 4.000, sedangkan iterasi yang dilakukan lebih dari 4.000 akan menghasilkan *gap* yang cenderung melebar antara *training loss* dan *test loss,* hal ini menunjukkan kondisi yang sudah *saturated* atau jenuh. Kondisi inilah yang

menyebabkan hasil deteksi tidak bisa maksimal, oleh karena itu diambil hasil *training* dimana *gap* antara kedua kurva tidak terlalu signifikan. berdasarkan hasil yang diperoleh dari Gambar 4.2 dilihat dari grafik, model terbaik akan ada pada hasil *training* ke 4.000 hingga iterasi ke 5.000. Hal ini akan dipastikan pada poin 4.3 terkait validasi model AI.

4.3 *Validasi* Model AI

Proses validasi model AI dijelaskan pada poin 3.8, *weight* perolehan *training* diuji menggunakan hasil perolehan *dataset* sebagaimana dijelaskan pada poin 3.7. Berdasarkan penjelasan pada poin 4.2 dijelaskan mengenai *training* data sampai dengan 9.000 iterasi, dan 1.202 *Data Training-test* pada google colab untuk semua jenis plat nomor yang 4 berada di Indonesia yaitu :

- Plat Nomor Hitam
- Plat Nomor Kuning
- Plat Nomor Putih
- Plat Nomor Merah

Model AI yang sudah ditraining menjadi model AI yang pintar yang mampu mengenali data-data serupa, untuk menguji kemampuan model AI tersebut dilakukan uji validasi dengan menggunakan data yang berbeda atau data yang tidak termasuk dalam data *input* training. Data yang digunakan untuk memvalidasi sebayak 68 data plat nomor yang dilakukan dengan menggunakan iterasi dari 1.000 sampai dengan 9.000.

Data validasi ini diperoleh dari hasil pengambilan gambar plat nomor pada daerah Kawasan Metro di Pelataran parkir Metro Indah Mall di Kota Bandung pada pukul 12.00 -12.45 siang memiliki nilai Lumen sebesar 101.783 *lux* – 117.785 *lux*. Pengambilan data validasi dilakukan menggunakan kamera *handphone* Galaxy A9 Pro dengan spesifikasi dari kamera sebagai berikut :

1. Nomor model SM-A910F/DS
2. Nomor Seri RR8HC03L6EK

Gambar yang diambil dengan kamera *handphone* Galaxy A9 Pro yang digunakan untuk pengujian memiliki spesifikasi teknis sebagai berikut :

1. Size = 5,07 Mb
2. Focal Length=3,70mm

3. Resolution = 4608 x3456
4. ISO=40
5. Aperture =f1.9
6. Exp.time=1/549

Hasil validasi dari model AI yang pintar dengan menggunakan 68 data validasi, diperoleh hasil perolehan akurasi seperti pada tabel 4.2.

Tabel 4.2 Perolehan Akurasi

Iterasi Ke-	**berhasil dikenali**	**gagal dikenali**
1.000	63	5
	92,64%	7,36%
2.000	45	23
	66,17%	33,83%
3.000	57	11
	88,82%	16,17%
4.000	67	1
	98,52%	1,48%
5.000	63	5
	92,64%	7,36%
6.000	59	9
	86,74%	13,23%
7.000	65	3
	95,58%	4,42%
8.000	60	8
	88,23%	11,76%
9.000	66	2
	97,05%	2,94%

Berdasarkan Tabel 4.2 dapat dilihat bahwa nilai akurasi terbaik pada nilai iterasi ke 4.000 dengan jumlah berhasil dikenali sebesar 98,52 % , dengan pengujian model pada *weight* di iterasi ke 4.000 di dapatkan bahwa model bisa mengenali 67 gambar dari 68 total keseluruhan gambar. dan nilai akurasi terendah diperoleh pada iterasi ke 2.000 dengan tingkat pengenalan adalah 66,17%, dengan total pembacaan benar sebanyak 45 gambar dari total 68 gambar validasi. didapatkan hasil kecepatan pembacaan rata-rata beda pada 3,03 detik. Berdasarkan hasil uji validasi di atas, iterasi yang paling baik dengan kemampuan gagal dikenali paling kecil diperoleh pada iterasi ke 4.000.

```
Error: Different [yolo] layers have different number of classes = 80 and 1 - check your cfg-file!
Label file name is too short:
Can't open label file. (This can be normal only if you use MSCOCO):

detections_count = 0, unique_truth_count = 0
class_id = 0, name = plates, ap = 0.00%        (TP = 0, FP = 0)

for thresh = 0.25, precision = -nan, recall = -nan, F1-score = -nan
for thresh = 0.25, TP = 0, FP = 0, FN = 0, average IoU = 0.00 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.000000, or 0.00 %
Total Detection Time: 10.000000 Seconds
```

