PENINGKATAN AKURASI PADA PENGENALAN PLAT NOMOR KENDARAAN RODA EMPAT DENGAN MENGGUNAKAN SUPER RESOLUTION GENERATIVE ADVERSARIAL NETWORK

Tugas Akhir

Sebagai syarat untuk memperoleh gelar sarjana S-1 di Program Studi Informatika,
Jurusan Informatika, Fakultas Teknik Industri,
Universitas Pembangunan Nasional "Veteran" Yogyakarta

Disusun Oleh
Nicholas Nanda Sulaksana
123180049

PROGRAM STUDI INFORMATIKA
JURUSAN INFORMATIKA
FAKULTAS TEKNIK INDUSTRI
UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"
YOGYAKARTA
2022

HALAMAN PENGESAHAN PEMBIMBING

PENINGKATAN AKURASI PADA PENGENALAN PLAT NOMOR KENDARAAN RODA EMPAT DENGAN MENGGUNAKAN SUPER RESOLUTION GENERATIVE ADVERSARIAL NETWORK

Disusun oleh :
Nicholas Nanda Sulaksana
123180049

Telah diuji dan dinyatakan lulus oleh pembimbing
pada tanggal : .....

Menyetujui,

Pembimbing I	Pembimbing II
Dr. Awang Hendrianto Pratomo, S.T., M.T.	Bambang Yuwono, S.T., M.T.
NIDN. 0025077701	NIDN. 0512027401

Mengetahui,
Ketua Program Studi

Dr. Heriyanto, A.Md., S.Kom., M.Cs.
NIDN. 0508067703

PENINGKATAN AKURASI PADA PENGENALAN PLAT NOMOR KENDARAAN RODA EMPAT DENGAN MENGGUNAKAN SUPER RESOLUTION GENERATIVE ADVERSARIAL NETWORK

Disusun oleh :

Nicholas Nanda Sulaksana
123180049

Telah diuji dan dinyatakan lulus oleh penguji
pada tanggal : .....

Menyetujui,

Penguji I	Penguji II
Dr. Awang Hendrianto Pratomo, S.T., M.T.	Bambang Yuwono, S.T., M.T.
NIDN. 0025077701	NIDN. 0512027401
Penguji III	Penguji IV
Dessyanto Boedi Prasetyo, S.T., M.T.	Dr. Herlina Jayadianti, S.T., M.T.
NIDN. 0505127501	NIDN. 0527087701

SURAT PERNYATAAN
KARYA ASLI TUGAS AKHIR

Sebagai mahasiswa Program Studi Informatika Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta, yang bertanda tangan dibawah ini, saya:

Nama : Nicholas Nanda Sulaksana

NIM : 123180049

Menyatakan bahwa karya ilmiah saya yang berjudul:

PENINGKATAN AKURASI PADA PENGENALAN PLAT NOMOR KENDARAAN RODA EMPAT DENGAN MENGGUNAKAN SUPER RESOLUTION GENERATIVE ADVERSARIAL NETWORK

Merupakan karya asli saya dan belum pernah dipublikasikan dimanapun. Apabila dikemudian hari, karya saya disinyalir bukan merupakan karya asli saya, maka saya bersedia menerima konsekuensi yang diberikan Program Studi Informatika Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenar-benarnya.

Dibuat di : Sleman, DIY

Pada Tanggal : 22 Desember 2022

ang menyatakan,

Nicholas Nanda Sulaksana

PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini:

Nama : Nicholas Nanda Sulaksana

NIM : 123180049

Dengan ini menyatakan bahwa judul Tugas Akhir

PENINGKATAN AKURASI PADA PENGENALAN PLAT NOMOR KENDARAAN RODA EMPAT DENGAN MENGGUNAKAN SUPER RESOLUTION GENERATIVE ADVERSARIAL NETWORK

Adalah hasil kerja saya sendiri dan benar bebas dari plagiasi kecuali cuplikan serta ringkasan yang terdapat di dalamnya telah saya jelaskan sumbernya (sitasi) dengan jelas. Apabila pernyataan ini terbukti tidak benar maka saya bersedia menerima sanksi sesuai peraturan Mendiknas RI No 17 Tahun 2010 dan Peraturan Perundang-undangan yang berlaku.

Demikian surat pernyataan ini saya buat dengan penuh tanggung jawab.

Sleman, 22 Desember 2022

Yang membuat pernyataan,

Nicholas Nanda Sulaksana

NIM. 123180049

ABSTRAK

Pertumbuhan kendaraan yang terus meningkat berdampak terhadap padanya kendaraan di jalanan, yang juga mengakibatkan bertambahnya kasus pelanggaran peraturan lalulintas. Sampai saat ini pendeteksian dan penindakan kasus pelanggaran secara konvensional dirasa masih kurang optimal. Oleh karena itu deteksi dan penindakan pelanggaran secara otomatis dirasa penting, dengan adanya proses *automatic license plate recognition* (ALPR) atau pengenalan plat nomor kendaraan otomatis dapat membantu dalam proses penindakan pelanggaran. Pada sisi lain proses ALPR memiliki kekurangan, yaitu kemampuan CCTV Analog dalam menangkap citra masih memiliki kualitas yang rendah dikarenakan banyaknya noise. Dengan begitu peningkatan kualitas citra untuk meningkatkan akurasi proses pengenalan karakter dirasa penting.

Pada penelitian ini dilakukan peningkatan kualitas citra pada proses license plate recognition untuk pengenalan plat nomor kendaraan roda empat dengan menggunakan *super-resolution generative adversarial network* (SRGAN). SRGAN di implementasikan setelah proses lokalisasi plat nomor, dengan menghasilkan citra *super-resolution* memiliki 4x lebih banyak jumlah pixel daripada citra input, yang kemudian citra *super-resolution* dilakukan proses pengenalan karakter.

Pengujian dilakukan dengan melakukan pengukuran pada hasil akurasi, *recall* dan F1 *score* maupun nilai PSNR dan SSIM pada hasil pengenalan karakter pada plat nomor kendaraan roda empat. Perhitungan dilakukan pada citra hasil lokalisasi YOLOv4 sebelum dan setelah dikenai SRGAN, dengan menghasilkan peningkatan akurasi pada proses *license plate recognition* rata-rata sebesar 12%, 10%, dan 8% untuk nilai *recall*, akurasi, dan F1 *score* pada citra hasil lokalisasi YOLOv4. Dengan citra uji sintetis menghasilkan rata-rata peningkatan sebesar 98%, 94%, dan 0,63 untuk nilai *recall*, akurasi, dan F1 *score*, dengan rata-rata nilai PSNR dan SSIM sebesar 18,58 dB dan 0,83.

Kata kunci: plat nomor, pengenalan karakter, *super-resolution, generative adversarial network.*

ABSTRACT

The ever-increasing growth of vehicles has had an impact on vehicles on the streets, which has also increased cases of violation of traffic rules. Until now, conventional detection and prosecution of violation cases are still not optimal. Therefore automatic detection and enforcement of violations are considered important, with the process of automatic license plate recognition (ALPR) or automatic vehicle number plate recognition can assist in the process of prosecution of violations. On the other hand, the ALPR process has drawbacks, namely the ability of Analog CCTV to capture images that are still of low quality due to a large amount of noise. Thus improving image quality to improve the accuracy of the character recognition process is considered important.

In this study, image quality improvement was carried out in the license plate recognition process for recognizing four-wheeled vehicle license plates using a super-resolution generative adversarial network (SRGAN). SRGAN is implemented after the license plate localization process, by producing a super-resolution image having 4x the number of pixels than the input image, which then super-resolution image is subjected to a character recognition process.

The test is carried out by measuring the results of precision, recall, and F1 scores as well as PSNR and SSIM values on the results of character recognition on four-wheeled vehicle license plates. Calculations were performed on the images before and after being subjected to SRGAN, resulting in an average increase in accuracy in the license plate recognition process of 12%, 10%, and 8% for recall, accuracy, and F1 scores in YOLOv4 localized images. With synthetic test images, it produces an average increase of 98%, 94%, and 0.63 for recall, accuracy, and F1 score, with an average PSNR and SSIM value of 18.58 dB and 0.83.

Keywords: license plate, character recognition, super-resolution, generative adversarial network.

PRAKATA

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa yang senantiasa mencurahkan rahmat-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Peningkatan Akurasi pada Pengenalan Plat Nomor Kendaraan Roda Empat dengan Menggunakan Super Resolution Generative Adversarial Network".

Tugas Akhir ini merupakan salah satu syarat untuk menyelesaikan studi pada Jurusan Informatika, Fakultas Teknik Industri, Universitas Pembangunan Nasional "Veteran" Yogyakarta. Dengan harapan juga Tugas Akhir ini agar dapat menjadi motivasi bagi mahasiswa informatika dalam menyelesaikan studi maupun sebagai bahan kajian dalam penelitian selanjutnya. Penulis menyadari bahwa dalam penulisan Tugas Akhir sehingga dapat diselesaikan dengan baik tidak terlepas oleh bimbingan dan arahan dari berbagai pihak. Banyak terimakasih penulis haturkan kepada:

1. Tuhan Yang Maha Esa yang senantiasa mencurahkan berkat dan pertolongannya.
2. Bapak Dr. Awang Hendrianto Pratomo, S.T.,M.T selaku Ketua Jurusan Teknik Informatika UPN "Veteran" Yogyakarta dan selaku Dosen Pembimbing Tugas Akhir.
3. Bapak Bambang Yuwono, S.T., M.T. selaku Dosen Pembimbing Tugas Akhir.
4. Bapak Sidiq Permana selaku mentor Capstone Project BANGKIT 2021
5. Bapak Dessyanto Boedi Prasetyo, S.T., M.T. dan Ibu Dr. Herlina Jayadianti, S.T.,M.T. selaku Dosen Penguji Tugas Akhir.
6. Ibu Vynska Amalia Permadi, S.Kom., M.Kom. selaku Dosen Pengampu Mata Kuliah *Machine Learning* atas ilmu, saran, arahan, dan dukungan kepada penulis.
7. Seluruh dosen, karyawan, dan staff prodi Informatika UPN "Veteran" Yogyakarta yang telah membantu penulis selama menempuh pendidikan.
8. Kedua orang tua dan keluarga besar yang selalu memberikan semangat, dukungan, dan doa.
9. Sahabat dan teman-teman penulis, Henry, Daniel, Anin, Gaudio, Filia, Levy, Syka, Rosa Ariq, Abyan yang telah mendukung dan memberikan bantuan selama pengerjaan Tugas Akhir.
10. Teman-teman informatika UPN "Veteran" Yogyakarta, terkhusus Daffa, Yesyua, Fahmi, Bela, Andra, Abel, Hilmy, Yones, Iskhak, Khalil, Naufal, mba Cici, dan mas Adib, mas Jundi yang telah mendukung dan memberikan bantuan selama pengerjaan Tugas Akhir.

Seluruh isi tugas akhir ini merupakan tanggung jawab penulis. Disadari juga Tugas Akhir ini masih sangat jauh dari kesempurnaan, oleh karena itu kritik serta saran yang membangun penulis harapkan.

Yogyakarta, 15 November 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR ALGORITMA

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pertumbuhan kendaraan bermotor setiap tahunnya terus meningkat. Menurut (Badan Pusat Statistik, 2022) pada tahun 2018 sampai dengan 2020, di Indonesia, jumlah mobil penumpang meningkat dari 14,8 juta menjadi 15,79 juta kendaraan dan untuk sepeda motor dari 106,65 juta menjadi 115 juta kendaraan. Pertumbuhan kendaraan tersebut berdampak terhadap padatnya kendaraan di jalanan, yang juga mengakibatkan bertambahnya kasus pelanggaran peraturan lalulintas (Bhat et al., 2021). Deteksi dan penindakan pelanggaran secara otomatis telah dilakukan menggunakan proses *license plate recoginiton* (LPR). Namun pada penerapan proses LPR tersebut masih memiliki beberapa kelemahan, yaitu tangkapan CCTV analog yang memiliki citra beresolusi rendah atau terdapat banyak noise dan rendahnya kontras cahaya yang mengakibatkan penurunan akurasi dalam pengenalan.

License Plate Recognition (LPR), yaitu proses pengenalan plat nomor kendaraan secara otomatis, merupakan salah satu penerapan dari *Computer Vision* (CV) (Bhat et al., 2021). Beberapa penelitian telah dilakukan untuk melakukan pengenalan plat nomor kendaraan secara otomatis, (Asif et al., 2019) menggunakan metode *illumination-invariant* untuk memperkirakan bagian yang memiliki *illumination* atau kecerahan cahaya sepasang yang menggambarkan lampu belakang kendaraan roda empat, dilakukan lokalisasi untuk memperkirakan titk tengah dari dua cahaya untuk diketahui keberadaan plat nomor. Pada (Zhang, Wang and Li, 2021) mengembangkan *deep-learning* framework untuk deteksi plat pada video yaitu *Video-based License plate detection and recognition* (V-LDPR). Pada (Lin and Li, 2019) menggunakan Mask R-CNN untuk mendeteksi plat nomor kendaraan dalam kondisi sudut yang ekstrem sampai dengan 70°. Dengan 0~60 derajat memiliki hasil mAP sampai dengan 91% dibandingkan dengan YOLOv2. Digunakan K-NN pada (Gunawan, Rohimah and Rahmat, 2019), memiliki akurasi deteksi sebesar 92.86%.

Namun pada beberapa penelitian tersebut masih memiliki beberapa hambatan dan kelemahan dalam proses lokalisasi maupun pengenalan karakter. Pada metode *illumination-invariant* (Asif et al., 2019) dalam suatu kodisi dengan posisi kendaraan yang sejajar bersebelahan dapat mengakibatkan kesalahan perkiraan prediksi plat nomor. Pada V-LDPR milik (Zhang, Wang and Li, 2021) memiliki akurasi yang tinggi (95,44), tetapi waktu deteksi per *frame* dalam video hanya memiliki 6-7*fps* dibandingkan dengan rata-rata deteksi YOLOv4 (Bochkovskiy, Wang and Liao, 2020) yaitu 65*fps*.

Dalam penerapannya proses LPR masih memiliki beberapa masalah eksternal. Beberapa CCTV tidak mampu menangkap citra dengan cukup jelas, mengakibatkan citra memiliki resolusi yang rendah dan memiliki banyak *noise* (Hamdi, Chan and Koo, 2021), ketika objek bergerak terlalu cepat, terjadinya hujan, pada (Gunawan, Rohimah and Rahmat, 2019) memiliki kelemahan saat mendeteksi plat nomor dengan kondisi karakter yang berhimpitan, banyaknya kemiripan karakter, dan juga terlalu banyak atau kurangnya pencahayaan yang ada, yang mengakibatkan penurunan akurasi pengenalan karakter.

Beberapa kelemahan citra tersebut telah dicoba diatasi dengan menerapkan beberapa *pre-processing* pada citra untuk meningkatkan tingkat akurasi pengenalan karakter (Gunawan et al., 2021; Sham et al., 2021). Namun pada kasus citra *low-resolution*, proses *pre-processing* tidak mampu membantu membangkitkan *feature* pada citra (Hamdi, Chan and Koo, 2021). Oleh karena itu pengolahan citra digital untuk meningkatkan kualitas citra akan diaplikasikan pada penelitan ini untuk permasalahan tersebut, agar proses pengenalan dapat dilakukan dengan lebih akurat.