Gambar 4.5 Hasil perhitungan mAP (*Mean Average Precison)*

Dilakukan proses evaluasi lainnya menggunakan metode perhitungan mAP (*Mean Average Precision*) terhadap model *weight* ke-4.000, pada Gambar 4.5 didapatkan hasil pencarian mAP menggunakan fungsi yang berada pada (Alexey, 2020) Berdasarkan aplikasi fungsi yang telah diterapkan diperoleh hasil sebagaimana berikut :"*cant open label* file *(This can be normal if you use MSCOCO)"*. Dimana nilai mAP tidak berhasil untuk ditampilkan dikarenakan perbedaan penggunaan *dataset*, nilai mAP hanya bisa ditampilkan jika menggunakan *dataset* dari MS COCO (*Microsoft Common Object in Context).*

Nilai dari *precision* dan *recall* dari model terbaik berdasarkan (Shen, 2020), dapat diketahui dengan melakukan perhitungan. Dalam percobaan, FN digunakan untuk mewakili *false negative*, yang mewakili jumlah plat mobil yang diberi label tetapi tidak diprediksi. FP digunakan untuk mewakili *false positive*, yang merupakan singkatan dari jumlah plat mobil yang diprediksi oleh model tetapi tidak diberi label, TP digunakan untuk mewakili *true positive* menunjukkan bahwa prediksi tersebut benar, hal ini berarti bahwa jumlah plat mobil diprediksi benar oleh model. Hasil perhitungan dari *precision* dan *recall* dari model adalah 100% dan 98,52%.

Perbandingan hasil pengenalan plat nomor menggunakan YOLO V3 dengan beberapa *Author* pada (Pinto, 2019) memiliki hasil sejumlah tingkat *precision* dan *recall* sebesar 96% dan 95% dengan jumlah total *dataset* yang di sediakan oleh LAMSA (*Linha Amarela road in Rio de Janeiro)* sebanyak 1.421.774 buah pada penelitian (Shen, 2020) juga melakukan pendeteksian plat mobil menggunakan

YOLO V3 dengan total jumlah gambar yang di *training* dan di *test*ing sebesar 5.500 dan 1.168 dari *database* yang di dapatkan dari Caltech (Caltech, 2001), Zemris (Zemris 2003), Azam (Azam S, 2016) Aolpe (Hsu G S, 2017) mendapatkan nilai *precision* dan *recall* sebesar 90,75% dan 86,94%. Nilai akurasi dari sistem yang di buat sudah melebihi dari hasil penelitian sebelumnya (98,52%) begitupula nilai dari *precision* (100%) dan *recall* (98,52%), namun tentunya hal ini di pengaruhi dari volume *dataset* yang digunakan dalam pembuatan sistem. diharapkan pada kemudian hari Indonesia memiliki suatu sistem yang menyediakan *dataset* plat nomor agar bisa dilakukan pengembangan lebih lanjut terhadap keilmuan pengenalan plat nomor bagi kendaraan bermotor. Pada tabel 4.3 dijelaskan terkait perbandingan hasil dari nilai *precision* dan *recall* yang sudah didapat.

Tabel 4.3 Hasil perbandingan nilai *Precision* dan *Recall*

	YOLO-Farah	**Shen**	**Pinto**	**Aolpe**
Precision	100%	100%	96%	90,75%
Recall	98,52%	98,52%	95%	86,94%

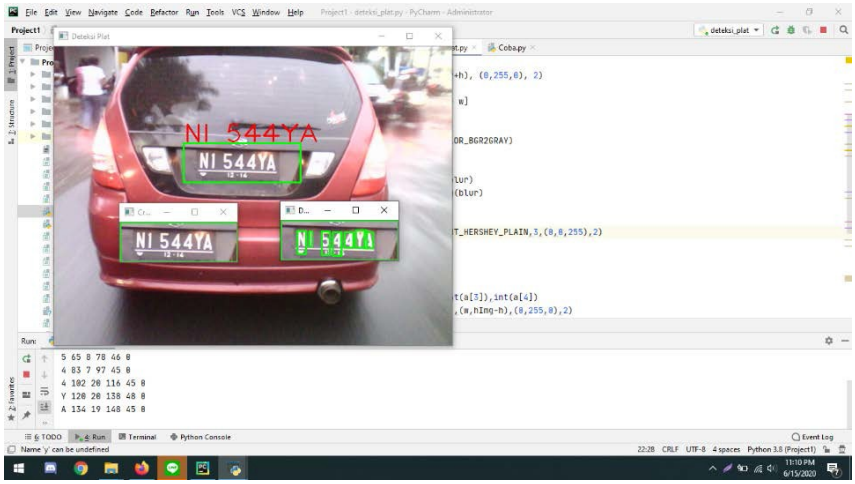
4.4 Hasil OCR menggunakan Tesseract

Hasil prediksi menggunakan *Software Tesseract* yang digunakan di Python dengan IDE Spyder dengan kodingan yang berada pada lampiran, dilakukan suatu *pre-processing* terlebih dahulu, *pre-processing* yang terbaik untuk ini adalah dengan penggunaan RGB2GRAY dan *effect median blur.* Sebelumnya dilakukan terlebih dahulu *thresholding* pada gambar namun hasil yang didapatkan tidak maksimal, hasil optimal didapatkan dari *pre-processin*g di perubahan gambar RGB menjadi *Gray* lalu dilanjutkan dengan penambahan *effect blur*. Setelah dilakukan *pre-processing* dilanjutkan dengan pendeteksian karakter alphanumerik sebanyak 68 gambar. Pada pengujian ini setiap gambar yang menjadi *input* akan di *cropping* pada bagian plat yang terdeteksi oleh software Tesseract, dan segmentasi per-karakter. Hasil deteksi dapat dilihat pada tabel 4.4.

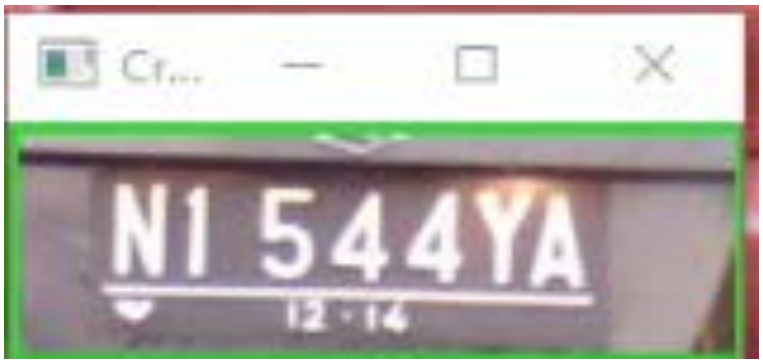
Tabel 4.4 Hasil Deteksi dari OCR

Nomor	JENIS	JUMLAH BENAR	JUMLAH SALAH	PERSENTASE
1	*Cropping*	68	0	100%
2	Segmentasi Karakter	16	52	30,76%
3	Deteksi perkarakter	12	56	21,42%

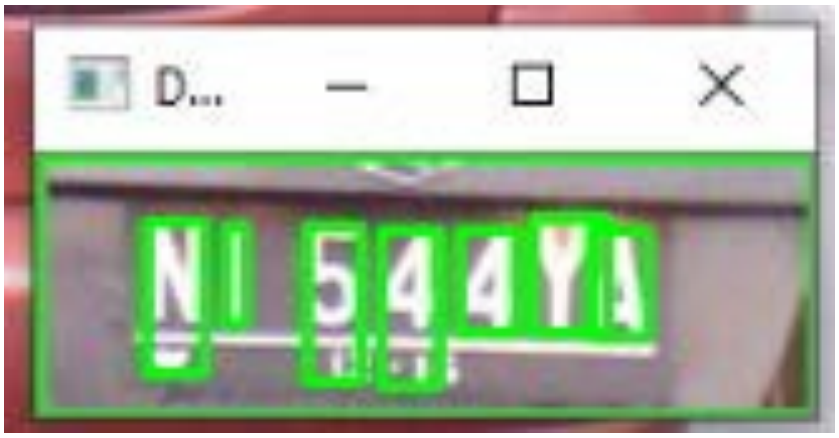
Hasil deteksi pada 68 *Dataset* dan tingkat keterkenalan menggunakan *Tesseract* adalah ± 20% dari total jumlah *dataset* yang bisa mengenali seluruh karakter pada plat mobil, yang berupa karakte*r* Alphanumerik sebanyak 6-7 data. Pada sistem yang digunakan berhasil mengenali gambar sebanyak 100%, membuat *cropping* berdasarkan plat sebesar 100%, hasil segmentasi karakter sebesar 30,76%. Pada Gambar 4.6 sampai Gambar 4.8 menjelaskan proses yang dilakukan oleh *Tesseract* agar bisa mengenali karakter pada plat nomor. *Tesseract* yang dipakai untuk melakukan tahap ini adalah *Tesseract* 4.11 yang merupakan versi terbaru. *Tesseract* merupakan *open source software* yang menyediakan lebih dari pengenalan 100+ *library* untuk pengenalan karakter dan bahasa, hal ini yang menyebabkan *Tesseract* unggul dalam perihal kemampuan pendeteksian, namun kegagalan yang terjadi pada pendeteksian *Tesseract* bisa disebabkan oleh pengaruh *sizing, contrast,* resolusi pada gambar dan ketersediaan *library*, hal ini disebabkan karena jenis teks yang digunakan untuk plat nomor di Indonesia masih belum masuk pada *library Tesseract*, sehingga hal tersebut menyebabkan *Tesseract* bisa mendeteksi suatu teks namun dengan tingkat keberhasilan yang rendah. Diharapkan dengan adanya suatu *update* terbaru dari *Tesseract* yang sudah memiliki karakteristik font plat nomor yang berada di Indonesia akan mampu mengenali plat nomor di Indonesia dengan tingkat kesalahan yang kecil.