Pengolahan citra digital untuk peningkatan kualitas citra dari citra *low-resolution* menjadi citra *high-resolution* atau proses *super-resolution* telah dilakukan oleh (Dong et al., 2014). Dengan menggunakan metode *super-resolution convolutional neural network* (SRCNN), memanfaatkan *non-linear mapping* pada proses *convolution* untuk memperoleh citra *super-resolution*, SRCNN memiliki struktur arsitektur yang sederhana dan telah memiliki hasil PSNR yang cukup baik (36.66dB *upscaling* 2x, 32.75dB *upscaling* 3x, dan 30.49dB *upscaling* 4x). Walaupun dengan hasil yang ada, SRCNN masih memerlukan *pre* maupun *post-processing* untuk memperoleh citra *super-resolution* yang baik. Pada (Ledig et al., 2016), menggunakan konsep *Generative Adversarial Network* (GAN) untuk menghasilkan *Super-Resolution Generative Adversarial Network* (SRGAN). Menggunakan SRResNet, arsitektur yang lebih dalam dari pada SRCNN, sehingga dapat menghasilkan citra *super-resolution* yang lebih tajam, dengan *upscaling* 4x memperoleh nilai PSNR 32.05dB. Pada (Hamdi, Chan and Koo, 2021) mencoba memperoses citra pada dua tahapan, *image enhancement* kemudian *super-resolution* yang mana kedua proses tersebut menggunakan bantuan GAN yaitu *Double Generative Adversarial Networks for Image Enhancement and Super Resolution* (D_GAN_ESR). Walaupun dapat menghasilkan tingkat PSNR yang baik yaitu 31.28dB, arsitektur ini memiliki tingkat komputasi yang sangat berat, dikarenakan melalui dua tahapan GAN yang berbeda.

Berdasarkan masalah yang terdapat pada proses LPR, yaitu kemampuan CCTV Analog dalam menangkap citra masih memiliki kualitas citra yang rendah dikarenakan banyaknya noise sehingga mengakibatkan penurunan akurasi dalam pengenalan karakter. Pada penelitian ini akan digunakan SRGAN (Ledig et al., 2016) untuk meningkatkan kualitas citra sebelum dilakukan proses pengenalan karakter, SRGAN digunakan dikarenakan tingkat PSNR yang paling unggul daripada arsitektur lainya dan juga tingkat komputasi yang lebih rendah daripada D_GAN_ESR milik (Hamdi, Chan and Koo, 2021). Untuk proses lokalisasi plat nomor akan digunakan arsitektur YOLOv4 (Bochkovskiy, Wang and Liao, 2020), YOLOv4 digunakan dikarenakan memiliki tingkat rata-rata deteksi yang baik.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang ada, proses *license plate recognition* (LPR) masih memiliki masalah eksternal pada kemampuan CCTV Analog dalam menangkap citra, yaitu resolusi citra yang rendah dan banyaknya *noise* pada citra tangkapan CCTV Analog, yang kemudian mengakibatkan penurunan akurasi pada proses pengenalan karakter.

1.3 Batasan Masalah

Adapun batasan masalah dalam penelitian ini agar lebih mudah dipahami dan tidak terlalu luas dalam penulisannya, sebagai berikut:

a. Proses *super-resolution* akan menghasilkan citra dengan ukuran empat kali lebih besar dari input *low-resolution*.
b. Data citra plat nomor adalah data sekunder berasal dari sumber data terbuka kaggle dan Google Open Images Dataset.
c. Data citra latih *super-resolution generative adversarial network* merupakan data sintetis, diperoleh dari *random cropping* pasangan citra *low-resolution* dan *high-resolution* dari DIV2K dataset 4x *scale*.
d. Data latih *super-resolution* yang digunakan berukuran *high-resolution* 96 x 96 *pixel* dan *low-resolution* 24 x 24 *pixel*.
e. Input berupa gambar jpg atau jpeg.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk meningkatkan akurasi pada proses *license plate recognition* (LPR) dengan dilakukan peningkatan kualitas citra sebelumnya.

1.5 Manfaat Penelitian

a. Mengetahui akurasi proses LPR setelah dilakukan peningkatan citra sebelumnya.
b. Dapat menjadi referensi penelitan peningkatan kualitas citra selanjutnya.
c. Dapat diterapkan pada sistem pengenalan plat nomor kendaraan secara otomatis.

1.6 Metodologi Penelitian dan Pengembangan Sistem

Tahapan metode penelitian yang dilakukan adalah sebagai berikut:

1. Pengumpulan Citra.
   Citra dikumpulkan dari *Open Image Dataset* V6 dan DIV2K.
2. Generasi citra sintetis.
   Citra sintetis dibuat dengan melakukan operasi *resize* pada citra asli.
3. Implementasi arsitektur deteksi lokalisasi plat nomor kendaraan.
   Implementasi pembuatan model deteksi menggunakan arsitektur YOLOv4 dilakukan pada web IDE Python Google Colab.
4. Implementasi arsitektur *super-resolution generative adversarial network*.
   Implementasi pembuatan model *super-resolution* dengan arsitektur SRGAN dilakukan pada web IDE Python Google Colab.
5. Pengujian
   Pengujian dilakukan terhadap kualitas citra *super-resolution* dan hasil deteksi karakter pada citra *super-resolution*.

Tahapan pengembangan sistem yang dilakukan adalah sebagai berikut:

1. Analisis kebutuhan sistem
2. Desain sistem
3. Implementasi
4. Pengujian sistem

1.7 Sistematika Penulisan

Proposal tugas akhir ini dibagi dalam beberapa topik pembahasan yang disusun secara sistematis sebagai berikut:

BAB I PENDAHULUAN

Bab ini membahas mengenai latar belakang masalah penelitian, perumusan masalah, batasan masalah penelitian, tujuan penelitian, manfaat penelitian, serta sistematika penulisan.

BAB II TINJAUAN LITERATUR

Pada bab ini membahas tentang dasar teori dan penelitian serupa yang pernah dilakukan sebelumnya pada *license plate recognition*, deteksi plat nomor, generasi citra *super resolution*, dan konsep dasar pendukung dalam pembangunan sistem lainnya.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan mengenai alur metode penelitian yang akan dilaksanakan, pengumpulan data untuk data latih YOLOv4 dan SRGAN, metode *pre-processing*, tahapan perancangan dan pembangunan model, dan rancangan pengujian model maupun sistem.

BAB IV IMPLEMENTASI DAN HASIL

Pada bab ini membahas hasil implementasi berdasarkan rancangan yang telah dibuat dan juga melakukan analisis hasil yang dilakukan pada parameter input dan hasil proses berdasarkan rancangan pengujian model maupun sistem yang telah dibuat.

BAB V KESIMPULAN DAN SARAN

Bab ini merupakan bagian terakhir dari penelitian yang dilakukan, dimana di dalamnya terdapat kesimpulan untuk menjawab rumusan masalah yang ada, serta saran pengembangan dari penelitian yang telah dilakukan untuk penelitian kedepannya.

BAB II
TINJAUAN LITERATUR

2.1. *Computer Vision*

Computer Vision (CV) merupakan sebuah algoritma atau usaha sebuah sistem komputer untuk dapat mengidentifikasi secara otomatis dan juga memahami dan menyimulasikan penglihatan manusia secara digital (Feng et al., 2019). Algoritma *Computer Vision* termasuk pada (i) *Object recognition*, yaitu untuk menentukan apakah citra memiliki objek tertentu, pada (Kangune, Kulkarni and Kosamkar, 2019) menggunakan CNN dan SVM, mencoba mengenali tingkat kematangan buah dengan satu input citra buah yang kemudian mengeluarkan dua kondisi, yaitu matang atau belum matang. (ii) *Object detection*, yaitu kemampuan lokalisasi objek, pada (Sham et al., 2021) menggunakan YOLO untuk mendeteksi dan mendapatkan koordinat *bounding-box* plat nomor kendaraan pada suatu citra. (iii) *Image segmentation*, yaitu kemampuan lokalisasi area objek pada akurasi area *pixel* yang membedakan segmen citra pada klasifikasi objek tertentu, setiap karakter yang terdeteksi pada hasil sub-citra *bounding-box* (Agarwal et al., 2018; Sham et al., 2021) ter-segmentasi antara latar belakang dan kontur *pixel* dari setiap karakter.

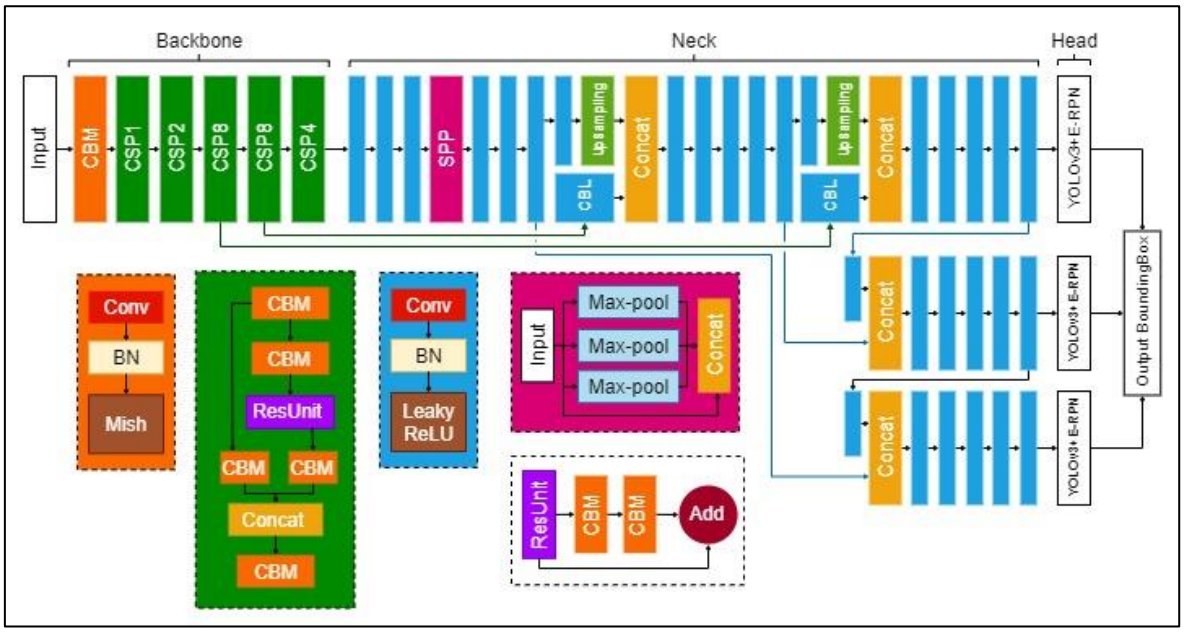
2.2. *License Plate Recognition*

License *Plate Recognition* adalah bagaimana algoritma *Computer Vision* dapat melakukan *Object Detection and Recognition* atau untuk mendeteksi dan mengenali plat nomor dari suatu kendaraan yang tertangkap melintasi area observasi kamera CCTV maupun kamera digital (Imaduddin et al., 2019). Pada umumnya dilakukan dalam tiga tahapan utama (Lin and Li, 2019), deteksi kendaraan, lokalisasi plat nomor, dan deteksi *bounding-box*. Dimana hasil *bounding-box* digunakan sebagai batasan pemotongan citra pada *Optical Character Recognition* (OCR).

Seiring dengan pertumbuhan kendaraan, proses *license plate recognition* (LPR) semakin banyak dimanfaatkan berbagai pihak dalam berbagai jenis kebutuhan. Seperti *smart parking management system* (Venkata Sudhakar et al., 2021; Gunawan et al., 2021) untuk kebutuhan manajemen tempat parkir otomatis, sampai dengan deteksi pelanggaran peraturan lalulintas (Agarwal et al., 2018) dengan menggunakan metode *Automatic License plate recognition* (ALPR). Beberapa algoritma yang digunakan untuk melakukan *Object Detection and Recognition* adalah, YOLO (Gunawan et al., 2021; Sham et al., 2021) memiliki kelebihan pada tingginya tingkat *frame rate* per detiknya dalam mendeteksi objek pada video. Namun membutuhkan dataset berjumlah besar untuk menghasilkan model yang akurat (Redmon et al., 2016). Pada (Imaduddin et al., 2019), CNN digunakan sebagai algoritma untuk lokalisasi plat nomor kendaraan, beberapa citra gagal dideteksi dikarenakan tekstur warna dari plat nomor serupa dengan kendaraan.

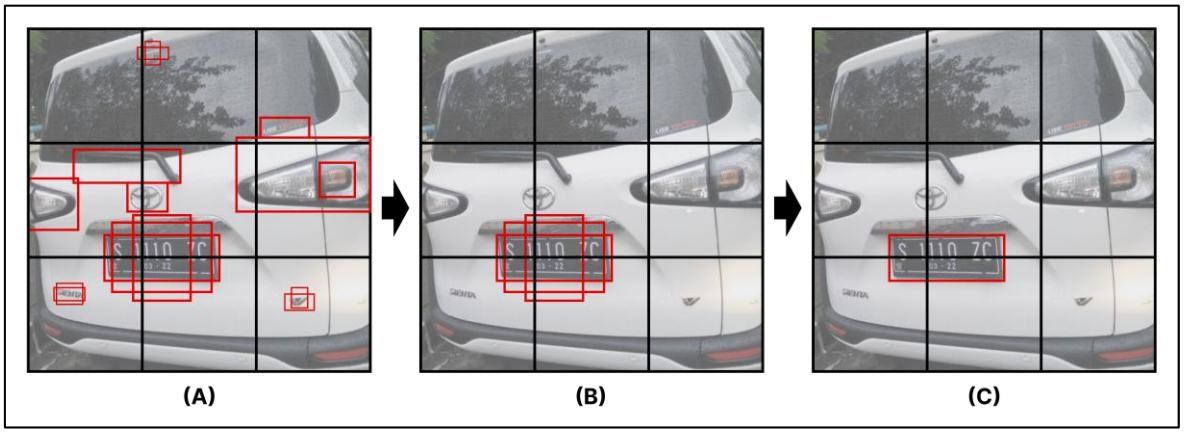
Hingga kini *License Plate Recognition* masih memiliki beberapa masalah dalam prosesnya, yaitu pada kemampuan CCTV dalam menangkap citra (Hamdi, Chan and Koo, 2021). Beberapa CCTV tidak mampu menangkap citra dengan cukup jelas ketika objek bergerak terlalu cepat, terjadinya hujan, maupun kemampuan CCTV dalam menangkap citra masih memiliki resolusi yang rendah. Dalam beberapa kasus, banyaknya noise dan juga distorsi warna juga menjadi tantangan pada proses ALPR.

Pada penelitian ini akan digunakan YOLOv4 (Bochkovskiy, Wang and Liao, 2020) dengan arsitektur sebagaimana Gambar 2.1. Dengan tiga struktur utama yaitu *backbone*, *neck*, dan *head*. Dimana pada *backbone* menggunakan CSPDarknet53 (Wang et al., 2020a) untuk ektraksi fitur pada citra, kemudian proses lokalisasi objek dilakukan pada *neck* dengan *path-aggregation network* (PAN), *spatial pyramid pooling layer* (SPP), dan *Spatial Attention Module* (SAM), dan terakhir pada *head* dilakukan penelusuran untuk setiap resolusi spasial sehingga didapati setiap kemungkinan *bounding-box* pada citra.