Gambar 4.5 Hasil Deteksi menggunakan *Tesseract*



Gambar 4.6 Hasil *Cropping* pada plat nomor



Gambar 4.7 Hasil Segmentasi karakter Alphanumerik di Plat Nomor

4.5 Analisa Kelemahan Deteksi OCR

Software Tesseract yang digunakan dalam melakukan deteksi OCR ini dilakukan secara *plug and play* yang menggunakan program *Tesseract* OCR seperti pada lampiran penelitian ini, akan tetapi program AI ini tidak dilakukan *training* sebelumnya, sehingga program AI ini masih belum pintar. Berdasarkan hasil pengujian didapatkan hasil kegagalan pendeteksian karakter alphanumerik sebesar 80,40%. Sebelum dilakukan pengujian deteksi OCR dilakukan *pre-processing* terlebih dahulu. Tahapan *pre-processing* yang dilakukan adalah perubahan gambar yang berwarna (RGB) menjadi *gray* dan penggunaan *effect median blur*. Sebelumnya telah dicoba beberapa efek *blurring* seperti *median blur, gaussian blur* dan *bilateral blur* namun setelah dilakukan *trial dan error* ini didapatkan hasil yang paling optimal yaitu *effect median blur*. Dengan penggunaan *effect median blur* ini, didapatkan sistem OCR mampu mengenali gambar secara keseluruhan, mampu melakukan pembuatan *cropping*, serta segmentasi per karakter. Beberapa kegagalan dalam pembacaan OCR di *Tesseract* disebabkan oleh beberapa faktor antara lain:

- *Size* dari citra yang tidak seragam atau tidak sesuai,
- Nilai kontras yang terlalu tinggi atau terlalu rendah,
- Pencahayaan yang terlalu tinggi atau terlalu rendah
- ketersediaan *library* dari *Tesseract* yang masih belum memadai untuk pendeteksian karakter.

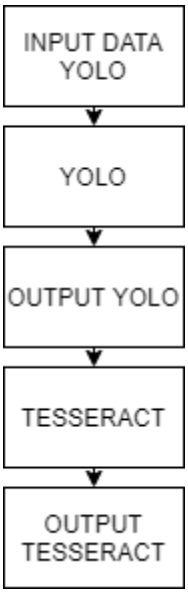
Hal ini di perkuat dikarenakan ketika proses OCR yang di lakukan *Tesseract* tergantung dengan ketersediaan *library* yang di miliki. Ketika ada suatu teks atau karakter yang gagal dikenali oleh sistem *Tesseract* akan diteruskan kepada *adaptive classifier* untuk agar menjadi suatu data pelatihan. Ketika *adaptive classifier* memiliki sampel yang cukup, *adaptive classifier* ini dapat memberikan hasil klasifikasi bahkan pada *pass* pertama untuk mendeteksi plat dari *Tesseract*.

Gambar 4.8 Kesalahan Pendeteksi pada OCR

Gambar 4.9 Kesalahan Pendeteksian dalam variabel tempat pada OCR

Gambar 4.10 Kegagalan dalam pendeteksian

Pada Gambar 4.9 terlihat bahwa *Tesseract* tidak mampu untuk mengenali karakter pada plat nomor.



Gambar 4.11 Proses yang terjadi di dalam penelitian

Pada Gambar 4.11 ditunjukan proses yang terjadi pada penelitian tugas akhir ini, dimulai dengan proses *input* data, proses pendeteksian objek menggunakan YOLO V3 yang menghasilkan *output* berupa gambar yang sudah teridentifikasi plat nomor. Berdasarkan hasil keluaran yang diperoleh dari proses deteksi objek oleh YOLO akan dimasukkan sebagai *input* pada proses OCR yang menggunakan *Tesseract*. *Output* yang dihasilkan dari *Tesseract* untuk mendeteksi plat nomor tersebut hasilnya masih belum memberikan yang bagus atau berbeda jauh bila dibandingkan dengan *output* YOLO yang bisa mendeteksi obyek dengan tingkat akurasi yang tinggi. *Output* dari YOLO memberikan tingkat ketelitian yang tinggi karena ada proses *training* yang dilakukan dengan iterasi sampai dengan 9.000 kali, sehingga model AI pada YOLO sudah pintar dan memiliki pengetahuan yang lengkap sehingga model AI mampu mengenali obyek baru dengan tingkat akurasi yang tinggi. *Software* AI yang digunakan pada *Tesseract* diambil dari publik atau Google *open source* (Google, 2019).