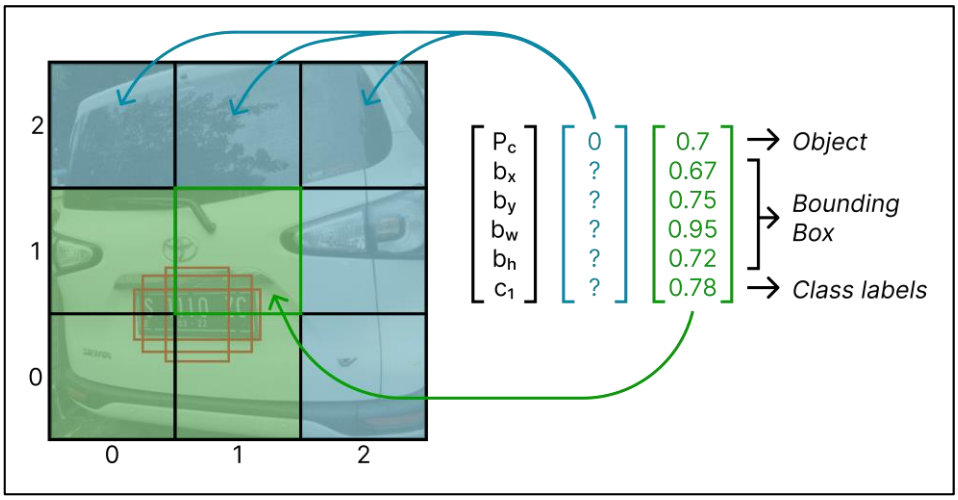
Gambar 2.1. Arsitektur YOLOv4
Sumber (Zakria et al., 2022)

Proses deteksi YOLO adalah dengan membagi citra sebagai SxS area bagian, sebagai ilustrasi pada Gambar 2.2 (C) menggunakan 3x3 *grid*, dengan setiap area bagian akan memiliki hasil prediksi objek dalam satu *boundary-box*, untuk memprediksi kemungkinan objek pada citra, terlihat pada Gambar 2.2 (A). Penggunaan *intersection over union* atau IoU untuk menyeleksi lebih lanjut nilai *bounding-box* apakah memiliki kelas objek yang diinginkan, sehingga mendapatkan hasil sebagaimana Gambar 2.2 (B).



Gambar 2.2. Ilustrasi proses deteksi YOLOv4

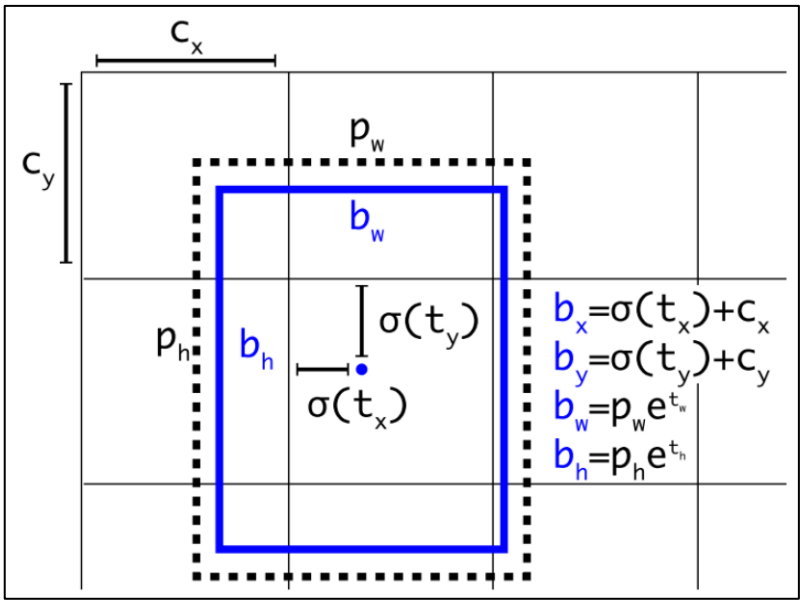
Untuk setiap *boundary-box* pada Gambar 2.2 (A) akan memiliki nilai keluaran sebagaimana Gambar 2.2, dengan P_c sebagai *objectness score*. Pada b_x dan b_y merupakan koordinat relatif dari *bounding-box*, b_w dan b_h merupakan nilai lebar dan tinggi relatif *bounding-box* terhadap lebar dan tinggi asli citra, dan dengan C_n adalah *class confidence score* untuk setiap kategori kelas yang ada, atau sebagai contoh apakah termasuk sebagai mobil, motor, ataupun manusia.



Gambar 2.3. Ilustrasi nilai keluaran setiap *boundary-box*

Hasil seleksi IoU pada citra dipersempit kembali menggunakan *non-max suppression* (Zheng et al., 2020) sehingga mendapatkat *bounding-box* dengan nilai *confidence* terbaik, sebagaimana Gambar 2.2 (C). Penggunaan *non-max suppression* (Zheng et al., 2020) berfungsi untuk menyeleksi nilai *bounding-box* yang memiliki nilai *objectness score* atau nilai probabilitas ada tidaknya objek pada suatu *boundary-box*, kurang dari atau sama dengan 0,6. Terlihat pada Gambar 2.2 dengan mengeliminasi *boundary-box* yang memiliki nilai *objectness score* kurang dari atau sama dengan 0,6 maka area *boundary-box* akan dianggap memiliki nilai *objectness score* 0 dan tereliminasi, sehingga tersisa *boundary-box* yang memiliki nilai *objecness score* lebih dari 0,6 dan memiliki *bounding-box* dengan nilai *class confidence score* tertinggi sebagai hasil akhir Gambar 2.2 (C).

Pada YOLOv4 prediksi koordinat *bounding-box* menggunakan regresi, dimana setiap sel area memiliki prediksi berdasarkan *offset* dengan *anchor boxes*, pada Gambar 2.4 dengan $\boldsymbol{C_x}$ dan $\boldsymbol{C_y}$ adalah nilai *offset* area bagian relatif dengan pojok kiri atas dari *grid*. $\boldsymbol{P_w}$ dan $\boldsymbol{P_h}$ sebagai nilai tinggi dan lebar dari *anchor*. Nilai $\boldsymbol{b_x}$ dan $\boldsymbol{b_y}$ merupakan koordinat titik tengah dari *bounding-box* yang terprediksi relatif dengan nilai *offset* dari *anchor*. $\boldsymbol{b_w}$ dan $\boldsymbol{b_h}$ merupakan lebar dan tinggi *bounding-box* terprediksi relatif dengan nilai *offset* dari *anchor*. $\boldsymbol{\sigma}(.)$ merupakan fungsi sigmoid, dan pada $\boldsymbol{t_x}$, $\boldsymbol{t_y}$, $\boldsymbol{t_w}$, dan $\boldsymbol{t_y}$ merupakan hasil prediksi dari *network* yang kemudian digunakan untuk penyesuaian akhir pada deteksi untuk *frame* tersebut.



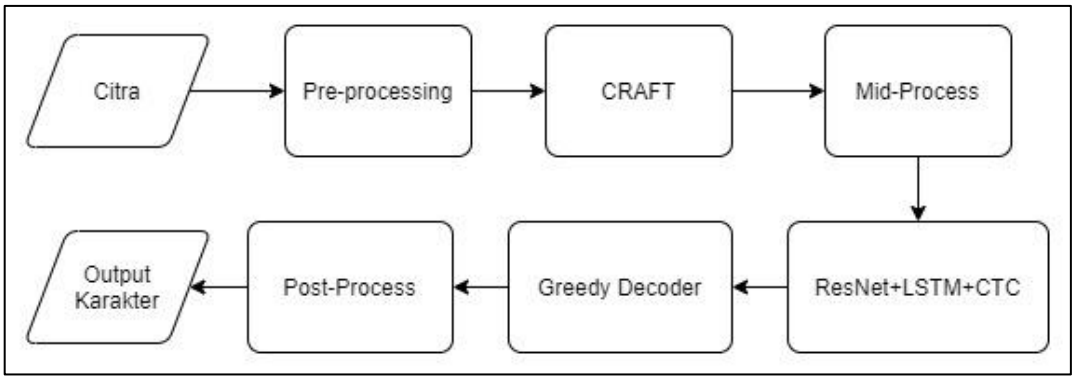
Gambar 2.4. Ilustrasi *anchor offset bounding-box* YOLOv4
Sumber (Redmon and Farhadi, 2018)

Hasil dari deteksi plat nomor menggunakan YOLOv4 adalah berupa koordinat *bounding-box* dan juga besar dari sebuah *bounding-box* yang telah terprediksi. Hasil keluaran tersebut bukan hasil mutlak, melainkan relatif dengan *offset anchor* yang kemudian harus dihitung kembali untuk mencari nilai mutlak.

2.3. *Scene Text Recognition*

Scene Text Recognition atau pengenalan teks pada sebuah citra atau dengan istilah lain *Optical Character Recognition* (OCR), telah digunakan pada berbagai aspek kehidupan, salah satunya diterapkan untuk kebutuhan *License Plate Recognition* atau pengenalan plat nomor kendaraan. Pada (Hamdi, Chan and Koo, 2021; Sham et al., 2021) digunakan *Open Source* OCR *engine*, yaitu, *Tesseract* (Smith and Podobny, 2021) dan easyOCR (JaidedAI, 2021; Shi, Bai and Yao, 2017; Baek et al., 2019) untuk pengenalan karakter pada plat nomor. MobileNetV2 digunakan oleh (Gunawan et al., 2021) sebagai algoritma pengenalan karakter, dengan penerapan proses segmentasi pada setiap karakter terlebih dahulu. Pada penelitian ini akan digunakan easyOCR dikarenakan pada algoritma Tesseract dan MobileNetV2 dibutukan banyak *pre-processing* untuk bisa mengeluarkan hasil pengenalan karakter yang optimal dari pada easyOCR yang membutuhkan lebih sedikit *pre-processing.*

Alur algoritma dari EasyOCR sebagaimana pada Gambar 2.5, menggunakan algoritma *Character-Region Awareness For Text detection* (CRAFT) milik (Baek et al., 2019) untuk melakukan lokalisasi karakter maupun kalimat pada citra. Pengenalan karakter menggunakan algoritma CRNN, dengan ekstraksi fitur menggunakan ResNet dan VGG, dengan *Long Short-Term Memory* (LSTM) milik (Hochreiter and Schmidhuber, 1997) sebagai *sequence labeling* dan *Connectionist Temporal Classification* (CTC) milik (Adams and Saaty, 2006) untuk decoding.



Gambar 2.5. Alur algoritma EasyOCR
Sumber (JaidedAI, 2021)

Algoritma *optical character recognition* memiliki kelemahan pada pengenalan citra jika terdapat banyaknya *noise* maupun kurangnya kontras antara karakter dengan latar, yang akan mengakibatkan menurunnya akurasi pengenalan karakter (Dong et al., 2015). Beberapa kelemahan OCR telah dicoba diatasi pada (Smith and Podobny, 2021) dengan diterapkan *pre-processing* agar dapat terdeteksi pada algoritma OCR. G*rayscaling* yaitu mengubah citra RGB 24-bit menjadi citra monokromatik 8-bit untuk mengurangi rentang warna yang ada. Pengurangan *noise* menggunakan *gaussian smoothing*. *Otsu's binarization and thresholding* digunakan untuk mengubah nilai *pixel* citra menjadi 0 atau 1, dan *segmentation* untuk membedakan setiap *pixel* karakter dengan latar belakang.

Penerapan *pre-processing* tersebut telah dapat meminimalisir kegagalan dalam pengenalan setiap karakternya. Namun pada kasus citra *low-resolution*, proses *pre-processing* tidak mampu membantu membangkitkan *feature* pada citra, dikarenakan hilangnya detail citra dalam skala *pixel* yang terlalu kecil (Hamdi, Chan and Koo, 2021).

2.4. Citra *Super Resolution*

Super-resolution (SR) adalah suatu proses dalam melakukan restorasi citra kembali pada keadaan *high-resoution* (HR) berdasarka satu atau lebih referensi citra *low-resolution* (LR) (Bashir et al., 2021). Dibedakan menjadi dua kategori, (i) *single image super-resolution* (SISR), dimana satu pasang citra LR-HR digunakan untuk proses *super-resolution*. (ii) *Multi-image super-resolution* (MISR), dengan beberapa pasang referensi citra LR-HR untuk melakukan restorasi. Dimana proses SISR lebih banyak digunakan dikarenakan alasan efisiensi pemrosesan dan sumberdaya komputasi (Yang et al., 2019).

Beberapa algoritma telah digunakan untuk melakukan metode *single image super-resolution*. Pada (Dong et al., 2014) digunakan *deep convolutional neural network* (CNN) atau disebut sebagai SRCNN, yang juga merupakan dasar pengembangan dari beberapa algoritma lain. (Shi et al., 2016) menggunakan *efficient sub-pixel convolutional neural network* (ESPCN), menghasilkan efisiensi komputasi dan hasil *Peak Signal-to-Noise Ratio* (PSNR) sedikit lebih baik (23.72db) dibandingan SRCNN (Dong et al., 2014) (23.67db). (Ledig et al., 2016) Menggunakan *Generative Adversarial Network* (GAN) pada proses *super-resolution* (SRGAN). Dengan hasil SRGAN memiliki nilai PSNR lebih baik (29.40) dibandingkan SRCNN, ESPCN, dengan juga SRGAN memiliki nilai MOS paling tinggi dibandingkan dengan algoritma *super-resolution* lainnya.

2.5. *Generative Adversarial Network*

Generative Adversarial Network (GAN) (Goodfellow et al., 2014) adalah sebuah *framework* atau algoritma yang didasarkan pada teori *minimax two-player game* dari dua *adversarial network*, yaitu, *generator* dan *discriminator*. Dimana alur dari proses GAN dapat dilihat pada Gambar 2.6, dengan tujuan dari GAN adalah untuk memaksimalkan *output* sebuah *generator* atau $G(z;\theta^{(G)})$ agar memiliki hasil yang dapat menipu *discriminator* untuk menganggap hasil serupa dengan data asli. Dengan *vector noise z* sebagai input pada *generator* dengan hasil *output* data $x = G(z;\theta^{(G)})$ atau $p_{model}(x)$ sebagai input pada *discriminator*. Data *output* dari *generator* dan juga data asli $p_{data}(x)$ digunakan menjadi data input pada *discriminator*. Input tersebut digunakan sebagai data latih pada *discriminator* untuk dapat meminimalisir lolosnya data buatan dari model *generator* $p_{model}(x)$ (Salimans et al., 2016).

Hasil dari *discriminator* digunakkan untuk membantu mengevaluasi input *vector noise* pada *generator* untuk dapat menghasilkan *output* yang dapat membohongi *discriminator* agar terklasifikasikan sebagai data asli atau dapat dimengerti sebagai data yang sangat mirip dengan distibusi data asli, *output* evaluasi tersebut juga digunakan untuk melatih *discriminator* dalam membedakan data asli atau hasil buatan (Goodfellow et al., 2014). Berdasarkan (Salimans et al., 2016; Goodfellow et al., 2014) didapati alur dari proses GAN pad Gambar 2.6. Dideskripsikan pula algoritma pada Algoritma 1 dari GAN menurut (Goodfellow et al., 2014).

Algoritma 1 : ***Minibatch stochastic gradient descent training*** **GAN**

The number of steps to apply to the discriminator, k, is a hyperparameter. We used k = 1, the least expensive option, in our experiments.

Begin for n trainning itteration **do**

  Begin for k steps **do**

    Sample minibatch of m noise samples $\{\boldsymbol{z^{(1)}},\dots,\boldsymbol{z^{m}}\}$ from noise prior $\boldsymbol{p_{model}(x)}$.

    Sample minibatch of m examples $\{\boldsymbol{x^{(1)}},\dots,\boldsymbol{x^{m}}\}$ from data generating distribution $\boldsymbol{p_{data}(x)}$.

    Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D(x^i)+\log\left(1-D\left(G(z^{(i)})\right)\right)\right].$$

  End for

  Sample minibatch of m noise samples $\{\boldsymbol{z^{(1)}},\dots,\boldsymbol{z^{m}}\}$ from noise prior $\boldsymbol{p_{model}(x)}$.