Dalam penelitian ini, ada dua *software* AI yang digunakan yaitu YOLO V3 dan *Tesseract* 4.11, dengan penjelas dan sebagai berikut :

a. YOLO V3, dengan data *input* yang digunakan untuk melatih *software* AI – YOLO V3, mampu untuk meningkatkan kemampuan *software* AI menjadi pintar atau memiliki kapabilitas untuk belajar. Hal itu ditunjukkan dalam hasil

pengetesan terhadap plat nomor yang bisa dikenali sampai 98.52%. Hal ini menunjukkankan bahwa *Software* AI YOLO V3, terbukti menjadi *software* yang pintar dan memiliki kemampuan untuk belajar.

b. *Tesseract* versi 4.11. *Software* yang digunakan dalam tugas akhir ini tidak dikembangkan dari awal seperti yang dilakukan pada *software* AI YOLO V3 akan tetapi dilakukan dengan menggunakan *software* AI yang sudah jadi, ada di publik dan merupakan versi terbaru. Akan tetapi *software* AI tersebut saat digunakan masih belum tahu apakah cukup pintar untuk mengenali karakter plat nomor di Indonesia. Setelah digunakan, ternyata memiliki kelemahan, dengan deteksi yang rendah. Untuk meminimalisir kelemahan ini dengan melakukan suatu upaya untuk memperbaiki kelemahan deteksi OCR dapat dilakukan dengan Penambahan *library* plat nomor Indonesia pada *Tesseract*. Upaya penambahan *library* bisa dilakukan oleh :
    - Google
    - Developer, yang memiliki kemampuan untuk mengkases program aplikasi *Tesseract* dan tertarik untuk menambah *library* plat nomor Indonesia.

Halaman ini sengaja dikosongkan

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisa data yang telah dilakukan, dapat diambil beberapa kesimpulan dari studi mengenai perancangan sistem pengenalan plat nomor otomatis menggunakan metode YOLO V3 dengan *framework darknet* adalah sebagai berikut :

1. Sistem deteksi plat nomor otomatis yang menggunakan algoritma YOLO v3 dengan *framework darknet,* diperoleh hasil terbaik pada Iterasi ke 4.000 dengan prosentase plat nomor yang berhasil dikenali sebesar 98,52% dari total keseluruhan plat nomor sedangkan nilai akurasi terendah diperoleh pada iterasi ke 2.000 dengan prosentase sebesar 66,17%, dimana hasil pembacaan rata-rata sebesar 3.03 detik.
2. Prediksi Alphanumerik menggunakan *software Tesseract* yang menggunakan data sebanyak 68 *dataset* plat nomor diperoleh hasil prosentase deteksi dan tingkat keterkenalan sebesar ± 20% dari total jumlah *dataset* untuk mengenali seluruh karakter pada plat mobil, yang berupa karakter Alphanumerik sebanyak 6-7 karakter.

5.2 Saran

Adapun saran yang dapat diberikan berdassarkan hasil penelitian tugas akhir ini adalah sebagai berikut:

1. Performansi sistem pendeteksi ini dapat ditingkatkan dengan menambah volume *dataset* yang akan digunakan dalam training model AI oleh karena itu diharapkan agar dibuatnya suatu *database* yang komprehensif dan disimpan di *GitHub* untuk studi kasus pengenalan plat nomor di Indonesia, baik dalam konteks *Object detection* ataupun *optical character recognition.* Hal ini di perlukan agar meminimalisir waktu proses anotasi dan pengumpulan *dataset*.
2. Dibuatnya suatu sistem tersendiri yang mampu untuk mengenali jenis plat nomor di Indonesia, baik dari segi klasifikasi kendaraan dan warna plat nomor

DAFTAR PUSTAKA

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network*, " 2017 International Conference on Engineering and Technology (ICET)*, Antalya, pp 1-6.

Alexey, A. (2016). Windows and Linux version of Darknet YOLO v3 and v2 Neural networks for object detection. Retrieved from <https://github.com/AlexeyAB/*darknet*>

Azam S, I. M. (2016). Automatic license plate detection in hazardous condition. Journal of Visual Communication and Image Representation, 36: 172-186.

Caltech. (2001). Retrieved from Caltech: Caltech License Plate Dataset

Dewantoro, N. P. (2020). YOLO Algorithm Accuracy Analysis in Detecting Amount of Vehicles at the Intersection. The 3rd International Conference on Eco Engineering Development (pp. doi:10.1088/1755- 1315/426/1/012164). IOP Publishing, IOP Conf. Series: Earth and Environmental Science.

Du, J. (2018). Understanding of Object detection Based on CNN Family a. IOP Conf. Series: Journal of Physics: Conf. Series 1004 (2018) 012029 doi :10.1088, 1742-6596.

Febriandita, L. (2016). impelementasi Metode Histograms of Oriented Gradients dengan Optimasi Algoritma Frei-Chen untuk Deteksi Citra Manusia. Bandung: e-*library* UNIKOM.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 193-202.

Gazali, W. (2012). Penerapan Metode Konvolusi Dalam Pengolahan Citra Digital. Jurnal Mat Stat, 103-113.

Girshick R., D. J., & Malik, J. R. (2014). Feature Hierarchies for Accurate Object detection and SemanticSegmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern recognition, pp. 580–587.

Goodfellow, I. B. (2016). Deep Learning. Cambridge, Massachusetts: MIT Press.

Hasan, N. A. (2015). Optical Character Recognition (OCR) System. *IOSR Journal of Computer Engineering (IOSR-JCE)*, e-ISSN: 2278-0661, p-ISSN: 2278- 8727, Volume 17, Issue 2, Ver. II.

Han, Q. G. (2012). Study on Web Mining Algorithm based on Usage Mining. 2008 9th *International Conference on Computer-Aided Industrial Design and Conceptual Design*, 1121-1124.

He, K. Z. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual *Recognition*. *IEEE Trans. Pattern Anal. Mach. Intell*, *37*(9), 1904 - 1916.

Hendry, H. C.-C. (2019). Automatic License Plate *Recognition* via sliding-window *darknet*-YOLO *deep learning*. *Image and Vision Computing*, *87*, 47-56.

Hsu G S, A. A. (2017). Robust license plate *detection* in the wild. *14th IEEE International Conference on Advanced Video and Signal Based* (pp. pp. 1- 6.). IEEE .

Huang, Y.-Q. J.-C.-D.-F. (2020). Optimized YOLOv3 Algorithm and Its Application in Traffic Flow Detections. *Applied Science :Multidisciplinary Digital Publishing Institute*, Appl. Sci. 2020, 10, 3079. doi:app10093079

J. Redmon, S. D. (2016). "You Only Look Once: Unified, Real-Time Object detection". *IEEE Conference on Computer Vision and Pattern recognition (CVPR),* , pp. 779-788, doi: 10.1109/CVPR.2016.91.

Jalled, F. (2016, November 23). Object detection Using Image processing, . *ArXiv: Computer Science and Pattern recognition*, p. abs/1611.07791.

Jamstho, Y. R. (2020). Real-time Bhutanese license plate localization using YOLO. *ICT Express*, *6*(2), 121-124.

Kaiming He, X. Z. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual *Recognition*. *IEEE Trans. Pattern Anal. Mach. Intell*.

Krizhevsky, A. S. (2012). Imagenet classification with deep convolutional neural network. In advances in neural information processing systems. *Proceeding of 25th Neural Information Processing Systems Conference* (pp. 1097- 1105). Nevada, United States of America: Curran Associates, Inc.

Kwanggi, K. (2016). Book Review: Deep learning. *Healthcare Informatics Research* (p. 351). Seoul, South Korea: Healthcare Informatics Research. doi:10.4258/hir.2016.22.4.351

Le Cun, Y. (2015). Deep learning. *Nature*(7553), 436-44.

Le Cun, Y. B. (1990). Handwritten Digit Recognition with a Backpropagation Network, ". *Proceeding of 2nd Neural Information Processing Systems* (pp. 396-404). Denver, Colorado: Morgan-Kaufmann.

Liu, J. (2020). Research on Vehicle Object detection Algorithm Based on. *Journal of Physics: Conference Series*, doi:10.1088/1742-6596/1575/1/012150.

Liu, T. d. (2015). Implementation of Training Convolutional Neural network. Beijing.

Lukas, S. Y. (2013). Identification of Indonesian Vehicle Registration Plate by Adaptive Thresholding and Region. *2013 Eleventh International Conference on ICT and Knowledge Engineering* , 1-4.

Munir, R. (2004). Pengolahan citra digital dengan pendekatan algoritmik. In R. Munir, *Pengolahan citra digital dengan pendekatan algoritmik.* Bandung: Informatika .

Nilsson, N. J. (1998). introduction of machine learning . In N. J. Nilsson, *introduction of machine learning* (p. 176). USA: Robotics Laboratory, department of computer science, Stanford University.

Lukas, S. Y. (2013*)*. Identification of Indonesian Vehicle Registration Plate by Adaptive Thresholding and Region. *2013 Eleventh International Conference on ICT and Knowledge Engineering* , 1-4.

Pinto, P. A. (2019). A YOLOv3-based Brazilian Automatic License Plate Recognition Tool. *Anais Estendidos do XXV Simpósio Brasileiro de SistemasMultimídiaeWeb,* (p.DOI:10.5753/webmedia_estendido.2019.8149). Brazil.

Peraturan Kepala Kepolisian Republik Indonesia. (2012). Peraturan Kepala Kepolisian Republik Indonesia Nomor 5 tahun 2012 tentang registrasi dan identifikasi kendaraan bermotor. Markas Besar Polisi Repulik Indonesia

Prasantha, S. H. (2013). Fast Computation of Image Scaling Algorithms Using Frequency Domain Approach. *Advances in Intelligent Systems and Computing*, 201-208

Qiu, D. L. (2010). Application of graphic minimal covering algorithm in the distribution of surveillance cameras in small and medium-sized city road networks. *International Conference On Computer Design and Applications* , vol 1 pp V1–200–V1–203.

Raden Sofian Bahri, I. M. (2013). Perbandingan Algoritma Template Matching dan Feature Extraction Pada Optical Character Recognition*,* . *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)*, Edisi I Volume 1, Bandung.

Redmon, J. D. (2016). You Only Look Once: Unified, Real-Time Object detection. *IEEE Conference on Computer Vision and Pattern recognition (CVPR)*, (pp. pp. 779-788). Las Vegas.