  Update the generator by descending its stochastic gradient:

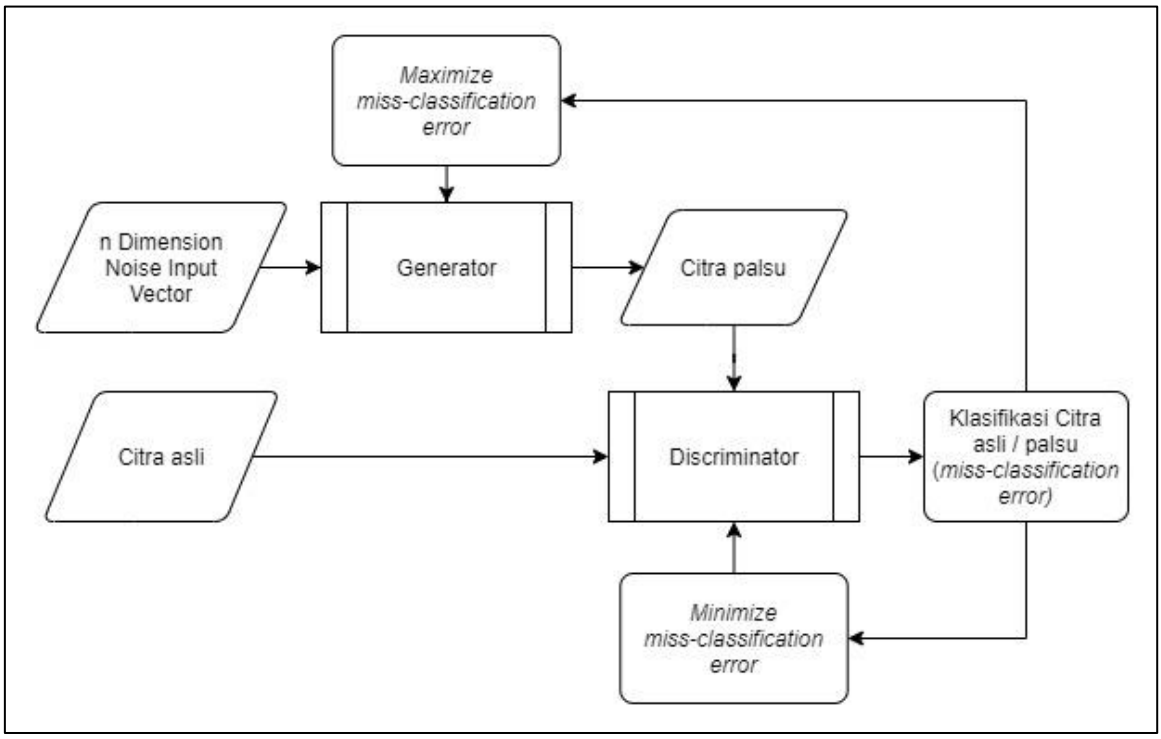
$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G(z^{(i)})\right)\right).$$

End for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

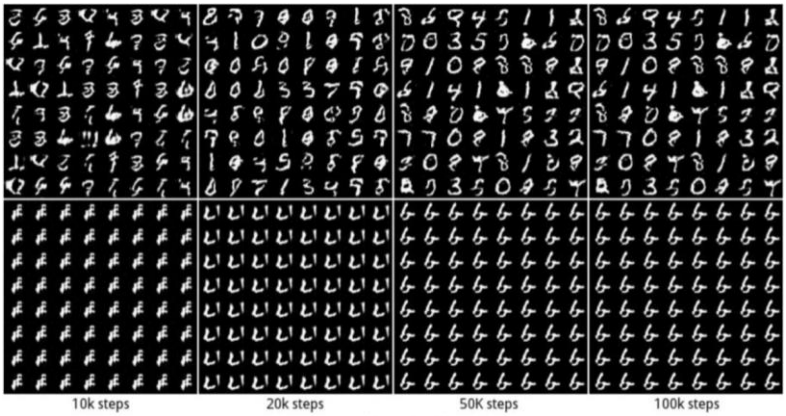
Beberapa penelitian tentang GAN yaitu, *Image-to-image Translation* menggunakan CycleGAN (Jay et al., 2017), GAN-*based Video Generation* menggunakan TecoGAN (Chu et al., 2020), dan generasi citra *Super Resolution* (SR) (Ledig et al., 2016; Wang et al., 2020b; Hamdi, Chan and Koo, 2021; Lin and Li, 2019) dengan target *output generator* pada metode SR adalah sebuah citra beresolusi tinggi atau citra *Super Resolution* (SR.

Sayangnya pada proses *training* GAN masih terdapat masalah (Salimans et al., 2016), adalah ketika GAN sulit untuk mencari *Nash equilibrium* dari *two-playe non-cooperative game*, atau dimana *Generator* akan tetap menggunakan kombinasi *vector noise* yang sama, dikarenakan hasil pertimbangan informasi yang didapat setelah evaluasi tidak memberikan perubahan keputusan pada *Generator*.



Gambar 2.6 Alur proses GAN
Sumber (Goodfellow et al., 2014)

Pada istilah lain *Discriminator* tidak mengirimkan hasil evaluasi yang signifikan dikarenakan *Generator* menghasilkan citra yang dianggap *Discriminator* telah serupa dengan distribusi data asli, mengakibatkan pula *Generator* menghasilkan data tanpa peningkatan *feature* atau data serupa berulang kali, ilustrasi dapat dilihihat pada Gambar 2.7, yang sering disebut sebagai keadaan *mode collapse*.

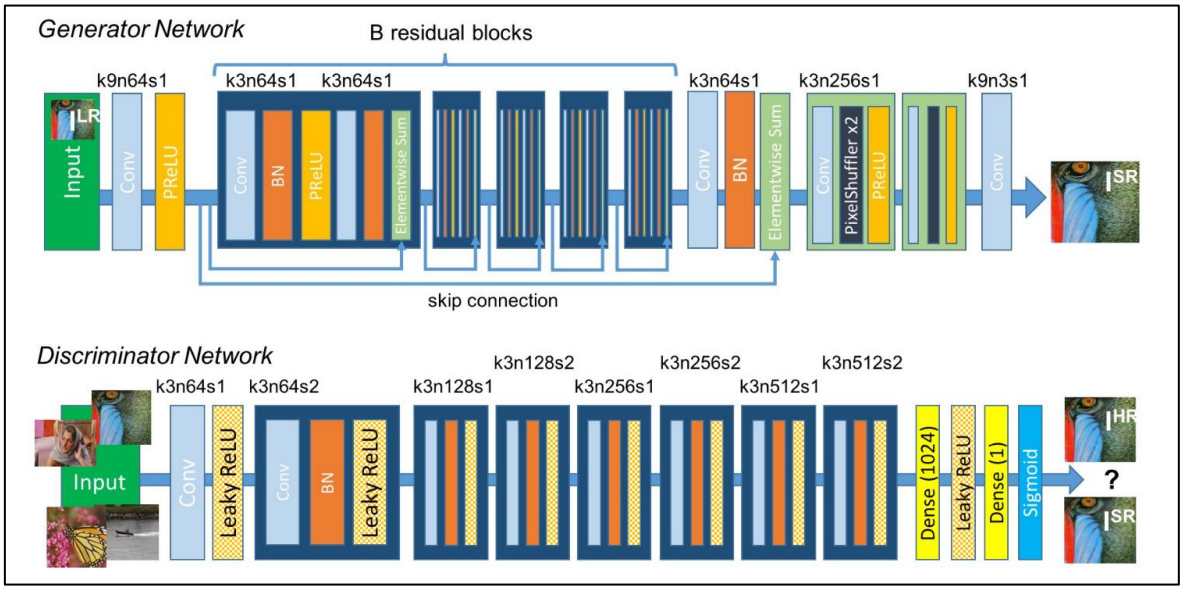


Gambar 2.7 Ilustrasi *generator* dalam mode *collapse*
Sumber (Sanjeevi, 2019)

2.6. *Super-Resolution Generative Adversarial Network*

Super-Resolution Generative Adversarial Network (SRGAN) (Ledig et al., 2016) menggunakan pasangan input *low-resolution* (LR) dan *high-resolution* (HR) untuk melakukan generasi citra *super-resolution* (SR). Dengan melakukan perubahan input pada *generator*, sesuai dengan alur proses GAN Gambar 2.6, dari *vector noise* menjadi citra *low-resolution* (LR). Menghasilkan citra empat kali lebih besar daripada input citra *low-resolution*. Penggunaan *perceptual loss* sebagai bahan evaluasi hasil *generator* untuk dapat memaksimalkan hasil generasi semirip mungkin dengan citra HR. *Perceptual loss* adalah hasil akumulasi dari dua *loss function*, yaitu, VGG19 *content loss* dan *adversarial loss*.

Generator pada Struktur SRGAN (Ledig et al., 2016) dapat dilihat pada Gambar 2.8, menggunakan arsitektur SRResNet. Dengan input citra *low-resolution*, masuk kedalam layer *convolution* yang pertama, dengan kernel 9x9 dan 64 *feature map*, *convolution* layer tersebut diikuti oleh fungsi aktivasi *parametric* ReLU milik (He et al., 2015). Layer berikutnya adalah *feed-forward fully convolutional* SRResNet, memanfaatkan tumpukan *residual block* berjumlah 16 buah berisikan layer *convolution* dengan kernel 3x3 dan 64 *feature map*, diikuti dengan *batch normalization* menurut (Ioffe and Szegedy, 2015), dan aktivasi *parametric* ReLU sesuai pada (He et al., 2015), diikuti dengan layer *convolution* dan *batch normalization* terakhir, layer terakhir pada blok adalah metode *elementwise sum*. Metode *elementwise sum* menggunakan hasil output dari struktur *feed-forward* dan output dari *skip connection* untuk menghasilkan output akhir yang akan dibawa pada *residual block* selanjutnya. Dua blok terakhir adalah layer yang berguna untuk meningkatkan resolusi citra empat kali atau dua kali dua proses *upscale*. Dengan *pixel shuffler* berisikan dua layer *sub-pixel convolution*.



Gambar 2.8. Arsitektur *Generator* dan *Discriminator* dari SRGAN
Sumber (Ledig et al., 2016)

Pada dasarnya algoritma GAN adalah bagaimana *discriminator* berusaha membedakan citra asli dengan citra buatan, sedangkan *generator* mencoba menghasilkan citra *super-resolution* ,yaitu citra yang semirip mungkin dengan citra *high-resolution*, sehingga dapat

lolos dari deteksi *discriminator*. Arsitektur *discriminator* pada SRGAN (Ledig et al., 2016) dapat dilihat pada Gambar 2.8, dengan layer *convolutional* dan *Leaky* ReLU (Xu et al., 2015) *alpha* pada 0,2. Struktur blok layer *convolutional* terdiri dari layer *convolutional* dengan 64x64 *feature* meningkat dua kali lipat setiap dua blok sampai dengan 512x512 *feature*. Dengan layer terakhir berupa *dense layer* diikuti dengan aktivasi *sigmoid* yang digunakan untuk klasifikasi citra asli dengan citra buatan atau citra *super-resolution*. Digunakan *binary crossentropy* sebagai *loss function* dan adam sebagai *optimizer* untuk klasifikasi pada *discriminator*. Algoritma *training* dari SRGAN dapat dilihat pada Algoritma 2. Pada `GANModel` digunakan dua loss function yaitu *binary crossentropy* dan *mean squared error* (MSE) dengan adam sebagai *optimizer*.

Algoritma 2 : Algoritma *training super-resolution* GAN

```
INIT GANModel, lrImages, hrImages
SET steps TO 200000

BEGIN FOR steps training itteration DO
   PASS lrImages INTO GANModel
      SAVE OUTPUT INTO perceptualLoss, discriminatorLoss
   COMPUTE MEAN OF perceptualLoss
   COMPUTE MEAN OF discriminatorLoss
   FOR EVERY 1000 steps DO
      SAVE model INTO checkpoint
      COMPUTE psnrValue USING validationDataset
END FOR
```

Ukuruan *batch* sama dengan satu digunakan sesuai pada (Ledig et al., 2016) dan (Goodfellow et al., 2014). Dengan `generatorModel` dan `discriminatorModel` sesuai dengan struktur arsitektur pada Gambar 2.8, dan dengan `GANModel` dideskribsikan sesuai pada Algoritma 3.

Algoritma 3 : Algoritma model *super-resolution* GAN

```
INIT generatorModel, discriminatorModel, vggLR, vggHR, lrImages, hrImages
INPUT lrImages, hrImages
OUTPUT perceptualLoss, discriminatorLoss

FUNCTION GANModel BEGIN
   PASS lrImages INTO generatorModel
      SAVE OUTPUT INTO srImages, genLoss
   PASS hrImages AND srImages INTO discriminatorModel
      SAVE OUTPUT INTO hrLoss, srLoss
   COMPUTE MSE OF vggHR AND vggLR
      SAVE OUTPUT INTO contentLoss
   COMPUTE contentLoss + 0.001 * genLoss
      SAVE OUTPUT INTO perceptualLossLoss
   COMPUTE hrLoss + srLoss
      SAVE OUTPUT INTO discriminatorLoss

   UPDATE generatorModel gradients USING perceptualLoss
   UPDATE discriminatorModel gradients USING discriminatorLoss
END
```

Ukuruan *batch* sama dengan satu digunakan sesuai pada (Ledig et al., 2016) dan (Goodfellow et al., 2014). Dengan `generatorModel` dan `discriminatorModel` sesuai dengan struktur arsitektur pada Gambar 2.8, dan dengan **`GANModel`** dideskribsikan sesuai pada Algoritma 3. Dengan perhitungan *perceptual loss* selanjutnya didapatkan menggunakan Persamaan 2.3 dengan content loss pada Persamaan 2.4 dan adversarial *loss* atau *generator loss* berdasarkan Persamaan 2.5. Ekstraksi fitur pada citra *high-resolution* dan *super-resolution* untuk mengetahui *content loss* digunakan layer VGG19 sebagai mana Gambar 2.10 (E), dimana digunakan 20 layer pertama, yaitu sampai dengan blok *convolusi* ke 5 layer ke 4, atau sebelum layer *maxpool* terakhir.

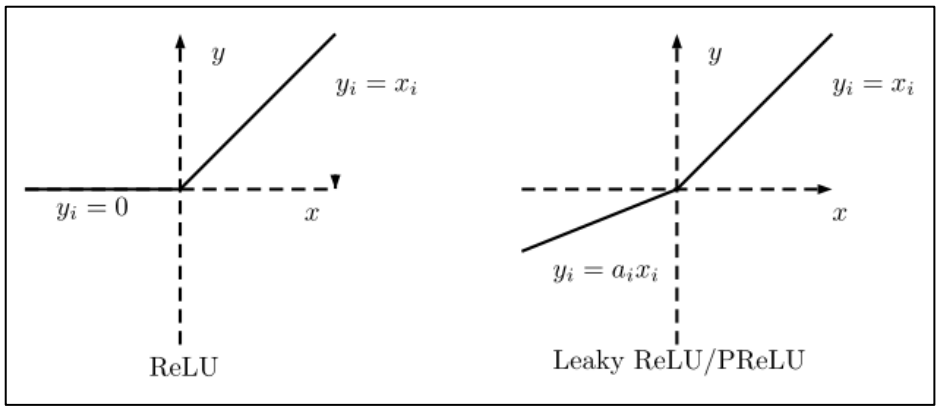
2.6.1. Fungsi Aktivasi

Pada arsitektur GAN digunakan dua jenis layer aktivasi, yaitu *leaky rectified linear units* (Leaky ReLU) dan *parametric rectified linear units* (PReLU) sesuai pada (Xu et al., 2015). Kedua aktivasi terebut memiliki dasar persamaan seperti *rectified linear units* (ReLU), dapat dilihat pada Persamaan 2.1 sebagai ReLU dan Persamaan 2.2 sebagai Leaky ReLU dan juga PReLU.

$$y_i \begin{cases} x_i & if\ x_i \geq 0 \\ 0 & if\ x_i < 0 \end{cases} \quad \text{.....(2.1)}$$

$$y_i \begin{cases} x_i & if\ x_i \geq 0 \\ \frac{x_i}{a_i} & if\ x_i < 0 \end{cases} \quad \text{.....(2.2)}$$

Dapat dilihat pada Gambar 2.9, fungsi aktivasi PReLU digunakan untuk mengubah nilai pixel dibawah nol menjadi y_i dengan nilai a atau *alpha* didapatkan secara berkala dari *training* melalui *back propagation*.



Gambar 2.9. Fungsi aktivasi
Sumber (Xu et al., 2015)

Pada dasarnya PReLU akan menjadi ReLU jika nilai a_i yang didapat pada proses training memiliki nilai sama dengan 0 dan akan menjadi Leaky ReLU jika nilai a_i yang didapat bernilai desimal kurang dari 1.

2.6.2. *Perceptual loss function*

Perceptual loss atau l^{SR} adalah hasil akumulasi dari dua *loss function*. Dengan $l^{SR}_{VGG/i.j}$ adalah VGG19 *content loss* dan $10^{-3}l^{SR}_{Gen}$ sebagai *adversarial loss*. *Perceptual loss* akan digunakan untuk meningkatkan dan mengevaluasi kinerja dari *generator*. Ditunjukan dengan persamaan menurut (Ledig et al., 2016) sebagaimana Persamaan 2.3,

$$l^{SR} = l^{SR}_{VGG/i.j} + 10^{-3}l^{SR}_{Gen} \quad (2.3)$$

Menggunakan VGG *content loss* berdasarkan *pre-trained* arsitektur VGG19 milik (Simonyan and Zisserman, 2015) sampai dengan *maxpool* layer sebelum *fully-connected* layer pertama atau $VGG_{5.4}$, dengan struktur arsitektur sebagaimana Gambar 2.10 kolom E.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Gambar 2.10. Struktur layer VGG19 pada kolom E
(Simonyan and Zisserman, 2015)

Memanfaatkan hasil ektraksi fitur VGG19 untuk membandingkan fitur citra *super-resolution* dengan citra *high-resolution*, dikarenakan MSE loss tidak dapat mengatasi *high frequency content* yang menghasilkan citra terlalu halus (Ledig et al., 2016). VGG *content*

loss ditunjukan dengan Persamaan 2.4, dengan *feature map* didapat pada indeks konvolusi j dan sebelum *maxpooling* ke i, dengan $G_{\theta_G}(I^{LR})$ sebagai citra hasil generasi dari *generator* dan I^{HR} adalah citra *high-resolution*, dan $W_{i,j}H_{i,j}$ sebagai dimensi dari *feature map*.

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\varphi_{i,j}(I^{HR})_{x,y} - \varphi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y}\right)^2 \quad \text{.....(2.4)}$$

Dengan *Adversarial loss* ditunjukan pada Persamaan 2.5, dengan $D_{\theta_D}\left(G_{\theta_G}(I^{LR})\right)$ sebagai nilai dari kemungkinan citra dianggap sebagai citra *high-resolution*.

$$l^{SR}_{Gen} = \sum_{n=1}^{N} -log\, D_{\theta_D}\left(G_{\theta_G}(I^{LR})\right) \text{.....(2.5)}$$

2.7. Operasi Pengolahan Citra Digital

Beberapa operasi untuk pengolahan citra dalam penelitian ini, diantaranya adalah,

2.7.1. *Grayscaling*

Pengubahan warna citra plat nomor kendaraan RGB menjadi *grayscale* dengan menggunakan Persamaan 2.6 menurut (Kumar, Mishra and Nandan, 2020),

$$Y = \left((0{,}299) \times R\right) + \left((0{,}587) \times G\right) + \left((0{,}114) \times B\right) \text{.....(2.6)}$$

Keterangan:

Y : output *grayscale*
R : nilai *red* pada citra RGB
G : nilai *green* pada citra RGB
B : nilai *blue* pada citra RGB

2.7.2. *Otsu's thresholding and binarization*

Otsu's method adalah metode untuk meminimalkan varians dalam kelas tertimbang. Dengan maksud mencari nilai minum di dalam suatu varians kelas dan atara varians kelas, untuk memisahkan latar belakang dengan objek. Nilai minimum di dalam suatu varians kelas dihitung dengan Persamaan 2.7 sebagaimana (Sham et al., 2021),

$$\sigma_w^2 = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \text{.....(2.7)}$$

Pada Persamaan 2.7 mencari nilai t yang berada diantara dua nilai tertinggi, sehingga didapati nilai minimum dari varians kedua kelas. Dimana, mencari nilai minimum pada intra kelas sama dengan mencari nilai maksimum pada varians atar kelas, yang dapat dihitung dengan Persamaan 2.8 (Sham et al., 2021).

$$\sigma_b^2 = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2 \text{.....(2.8)}$$

Keterangan:

ω_0 : *background weight*
ω_1 : *foreground weight*
μ_0 : rata – rata *background*
μ_1 : rata – rata *foreground*

2.7.3. *Resizing*

Operasi *resize* atau pengubahan ukuran pada citra umumnya menggunakan metode *interpolation* terhadap pixel citra, operasi tersebut dapat mengakibatkan *artifact* dan *distortion* pada citra (Priyanka C. Dighe, 2014), hal tersebut baik untuk dapat mensimulasikan citra *low-resolution* sesungguhnya dan dapat digunakan sebagai data sintetis untuk evaluasi *model* algoritma SRGAN.

2.7.4. *Cropping*

Operasi *cropping* atau pemotongan area tertentu pada citra digunakan sebagai data input untuk proses *trainning* model algoritma SRGAN. Penerapan *cropping* diterapkan sesuai pada (Ledig et al., 2016). Dengan proses *cropping* dilakukan secara acak, yaitu pemotongan berukuran 96x96 pixel pada citra *high-resolution* dan 24x24 pixel pada citra *low-resolution* atau sebanding dengan skala perbesaran 4x.

2.7.5. *Rotation*

Operasi augmentasi rotasi pada citra ditambahkan untuk memperbanyak variasi dataset yang ada. Yaitu proses memutar citra dataset sebesar 90° maupun kelipatanya atau sama sekali tidak dilakukan proses rotasi tersebut.

2.7.6. *Flipping*

Operasi augmentasi flipping atau pembalikan citra ditambahkan untuk memperbanyak variasi dataset yang ada. Yaitu proses pertukaran citra berdasarkan pada sumbu y atau secara horizontal.

2.7.7. *Image Noise*

Operasi penambahan noise atau variasi acak pada nilai kecerahan dan warna yang ada pada pixel citra digunakan untuk mensimulasikan keadaan sebenarnya.

2.7.8. *Min-max normalization*

Proses *min-max normalization* atau *min-max scaling* pada citra dilakukan untuk mengubah nilai pixel dari skala [0, 255] menjadi [0, 1], berguna agar rentang data tidak terlalu lebar dan variatif. Proses *min-max normalization* dilakukan menggunakan Persamaan 2.9 (Han, Kamber and Pei, 2012).

$$x' = \frac{x - min_x}{max_x - min_x} \quad (2.9)$$

Keterangan:

x : nilai pixel awal, [0, 255]
x' : nilai pixel hasil normalisasi, [0, 1]
min_x : nilai x minimum awal, [0]
max_x : nilai x maksimum awal, [255]

2.8. *Peak Signal-to-Noise Ratio*

Peak Signal-to-Noise Ratio (PSNR) digunakan sebagai evaluasi kuantitatif dari hasil proses *super-resolution*, yaitu kalkulasi generasi citra *low-resolution* yang telah rusak karena *noise*. PSNR didapat berdasarkan perhitungan rasio antara nilai maksimum suatu *pixel* dengan nilai *noise* yang mempengaruhi citra beresolusi tinggi. Semakin tinggi nilai PSNR,

semakin baik kualitas citra *Super-Resolution* yang dihasilkan. Persamaan PSNR dideskribsikan sebagaimana Persamaan 2.10 (Symolon and Dagli, 2021),

$$PSNR = 10.log_{10}\left(\frac{N^2}{MSE}\right) = 20.log_{10}\left(\frac{N}{\sqrt{MSE}}\right) \quad (2.10)$$

Keterangan:

N : *dynamic range* dari sebuah *pixel*, untuk warna 8-bit, N = 255.

Dengan *Mean Squared Error* (MSE) didapat berdasar Persamaan 2.11 (Yang et al., 2019),

$$MSE = \frac{1}{K}\|x - y\|_F^2 \quad (2.11)$$

Keterangan:

K : Banyaknya pixel

x : Matriks citra *high-resolution*

y : Matriks citra *super-resolution*

$\| . \|_F^2$: F*robenius norm* atau *Euclidean norm*.

2.9. *Structural Similarity Index*

Berdasar kemiripan antara dua citra, *Structural Similarity Index* (SSIM) mengevaluasi bahwa perbandingan posisi *pixel* pada dua citra yang berdekatan secara spasial bisa jadi menggambarkan bentuk objek yang sama. SSIM dinyatakan dengan nilai desimal dengan rentang antara 1 dan 0, dimana 1 menyatakan kemiripan sempurna. SSIM didapat dengan Persamaan 2.12 (Symolon and Dagli, 2021).

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (2.12)$$

Keterangan:

μ_x dan μ_y : rata-rata citra x dan y

σ_x^2 dan σ_y^2 : varians citra x dan y

σ_{xy} : kovarians (x, y)

Dengan c_1 dan c_2 fungsi penyeimbang jika *denominator* bernilai terlalu kecil, dinyatakan dengan Persamaan 2.13 dan 2.14 (Bakurov et al., 2022),

$$c_1 = (K_1 L)^2 \quad (2.13)$$

$$c_2 = (K_2 L)^2 \quad (2.14)$$

Keterangan:

L : *dynamic range* dari sebuah *pixel*, untuk warna 8-bit, N = 255.

K_1 : 0.01 (nilai default)

K_2 : 0.03 (nilai default)

2.10. *Confusion Matrix*

Dalam penggunaannya *confusion matrix* atau *error matrix* digunakan untuk menggambarkan tingkat performa akurasi dari sebuah algoritma *machine learning*

(Stehman, 1997). Dengan setiap baris mewakili nilai yang sebenarnya dan untuk setiap kolom mewakili hasil prediksi dari sistem. Dengan *confusion matrix* dapat dilihat pada Tabel 2.1 (Fawcett, 2006).

Tabel 2.1. *Confusion Matrix*

		Nilai Prediksi	
		***Positive* (Y)**	***Negative* (N)**
Nilai Asli	***Positive* (P)**	*True Positive* (**TP**)	*False Negative* (**FN**)
	***Negative* (N)**	*False Positive* (**FP**)	*True Negative* (**TN**)

Berdasar Tabel 2.1 didapati empat kemungkinan hasil (Fawcett, 2006). Pertama, jika nilai prediksi menghasilkan *positive* (Y) dan *positive* (P) pada nilai asli maka dihitung sebagai *True Positive* (TP), kedua, jika nilai asli bernilai *negative* (N) maka dihitung sebagai *False Positive* (FP). Ketiga, pada nilai prediksi *negative* (N) dan nilai asli terhitung *positive* (P) maka dihitung sebagai *False Negative* (FN), keempat, jika pada nilai asli terhitung *negative* (N) maka hasil dihitung sebagai *True Negative* (TN). Berdasarkan nilai tersebut didapati persamaan untuk menghitung *recall* yaitu rasio nilai asli *positive* yang diprediksi bernilai *positive*, dapat dihitung pada Persamaan 2.15 menurut (Powers, 2020).

$$recall = \frac{TP}{TP+FN} \quad (2.15)$$

Dan dengan nilai akurasi atau rasio banyaknya nilai prediksi benar pada nilai asli positif (TP) dan prediksi benar pada nilai asli negatif (TN), nilai akurasi didapat dengan Persamaan 2.16 (Powers, 2020).

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.16)$$

Nilai F1-*Score* adalah nilai akurasi prediksi yang dihitung berdasarkan nilai *precission* dan nilai *recall*, dengan nilai *precission* didapati berdasarkan hasil prediksi positif (TP) dibagi dengan jumlah semua hasil positif termasuk yang tidak teridentifikasi dengan benar (TP+FP), maka didapati persamaan untuk mencari nilai F1-*Score* sebagaimana Persamaan 2.17 (Fawcett, 2006).

$$F1_{Score} = \frac{2TP}{2TP+FP+FN} \quad (2.17)$$

2.11. Penelitian Terkait

Penelitian-penelitian sebelumnya mengenai deteksi dan pengenalan plat nomor kendaraan menggunakan metode, jenis, dan jumlah dataset yang berbeda. Kebutuhan dan masalah yang diangkat mempengaruhi penentuan dataset oleh setiap peneliti. Pada (Jia et al., 2014) menggunakan 1000 citra plat nomor kendaraan yang diambil secara langsung dengan berbagai kombinasi latar belakang dan kondisi agar menyerupai kodisi saat penerapan di lapangan. (Hamdi, Chan and Koo, 2021) juga menggunakan dataset milik mereka sendiri dan dipadukan dengan dataset hasil tangkapan citra pada kamera analog. Pada (Gunawan et al., 2021; Ledig et al., 2016) memilih untuk menggunakan dataset yang

sudah ada dan telah digunakan pada penelitian sebelumnya, yaitu, 5000 - 15000 plat nomor ber-label dan binarized plat nomor kendaraan Indonesia Tel-U License Plate Data-Set V1 dan juga citra acak sejumlah 350 ribu dari ImageNet. Menggunakan dataset yang tersedia secara online (Sham et al., 2021), 8000 citra kendaraan roda empat beserta *bounding-box* pat nomor dipilih dalam penelitiannya.

Beberapa peniliti juga menggunakan dataset sintetis atau buatan dalam proses latihnya. Dalam penelitian (Ledig et al., 2016; Hamdi, Chan and Koo, 2021) pada generasi citra *Super-Resolution* (SR), menggunakan dataset sintetis dari citra *High-Resolution* (HR) yang telah dilakukan *bicubic downsampling* skala 4 maupun *image-to-image style translation network* menggunakan CycleGAN, yang kemudian menghasilkan citra *Low-Resolution* (LR) yaitu dua sampai empat kali lebih kecil skala dan kualitasnya dari citra asli atau HR. Citra LR tersebut dibuat sedemikian rupa untuk menyimulasikan *noise* dan *distorsi* yang ada pada kondisi lapangan tanpa harus mengambil data secara langsung.

Metode CNN digunakan (Imaduddin et al., 2019) untuk mendeteksi plat nomor kendaraan secara *realtime*. Proses deteksi plat nomor kendaraan akan dieksekusi ketika kendaraan telah berada pada area zebra cross. Data *live streaming* akan diekstraksi untuk setiap *frame* nya sebagai input pada CNN. Pada penerapannya model berhasil mendapatkan tingkat akurasi validasi sebesar 80 persen, namun beberapa plat nomor kendaraan gagal dideteksi dikarenakan kurangnya pencahayaan, tekstur warna dari plat nomor serupa dengan kendaraan, maupun plat nomor tertutup oleh kendaraan lain. Deteksi plat nomor kendaraan secara realtime juga dilakukan oleh (Gunawan et al., 2021). Menggunakan YOLOACT untuk deteksi dan MobileNetV2 untuk pengenalan karakter. *Frame sampling* digunakan untuk meningkatkan performa dari YOLOACT. Beberapa *pre-processing* dilakukan dalam proses segmentasi, yaitu, *Histogram equalization*, *Otsu's thresholding*, tiga kali *erosion kernel* 3x3, dan *closing & opening kernel* 3x3. Dilakukan beberapa set percobaan, dimana hasil terbaik pada tingkat *frame sampling* 250ms, dengan peningkatan performa YOLOACT sebesar 78% menjadi 83%. Dengan rata-rata akurasi MobileNetV2 pada 100 epoch pada rasio split data 0,1. Ditemukan bahwa penggunaan *frame sampling* dapat mengurangi waktu deteksi, namun *frame sampling* yang tinggi dapat menyebabkan kegagalan dalam deteksi plat nomor. YOLO kembali digunakan pada (Sham et al., 2021), dengan menggunakan Dengan YOLO versi 4 milik (Bochkovskiy, Wang and Liao, 2020). Dilakukan ekstraksi sub-citra berdasarkan hasil *bounding-box* YOLO. Beberapa *pre*-processing diterapkan pada hasil ekstraksi sub-citra, yaitu, *Grayscaling*, *Gaussian Smoothing*, *Thresholding and binarization using Otsu's method, Morphological Transformations, contours*, dan *segmentation* digunakan untuk memisahkan setiap karakter yang ada pada plat nomor. Operasi *bitwise-not* dilakukan untuk membalik warna karakter menjadi hitam dan latar belakang menjadi putih agar sesuai dengan kebutuhan input citra pada *Tesseract* (*Open Source*) OCR *engine* (Smith and Podobny, 2021). Dilakukan uji coba pada 30 citra, didapatkan 92% akurasi pada deteksi plat nomor kendaraan, dan 81% akurasi pada pengenalan karakter menggunakan *Tesseract* (*Open Source*) OCR *engine*.