Shaikh, S. A. (2020). Object detection and Tracking using YOLO v3 *Framework* for Increased Resolution Video. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, ISSN: 2278-3075, Volume-9 Issue-6.

Shen, S. *,* (2020). Car Plate Detection Based on YOLOv3. *International Conference on Signal Processing* (pp. 1742-6596). Beijing: IOP Publishing. doi:doi:10.1088

Shreyas, R. K. (2017). Dynamic traffic rule violation monitoring system using automatic number plate recognition with sms feedback. *2nd International Conference on Telecommunication and Networks (TEL-NET),* , 1-5. doi:10.1109/TEL-NET.2017.8343528.

Smith, R. (2007). An Overview of the *Tesseract* OCR Engine. *ICDAR (International Conference in Document Analysis and Recognition) Volume 2*, DOI: 10.1109/ICDAR.2007.4376991.

Stathakis, D. (2009). How Many Hidden *Layer*s And Nodes. *International Journal of Remote Sensing 30*, *30*, 2133-2147. doi:01431160802549278

Suartika, I. W. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101. *JURNAL TEKNIK ITS* , 2301-9271

Sun, H. F. (2019). License Plate Detection and Recognition Based on the YOLO Detector and CRNN-12. In S. Sun (Ed.), *The 4th International Conference on Signal and Information Processing*, *Networking and Computers (ICSINC)* (pp. 66-74). Qingdao, China: Springer Nature Singapore.

Sun, P. (2019). Yi Characters Recognition Based on *Tesseract*-OCR*, . IEEE 3rd Advanced Information Management*, *Communicates*, *Electronic and Automation Control Conference (IMCEC)*.pp. 102-106, doi: 10.1109/IMCEC46724.2019.8983913.

Thakar, V. S. (2018). Efficient Single-Shot Multibox Detector for Construction Site Monitoring. *IEEE International Smart Cities Conference (ISC2)*, (pp. pp. 1- 6). Kansas City.

Tianyi Liu, S. (2015). *Implementation of Training Convolutional Neural networks.* Retrieved from <https://arxiv.org/abs/1506.01195: https://arxiv.org/abs/1506.01195>

Undang Undang Republik Indonesia. (2009). Undang Undang Republik Indonesia nomor 22 tahun 2009 tentang lalu lintas dan angkutan umum. Dewan Perwakilan Rakyat Republik Indonesia.

V. Thakar, H. S. (2018). "Efficient Single-Shot Multibox Detector for Construction Site Monitoring*, "*. *IEEE International Smart Cities Conference (ISC2)*, pp. 1-6, doi: 10.1109/ISC2.2018.8656929.

Villman, T. K. (2014). Precision-Recall-Optimization in Learning Vector Quantization Classifiers for Improved Medical Classification Systems. Orlando: *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*.

Xiang-wei, L. Y.-f. (2012). A Data Preprocessing Algorithm for Classification Model Based On Rough Sets. *Physics Procedia*, 2025-2029.

Yahya, s. (2012). Fuzzy Logic, Neural network, Genetic Algorithm & Knowledge Based Expert System and Computational Intelligence. Bandung: Politeknik Negeri Bandung.

Yuli, S. H. (2012). Indonesian Vehicle Plate Recognition and Identification Based on Digital Image processing and Artificial Neural network. *International*

Conference on Soft Computing, Intelligent System and Information Technology, (p. DOI: 10.13140/2.1.1917.6481).

Yusuf, T. (2016). Kelebihan dan Kekurangan ANN (Artificial Neural network.Bandung: Politeknik Negeri Bandung.

Zemris:.(2003).Retrievedfrom<http://www.zemris.fer.hr/projects/LicensePlates/hrvatski/rezultati.shtml>

LAMPIRAN

A. Kode Python untuk menguji deteksi objek YOLO

```
1. import numpy as np
2. import cv2
3. import os 4.
5. #image_name    = "HITAM2.jpg" 6.
7. model_directory = "data test 9.000"
8. model_name   = "YOLOv3_9.000.weights"
9. cfg_name       = "YOLOv3.cfg"
10. label_names   =       "label.names"
11. confidence_val        =       0
12. thresh_val      =       0
13.
14. LABELS = open(os.path.join(model_directory,
label_names)).read().strip().split("\n")
15. np.random.seed(42)
16. COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
dtype="uint8")
17. font = cv2.FONT_HERSHEY_SIMPLEX 18.
19. weightsPath = os.path.sep.join([ model_directory, model_name])
20. configPath = os.path.sep.join([ model_directory, cfg_name])
21. 21.
22. print("[INFO] loading YOLO from disk...")
23. net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
24. ln = net.getLayerNames()
25. ln = [ ln[i[0] - 1] for i in net.getUnconnectedOutLayers() ]
26. 26.
27. 27.      #frame = cv2.imread(os.path.join("data-test", image_name))
28. 28.
29. for image in os.listdir('data-test'):
```

30. frame = cv2.imread(os.path.join("data-test", *image*))
31. 31.
32. 32. (H, W) = frame.shape[:2] 33.
33. *blob* = cv2.dnn.*blob*From*Image*(frame, 1 / 255.0, (416, 416), swapRB=True, crop=False)
34. net.set*Input*(*blob*)
35. *layerOutput*s = net.forward(ln) 37.
36. boxes = []
37. confidences = []
38. *class*IDs = [] 41.
39. for *output* in *layerOutput*s:
40. for *detection* in *output*:
41. scores = *detection*[5:]
42. *class*ID = np.argmax(scores)
43. confidence = scores[*class*ID]
44. if confidence > confidence_val:
45. box = *detection*[0:4] * np.array([W, H, W, H])
46. (centerX, centerY, *width*, *height*)
47. = box.astype("int")
48. x = int(centerX - (*width* / 2))
49. y = int(centerY - (*height* / 2)) 52.
50. 53. boxes.append([x, y, int(*width*), int(*height*)])
51. 54.
52. confidences.append(float(confidence))
53. 55. *class*IDs.append(*class*ID) 56.
54. idxs = cv2.dnn.NMSBoxes(boxes, confidences, confidence_val, thresh_val)
55. if len(idxs) > 0:
56. for i in idxs.flatten():
57. 60. (x, y) = (boxes[i][0], boxes[i][1])
58. 61. (w, h) = (boxes[i][2], boxes[i][3])
59. color = [int(c) for c in COLORS[*class*IDs[i]]]

60. frame = cv2.rectangle(frame, (x, y), (x+w, y+h), color)
61. text = f"{LABELS[*class*IDs[i]]} :
62. {confidences[i]}"
63. frame = cv2.putText(frame, text, (x, y-3), *font,*
64. 66. 0.5, color, 2,
65. cv2.LINE_AA) 67.
66. 68.
67. 69.
68. # Display the resulting frame
69. cv2.imshow('frame', frame)
70. cv2.waitKey(0)
71. cv2.destroyAllWindows()

B. Kode Python untuk menguji OCR

1. **i**mport cv2
2. import py*Tesseract*
3. ### Deteksi Plat ### faceCascade =
4. cv2.Cascade*Classifier*("haarcascade_plate_number.xml") img = cv2.imread("lah.png")
5. imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
6. face = faceCascade.detectMultiScale(imgGray, 1.1, 2) for (x, y, w, h) in face:
7. cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2) cropImg = img.copy()
8. cropImg = cropImg[y:y + h, x:x + w]
9. ### Baca Karakter ###
10. gray = cv2.cvtColor(cropImg, cv2.COLOR_BGR2GRAY) blur = cv2.medianBlur(gray, 1)
11. boxes = py*Tesseract.image*_to_boxes(blur) noPlat = py*Tesseract.image*_to_string(blur) print(noPlat)
12. print(boxes) cv2.putText(img, noPlat, (x, y), cv2.*FONT*_HERSHEY_PLAIN, 3, (0, 0, 2 55), 2)
13. hImg, wImg, _ = cropImg.shape charImg = cropImg.copy()
14. for a in boxes.splitlines(): a = a.split(' ')
15. x, y, w, h = int(a[1]), int(a[2]), int(a[3]), int(a[4])

16. cv2.rectangle(charImg, (x, hImg-y), (w, hImg-h), (0, 255, 0), 2)

17. cv2.imshow("Deteksi Plat", img) cv2.imshow("Crop", cropImg)
    cv2.imshow("Deteksi Karakter", charImg) cv2.waitKey(0)

BIODATA PENULIS

Penulis bernama Farah Qoonita Syuhaila yang lahir di Bandung 13 Juni 1998. Penulis telah menempuh pendidikan formal di TK Islam Al-Hambra yang lulus pada tahun 2004, SDIT Al-Biruni Bandung yang lulus pada tahun 2010, SMP Nurul Fikri *Boarding School* Lembang yang lulus pada tahun 2013, dan SMA Nurul Fikri *Boarding School* Lembang yang lulus pada tahun 2016 dan kemudian melanjutkan studinya ke jenjang sarjana di Departemen Teknik Fisika, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama menjadi mahasiswa, penulis aktif dalam berbagai kegiatan organisasi dan kepanitiaan baik di dalam kampus maupun diluar kampus. Penulis pernah menjadibagi an dari ITS MUN Club sebagai Direktur Pemberdayaan *Talent* dan *Proffesionalism* pada tahun 2018-2019. Selain itu, penulis juga pernah menjabat sebagai Wakil Ketua di *The Optical Society Association* ITS *Student Chapter*, Asisten Laboratorium pada Laboratorium Rekayasa Fotonika di ITS. Pada tahun terakhir perkuliahan, penulis melakukan magang di GMF Aeroasia Tbk, Husky CNOOC Madura Ltd Penulis dapat dihubungi melalui email di hifarahqoonita@gmail.com