Peningkatan kualitas citra menggunakan konsep *Generative Adversarial Network* (GAN) milik (Goodfellow et al., 2014) dilakukan oleh (Ledig et al., 2016). Dalam penelitiannya, GAN digunakan untuk proses *Super-Resolution* (SR) atau yang ia sebut

sebagai *Super-Resolution Generative Adversarial Network* (SRGAN). Dimana ResNet dipakai sebagai dasar struktur pada *Generator* untuk melakukan generasi citra SR yang berukuran empat kali lebih besar dari citra *Low-Resolution* (LR). *Mean Opinion Score* (MOS) *testing*, PSNR dan SSIM digunakan untuk mengevaluasi hasil SR. Pada hasil SSIM dan PSNR, SRGAN memiliki nilai menengah sampai terburuk. Namun pada hasil MOS testing, memiliki nilai MOS sangat mirip (bernilai 4, skala 1-5, 5 Sangat mirip atau citra HR itu sendiri) dalam perbandingan dengan citra HR. Dibandingkan dengan beberapa algoritma SR yang lain, yaitu, bicubic, DRCN, SRCNN, ESPCN, dan SRResNet (rata-rata bernilai 1-2, skala 1-5). Dalam penelitiannya ditemukan bahwa hasil evaluasi menggunakan PSNR dan SSIM menunjukan hasil yang terbalik. Dibandingkan evaluasi menggunakan MOS yang menunjukan hasil SRGAN memiliki kemiripan paling tinggi dibandingkan algoritma lain.

Plat nomor yang ditangkap melalui kamera CCTV cenderung memiliki kualitas citra resolusi yang rendah dengan banyak *noise*. Dalam penelitiannya (Hamdi, Chan and Koo, 2021) mencoba untuk memperbaiki citra deteksi plat nomor kendaraan dengan *Image Enhancement* dan *Super Resolution* menggunakan *Generative Adversarial Networks* (GAN) (Goodfellow et al., 2020) yang telah dimodifikasi menjadi *Double Generative Adversarial Networks for Image Enhancement and Super Resolution* (D_GAN_ESR). Dimana dua GAN digunakan secara berurutan, GAN pertama refs digunakan untuk pengurangan *noise* dan *blur*, kedua pada pengaplikasian *Super-Resolution* (SR). Sama seperti (Ledig et al., 2016) , dimana *output* citra akan menjadi empat kali lebih besar daripada citra input. Dalam evaluasi hasil SR, D_GAN_ESR memiliki hasil PSNR terbaik (29,5) dibandingkan dengan SRGAN(28,3), EDSR(29,0) dan ESRGAN(27,7). Dengan nilai SSIM (0,27) di bandingan SRGAN(0,204), EDSR(0,271) dan ESRGAN(0,067). Dalam evaluasi LPR menggunakan dua algoritma OCR, yaitu, *Tesseract* (*Open Source*) OCR *engine* (Smith and Podobny, 2021) dan easyOCR (*Open Source*) (JaidedAI, 2021; Shi, Bai and Yao, 2017; Baek et al., 2019), memiliki penurunan *error rate* dari (5.57) sebelum dan (1.58) sesudah penerapan D_GAN_ESR.

Berdasarkan tinjauan literatur dan penelitian sebelumnya, dalam penelitian ini akan diterapkan algoritma YOLOv4 (Bochkovskiy, Wang and Liao, 2020) untuk deteksi objek dan lokalisasi plat nomor kendaraan. Dengan algoritma *super-resolution* milik (Ledig et al., 2016), yaitu *Super-Resolution Generative Adversarial Network* (SRGAN), akan digunakan untuk melakukan proses *super-resolution* pada citra *low-resolution* untuk plat nomor kendaraan. *Library open source Optical Character Recognition* (OCR), yaitu Easy OCR milik (JaidedAI, 2021) akan digunakan untuk mengevaluasi hasil citra *super-resolution*. Dua metode *pre-processing*, yaitu *Grayscaling* dan *Thresholding and binarization using Otsu's method* akan digunakan pada citra sebelum dilakukan proses OCR.

Tabel 2.2. Rangkuman Penelitian Terkait

Penulis	Dataset	Pre-Processing	Metode	Temuan
(Ledig et al., 2016)	Random 350 000 ImageNet set. Citra LR di buat dengan melakukan bicubic downsampling skala 4 dari citra High-Resolution (HR)	Bicubic downsampling skala 4 dari citra High-Resolution (HR)	*Super-Resolution Generative Adversarial Network* (SRGAN). Dimana ResNet dipakai sebagai dasar struktur pada *Generator* untuk melakukan generasi citra SR dari citra *Low-Resolution* (LR).	Pada hasil SSIM dan PSNR SRGAN memiliki nilai menengah sampai terburuk. Hasil MOS testing, SRGAN memiliki nilai MOS sangat mirip dengan citra HR. PSNR dan SSIM kurang akurat dalam pengukuran hasil SR.
(Imaduddin et al., 2019)	±1000 citra milik mereka sendiri dari berbagai kombinasi latar belakang dan kondisi	-	*pretrained* CNN model milik (Jia et al., 2014)	Beberapa plat nomor kendaraan gagal dideteksi dikarenakan kurangnya pencahayaan, tekstur warna dari plat nomor serupa dengan kendaraan, maupun plat nomor tertutup oleh kendaraan lain.
(Gunawan et al., 2021)	5000-15000 plat nomor ber-label dan *binarized* plat nomor kendaraan Indonesia Tel-U License Plate Data-Set V1.	*Histogram equalization*, *Otsu's thresholding*, tiga kali *erosion kernel* 3x3, dan *closing & opening kernel* 3x3.	YOLOACT untuk deteksi dan MobileNetV2 untuk pengenalan karakter.	*frame sampling* yang tinggi dapat menyebabkan kegagalan dalam deteksi plat nomor.
(Hamdi, Chan and Koo, 2021)	Downsampling dan distorsi HR dengan CycleGAN dan DualGAN	*image-to-image style translation network* menggunakan CycleGAN	Dual GAN for Image Enhancement and Super Resolution (D_GAN_ESR) Menggunakan 2 tahapan GAN, Image Enhancement GAN kemudian Super Resolution GAN.	D_GAN_ESR memiliki akurasi lebih baik dibandingkan SRGAN, EDSR, dan ESRGAN. Namun lama dan lebih berat pada proses training dan juga tingkat PSNR yang lebih rendah dari arsitektur lain.

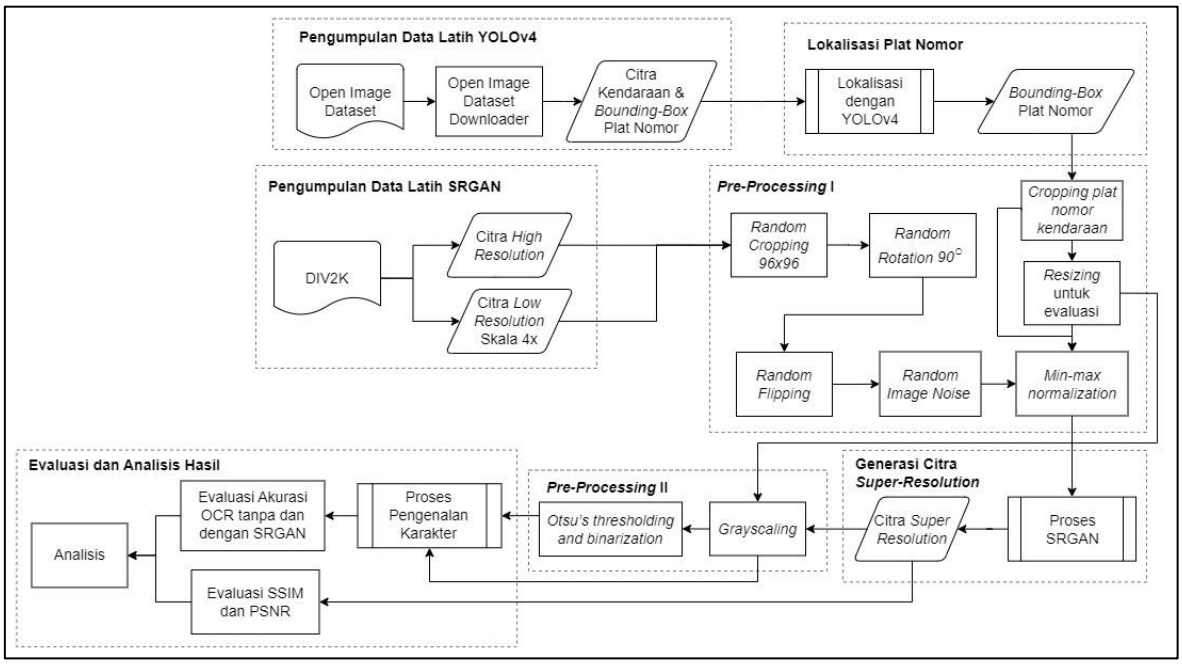
Tabel 2.3. Rangkuman Penelitian Terkait (Lanjutan)

(Sham et al., 2021)	Menggunakan dataset berjumlah 8000 citra kendaraan roda empat beserta bounding-box plat nomor	*Grayscaling*, *Gaussian Smoothing*, *Thresholding and binarization using Otsu's method, Morphological Transformations*, *contours*, dan *segmentation.*	YOLO versi 4 dan *pre-processing* sebelum citra dilakukan OCR menggunakan *Tesseract* OCR *engine.*	Dilakukan uji coba pada 30 citra, didapatkan 92% akurasi pada deteksi plat nomor kendaraan, dan 81% akurasi pada pengenalan karakter.
(Bochkovskiy, Wang and Liao, 2020)	MS COCO dataset, dengan *augmentasi* standar (*crop, rotation,flip,hue,saturation*) dan tambahan lain seperti, *self-adversarial training*, *cutout*, *mixup*, *cutmix*, *mosaic*, dan *blurring*.	-	YOLOv4, kombinasi *bag of special* dan *bag of freebies* untuk pengolahan dataset dan *detector*, YOLOv3 *anchor based* sebagai struktur *head*	YOLOv4 memiliki kecepatan inferensi frame per detik paling cepat dengan akurasi tertinggi pada kecepatan inferensi 60-90 FPS dengan presisi 64-66%.

BAB III
METODOLOGI PENELITIAN DAN PENGEMBANGAN SISTEM

3.1. Metodologi Penelitian

Pada penelitian ini metodologi penelitian dilakukan secara sistematis, adapun ilustrasi tahapan pengujian secara keseluruhan dapat dilihat pada Gambar 3.1. Diawali dengan pengumpulan data latih yang akan digunakan untuk proses *transfer learning* YOLOv4 dan SRGAN. Dataset yang diperoleh dari DIV2K akan dilakukan *pre-processing* sebelum digunakan sebagai data latih dari SRGAN.



Gambar 3.1 Ilustrasi alur metodologi penelitian

Lokalisasi plat nomor menggunakan YOLOv4 akan menghasilkan koordinat *bounding-box* pada citra. Hasil lokalisasi plat nomor dan *super-resolution* dari SRGAN akan masuk kedalam tahapan *pre-processing* yang ke dua, yang berguna untuk meningkatkan lebih lanjut hasil akurasi pengenalan karakter. Proses pengenalan karakter dilakukan untuk mengevaluasi penerapan SRGAN pada proses pengenalan plat nomor kendaraan.

3.1.1. Pengumpulan Data

Pada penelitian ini digunakan dataset sekunder yang diambil dari dua situs penyedia dataset secara terbuka, yaitu, DIV2K dan *Google Open Image Dataset* versi 6 (OIDv6). Dataset OIDv6 akan digunakan untuk proses *training* YOLOv4. Beberapa citra yang didapat dari dataset OIDv6 dapat dilihat pada Gambar 3.2, dimana dalam Gambar 3.2 terdapat citra bersama dengan visualisasi koordinat *bounding-box* plat nomor yang dihasilkan berdasarkan labeling koordinat pada file teks bersama dengan data citra.

Gambar 3.2. Contoh *Google Open Image Dataset* versi 6

Dataset yang pertama berasal dari *Google Open Image Dataset* versi 6, dilakukan pembagian data *training* dan *validation*, dengan pembagian sebagaimana Tabel 3.1.

Tabel 3.1. Pembagian data *training* dan *testing* YOLOv4 dari OIDv6

Nama	**Jenis**	**Jumlah**
Vehicle registration plate	*Training*	5368
Vehicle registration plate	*Validation*	386
Vehicle registration plate / Data Baru Pengujian Manual	*Testing*	20

Dataset kedua yang diperoleh dari DIV2K pada (Agustsson and Timofte, 2017), digunakan sebagai data latih SRGAN, yaitu berupa data citra dengan kualitas 2K atau 2560x1440 pixel dengan pasangan *high-resolution* dengan citra *low-resolution* berskala 4x lebih kecil dari pada citra HR. Pengumpulan data untuk data latih proses SRGAN dilakukan secara terpisah. Beberapa contoh data dari DIV2K dapat dilihat pada Gambar 3.3.

Gambar 3.3. Contoh dataset DIV2K

Data yang didapat kemudian dilakukan *pre-processing random crop, rotation, flipping,* dan *image noise*, dengan proses selanjutnya dapat dilihat pada subab 3.1.4. Proses SRGAN. Hasil dari proses SRGAN yaitu berupa citra *super-resoulution*. Dengan pembagian data sebagai data *training* dan *validation* dengan rasio 8 banding 1, dengan pembagian sebagaimana Tabel 3.2.

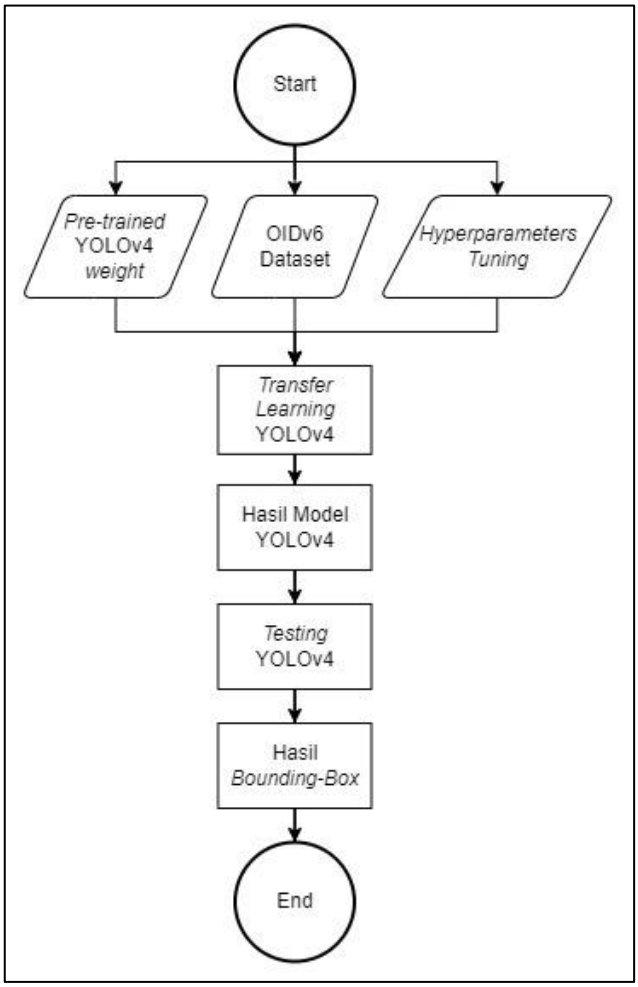
Digunakan data citra plat nomor tambahan yang didapat dari Kaggle penyedia dataset terbuka maupun pencarian secara manual untuk digunakan sebagai data uji baru sintetis. Sebanyak 20 citra akan digunakan dengan membuat pasangan citra *low-resolution* yang berukuran 32x64 pixel dan *high-resolution* berukuran 128x256 pixel untuk masing-masing data manual yang diperoleh. Pembuatan pasangan citra dilakukan menggunakan proses *resizing* pada tahapan *pre-processing* I.

Tabel 3.2. Pembagian data *training* dan *testing* SRGAN

Nama	Jenis	Jumlah
Citra *low-resolution*	*Training*	800
Citra *high-resolution*	*Training*	800
Citra *low-resolution*	*Validation*	100
Citra *high-resolution*	*Validation*	100
Citra *low-resolution* / Data Baru Pengujian Manual	*Testing*	20
Citra *high-resolution* / Data Baru Pengujian Manual	*Testing*	20

3.1.2. Proses Lokalisasi Plat Nomor

Data OIDv6 selanjutnya digunakan untuk proses *transfer learning* YOLOv4 untuk membuat model baru yang dapat melakukan lokalisasi plat nomor dengan hasil akhir yaitu *bounding-box* plat nomor, alur proses dapat dilihat pada Gambar 3.4.



Gambar 3.4 Ilustrasi alur lokalisasi plat nomor dengan YOLOv4

Untuk dapat melakukan *transfer learning* dari arsitektur YOLOv4 beberapa penyesuaian *hyperparameter* dibutuhkan, konfigurasi *hyperparameter* pada penelitian ini sesuai dengan (Bochkovskiy, Wang and Liao, 2020), dengan *class* berjumlah satu yaitu *license_plate*, *max-batches* didapat dari jumlah kelas dikalikan 2000 dengan hasil minimum 6000, dikarenakan pada penelitian ini hanya terdapat satu kelas makan *max-batches*

menggunakan nilai 6000. Untuk *steps* didapat berdasarkan 80% dan 90% dari *max-batches* yaitu 4800 dan 5400. Dengan nilai *filter* 18 didapat dari nilai *class* ditambah 5 kemudian dikalikan dengan 3. Nilai 608 digunakan pada *width* dan *height* sebagai ukuran *input* pada arsitektur untuk dapat menghasilkan akurasi yang baik. Agar proses *trainning* tidak terlalu berat dalam komputasi dengan menggunakan *google collab*, digunakan nilai 64 dan 16 untuk *batch* dan *subdivisions*. Konfigurasi *hyperparameter* dapat dilihat pada Tabel 3.3.

Tabel 3.3. Konfigurasi *hyperparameter* YOLOv4

Nama Parameter	**Nilai**
Batch	64
Subdivisions	16
Max-batches	6000
Width	416
Height	416
Steps	4800, 5400
Class	1
Convolution YOLO layer Filters	18

Hasil dari proses lokalisasi plat nomor adalah koordinat *bounding-box* dan juga nilai ukuran dari *bounding-box*. Akan dilakukan *cropping* pada tahapan *pre-processing* I berdasarkan koordinat mutlak dan nilai ukuran yang didapat untuk memperoleh citra yang berisikan plat nomor saja. Beberapa parameter hasil yang akan digunakan adalah '*relative_coordinates*' dan juga nilai *confidence* setiap *object* dalam satu *frame.* Berdasarkan nilai koordinat relatif hasil dari prediksi, digunakan Algoritma 4 dalam mencari koordinat nilai mutlak atau nilai pixel sebenarnya.

Algoritma 4 : Algoritma mencari koordinat pixel asli pada citra

```
INIT imgHeight, imgWidth, center_x, center_y, yoloWidth, yoloHeight
BEGIN
   CALCULATE x1 = ( center_x – ( yoloWidth / 2 )) * imgWidth
   CALCULATE x2 = ( center_x + ( yoloWidth / 2 )) * imgWidth
   CALCULATE y1 = ( center_y – ( yoloHeight / 2 )) * imgHeight
   CALCULATE y2 = ( center_y + ( yoloHeight / 2 )) * imgHeight
END
```

Diketahui nilai relatif 'center_x' atau b_x dan 'center_y' atau b_y sebagai koordinat titik tengah *bounding-box*, dan juga nilai relatif dari tinggi dan lebar yaitu 'yoloWidth' atau b_w dan 'yoloHeight' atau b_h dengan ilustrasi sebagaimana Gambar 3.5 (A). Koordinat pixel asli pada citra didapat berdasarkan Algoritma 4. Untuk menemukan nilai dari x_1 dilakukan pengurangan pada nilai '*center_x*' atau titik tengah dari *bounding-box* yang terdeteksi, '*center_x*' dapat dilihat pada Gambar 3.5 (A) titik merah ditengah kotak biru, dengan setengah lebar relatif dari *bounding-box* YOLOv4, dikarenakan '*center_x*' dan lebar dari YOLOv4 berupa nilai relatif terhadap citra maka dilakukan pengkalian dengan lebar asli pada citra yang dilakukan lokalisasi plat nomor. Pada x_2 nilai '*center_x*' ditambahkan dengan setengah lebar relative, kemudian untuk y_1 dan y_2 dilakukan proses yang serupa , yaitu dengan menggunakan nilai tinggi relatif dan asli dari citra sebagai pengganti lebar.

Dengan diketahui nilai berdasarkan hasil prediksi YOLO4 yaitu, ‘center_x’ adalah 0.656, ‘center_y’ bernilai 0.425, ‘yoloWidth’ dengan nilai 0.382, dan ‘yoloHeight’ bernilai 0.261, atau sesuai dengan Gambar 3.5 (B), maka dapat digunakan nilai nilai tersebut untuk mencari nilai pixel sebenarnya atau nilai mutlak menggunakan Algoritma 4, perhitungan selanjutnya dapat diketahui sebagai berikut,

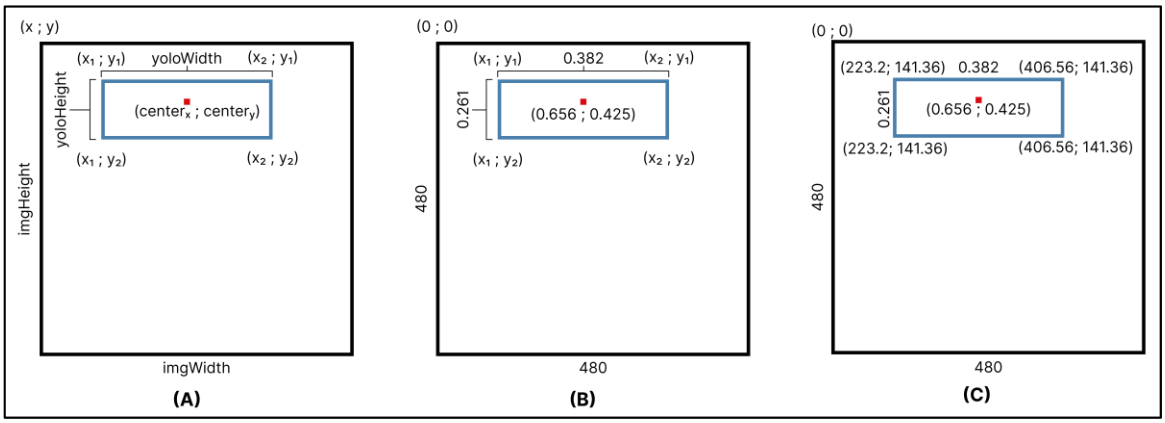
$$x_1 = \left(center_x - \frac{yoloWidth}{2} \right) \times imgWidth = (0.656 - \frac{0.382}{2}) * 480 = 223.2$$

$$x_2 = \left(center_x + \frac{yoloWidth}{2} \right) \times imgWidth = (0.656 + \frac{0.382}{2}) * 480 = 406.56$$

$$y_1 = \left(center_y - \frac{yoloHeight}{2} \right) * imgHeight = (0.425 - \frac{0.261}{2}) * 480 = 141.36$$

$$y_2 = \left(center_y + \frac{yoloHeight}{2} \right) * imgHeight = (0.425 + \frac{0.261}{2}) * 480 = 266.64$$

Dengan keseluruhan hasil perhitungan selanjutnya dapat dilihat pada Gambar 3.5 (C).



Gambar 3.5. Ilustrasi hasil *bounding-box* YOLOv4

Berdasarkan hasil koordinat sebenarnya pada Gambar 3.5 (C), maka nilai nilai yang didapat akan digunakan selanjutnya pada proses *pre-processing* I untuk kemudian dilakukan proses ekstraksi plat nomor, yaitu proses *cropping* citra asli untuk mendapatkan citra yang hanya mengandung area plat nomor kendaraan saja.

3.1.3. *Pre-processing* I

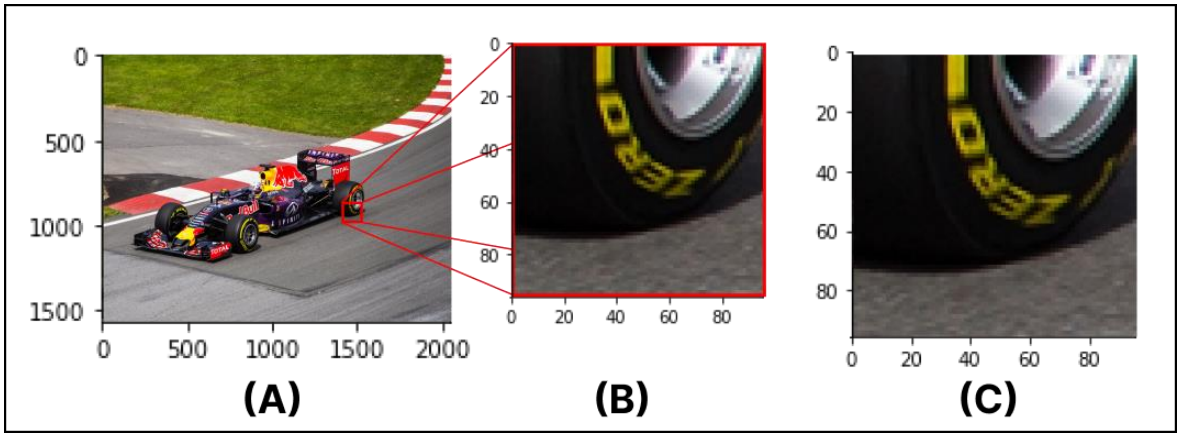
Dataset yang diperoleh dari DIV2K pada (Agustsson and Timofte, 2017) akan dilakukan *pre-processing* sebelum digunakan sebagai data latih dari SRGAN. Proses *cropping* dilakukan secara acak dengan ukuran 96x96 pixel diekstraksi satu kali secara acak pada satu citra input *high-resolution* . Dengan rasio skala 4x pada citra *low-resolution*, yaitu akan memiliki ukuran citra 24x24.

Pengambilan area citra secara acak dilakukan dengan generasi koordinat random menggunakan fungsi dari tensorflow yaitu random.uniform, dimana pengacakan dilakukan secara uniform atau setiap angka yang ada memiliki kemungkinan kemunculan yang sama. Random *cropping* didapati berdasarkan Algoritma 5.

Algoritma 5 : Algoritma *random cropping*

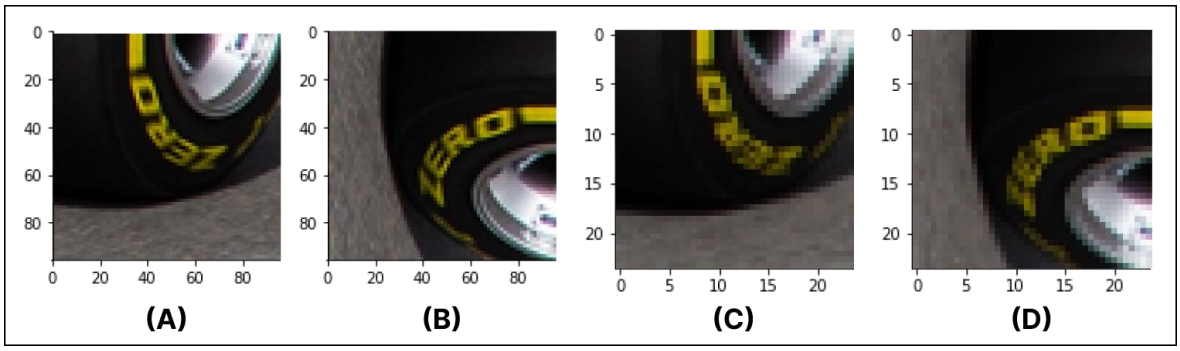
```
INIT lrImg, hrImg
SET hrCropSize TO 96 AND lrCropSize TO 24
BEGIN
    GET RANDOM UNIFORM lr_x WITH MAXVALUE SET TO lrImage_shape_x - lrCropSize
    GET RANDOM UNIFORM lr_y WITH MAXVALUE SET TO lrImage_shape_y - lrCropSize
    CALCULATE hr_x = lr_x * 4
    CALCULATE hr_y = lr_y * 4

    SAVE lrImg[lr_x : lr_x + lrCropSize, lr_y : lr_y + lrCropSize] TO lrCrop
    SAVE hrImg[hr_x : hr_x + hrCropSize, hr_y : hr_y + hrCropSize] TO hrCrop
END
```

Ilustrasi hasil *random cropping* dapat dilihat pada Gambar 3.6, dengan ilustrasi Gambar 3.6 (A) adalah dataset citra asli, Gambar 3.6 (B) adalah hasil *random cropping*, kemudian digunakan sebagai input data latih sintetis *high-resolution*, yaitu berukuran 96x96 pixel, Gambar 3.6 (C) adalah hasil *cropping* sebagai data latih sintetis untuk *low-resolution.*



Gambar 3.6. Ilustrasi *random cropping*

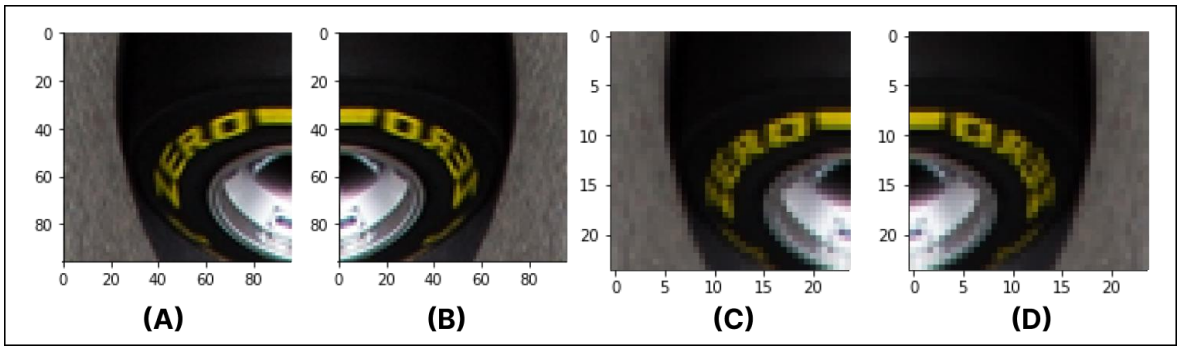
Hasil *random cropping* akan diproses pada pengenaan *random rotation* atau proses pemutaran secara acak sebesar 90° maupun kelipatanya atau sama sekali tidak dilakukan proses rotasi tersebut. Dengan input berasal dari *cropping* citra HR yaitu Gambar 3.6 (B) atau pada Gambar 3.7 (A), dan dengan hasil *random rotation* citra HR pada Gambar 3.7 (B). Pada hasil *cropping* citra LR yaitu Gambar 3.6 (C) atau ditunjukan pada Gambar 3.7 (C) dan hasil *random rotation* pada Gambar 3.7 (D).



Gambar 3.7. Ilustrasi citra terkenai proses *rotation*

Dengan ilustrasi pengenaan rotasi 90° sebanyak tiga kali dapat dilihat pada Gambar 3.7. Proses randomisasi masih menggunakan random.uniform dan *rotation* menggunakan fungsi image.rot90 pada tensorflow.

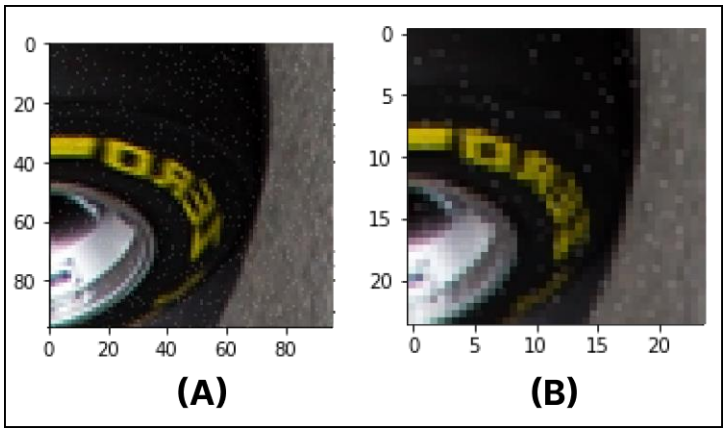
Setelah melalui proses *random rotation* maka citra akan diproses dalam pengenaan *random flipping* atau proses pencerminan secara acak terhadap sumbu y, dengan kemungkinan hasil yaitu, tidak terkenai *fliping* atau terkenai *flipping*, dengan hasil terkenai *flipping* dapat dilihat pada Gambar 3.8. Proses randomisasi menggunakan random.uniform dan rotation dengan fungsi image.rot90 pada tensorflow.



Gambar 3.8. Ilustrasi citra terkenai proses *flipping*

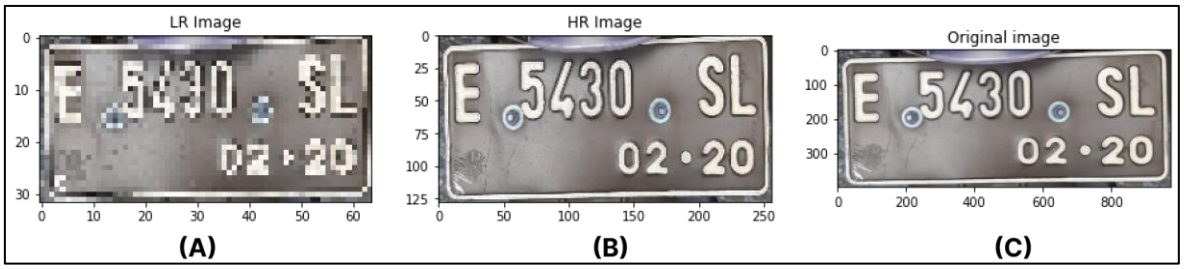
Dengan citra *high-resolution* hasil *rotation* pada Gambar 3.8 (A) dan citra HR terkenai *flipping* pada Gambar 3.8 (B). Pada Gambar 3.8 (C) adalah citra hasil *rotation* dari citra *low-resolution*, dan Gambar 3.8 (D) adalah citra hasil *flipping* citra LR.

Proses *image noise* secara acak dilakukan dengan menggunakan fungsi pada tensorflow yaitu image.random_jpeg_quality, dengan hasil dapat dilihat pada Gambar 3.9. Dengan Gambar 3.9 (A) adalah citra LR hasil *flipping* dan Gambar 3.9 (B) adalah hasil *image noise*.



Gambar 3.9. Ilustrasi citra terkenai proses *image noise*

Proses *resizing* menggunakan library python PIL Image dilakukan untuk mensimulasikan data input untuk proses evaluasi. Data citra plat nomor dilakukan *resize* menjadi 256x128 pixel sebagai data citra *high-resolution*, 64x32 pixel untuk mensimulasikan citra *low-resolution*. Citra hasil *resize* dapat dilihat pada Gambar 3.10, dengan citra (A) adalah citra *low-resolution* 64x32 pixel, (B) citra *high-resolution* 256x128 pixel, dan (C) citra asli berukuran 2828x1246 pixel.



Gambar 3.10. Contoh hasil *resize* citra

Normalisasi data matriks array dari citra dilakukan untuk setiap channel warna *red green* dan *blue*, sesuai pada Persamaan 2.6 untuk setiap nilai pixel pada Gambar 3.11 (A) akan dihitung sebagai berikut,

$min_x = 0$

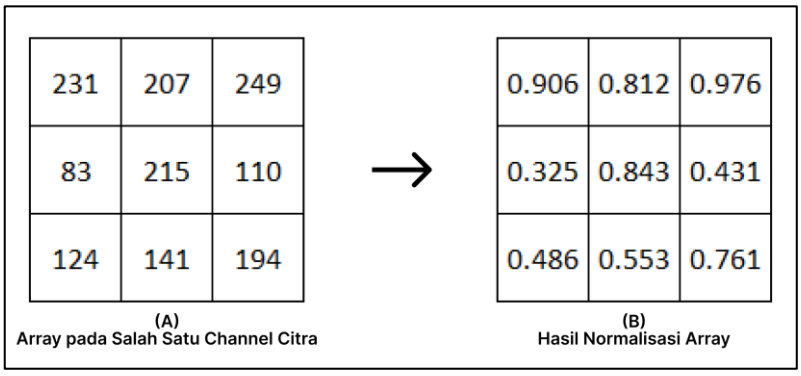
$max_x = 255$

$$x'_{(0,0)} = \frac{x_{(i,j)} - min_x}{max_x - min_x} = \frac{231 - 0}{255 - 0} = 0.906$$

$$x'_{(0,1)} = \frac{207}{255} = 0.812$$

$$x'_{(1,0)} = \frac{83}{255} = 0.325$$

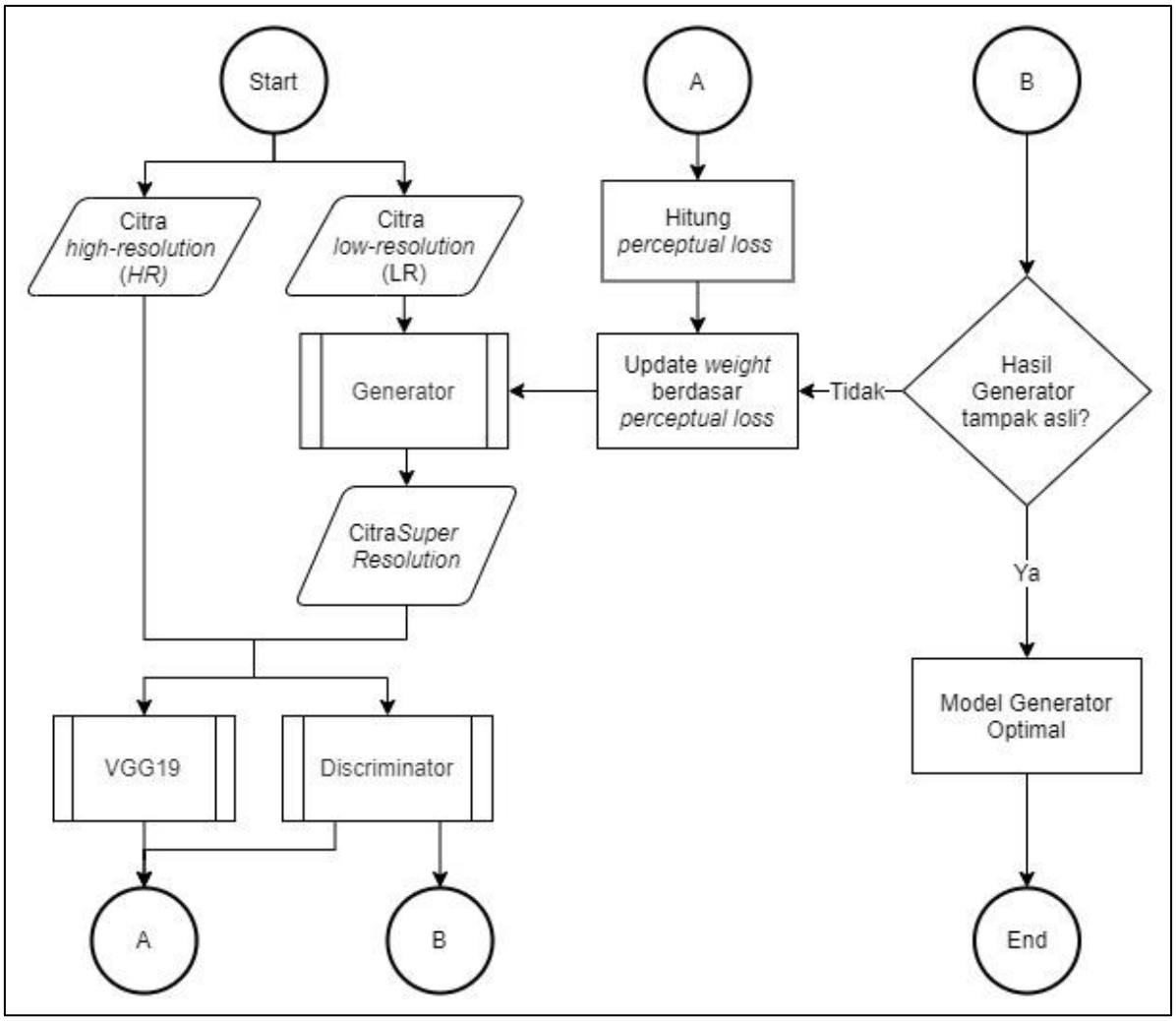
Sehingga didapatkan keseluruhan nilai matriks array citra hasil normalisasi sesuai dengan Gambar 3.11 (B).



Gambar 3.11. Ilustrasi hasil perhitungan matriks array

3.1.4. Proses SRGAN

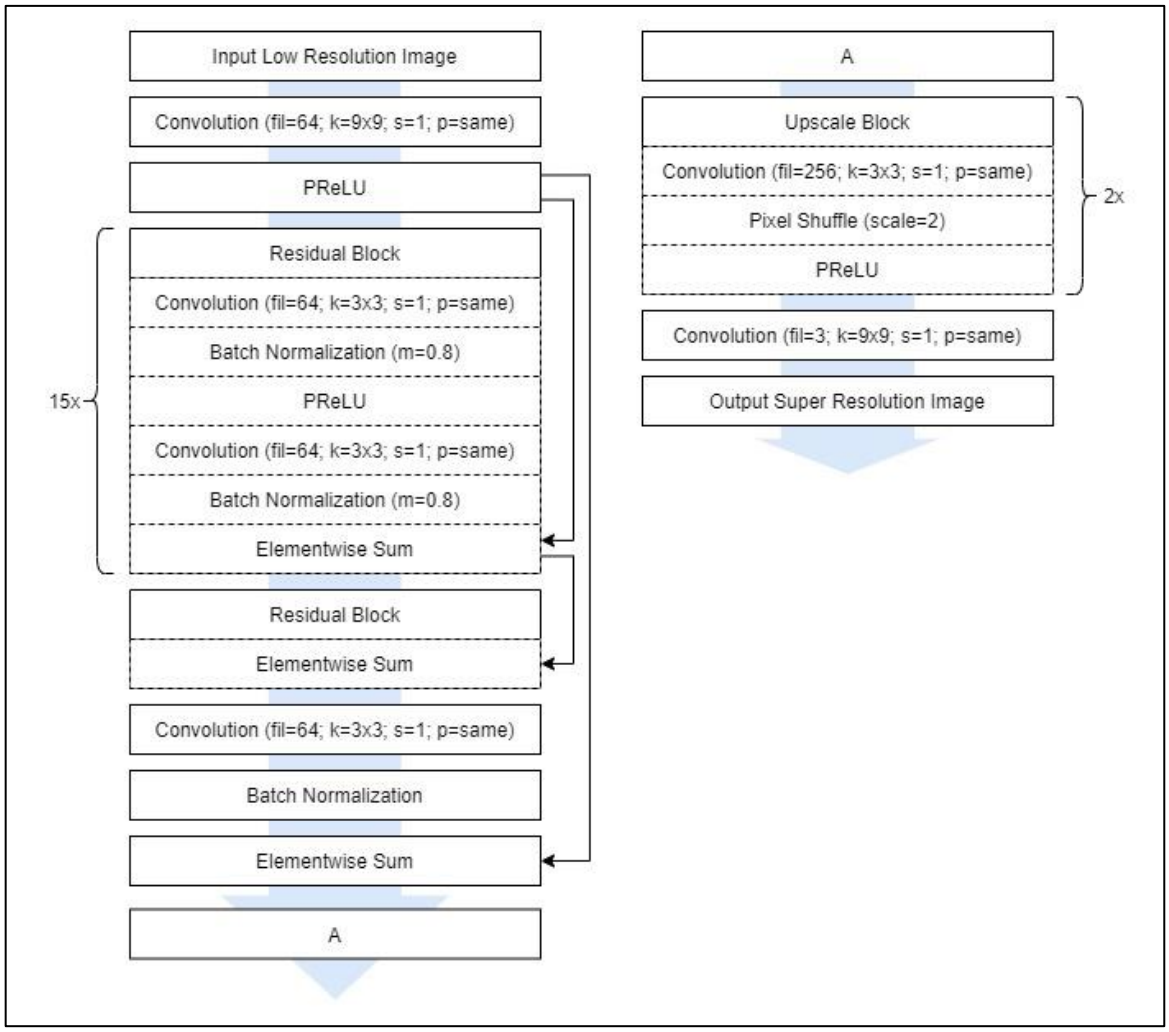
Proses training untuk membangun model SRGAN memiliki alur sebagaimana Gambar 3.12. Data yang diperoleh dari DIV2K kemudian digunakan sebagai data latih *generator model*.



Gambar 3.12 Ilustrasi alur proses *trainning* SRGAN

Struktur dari *generator* dapat dilihat pada Gambar 3.13 dan sesuai pada Gambar 2.8, layer *convolution* pertama memiliki kernel 9x9, 64 filter *feature map*, *stride* 1, dan *padding same*, dengan menggunakan fungsi aktivasi *Parametric ReLU* (PreLU) sesuai pada (Xu et al., 2015). Memiliki 16 *residual block* berstruktur identik, berisikan layer *convolution* pertama dengan kernel 3x3 dan 64 filter *feature map*, diikuti dengan *batch normalization* menurut (Ioffe and Szegedy, 2015) dengan momentum 0.8, dan diikuti dengan aktivasi PReLU, dengan layer *convolution* ke dua pada *residual block* menggunakan *hyperparameter* yang sama seperti *convolution* pertama dan *batch normalization* terakhir, layer terakhir pada blok adalah metode *elementwise sum*. *Elementwise sum* menggunakan hasil output dari hasil aktivasi PreLU awal dan output dari *residual block* sebelumnya untuk menghasilkan output akhir yang akan dibawa pada *residual block* selanjutnya.

Hasil dari *residual block* akan masuk pada layer *convolution* dan *batch normalization* terakhir yang kemudian akan dilakukan *elementwise sum* atau proses Pada dua blok terakhir adalah layer *upscale block* dengan berisikan layer *convolution* dengan kernel 3x3 dan 256 filter *feature* map, dilanjutkan pada layer *upsampling* yang berguna untuk meningkatkan resolusi citra empat kali atau dua kali proses pada dua *upscale block* dengan masing-masing memiliki layer *upsampling* dengan faktor 2x.



Gambar 3.13. Struktur model *generator*

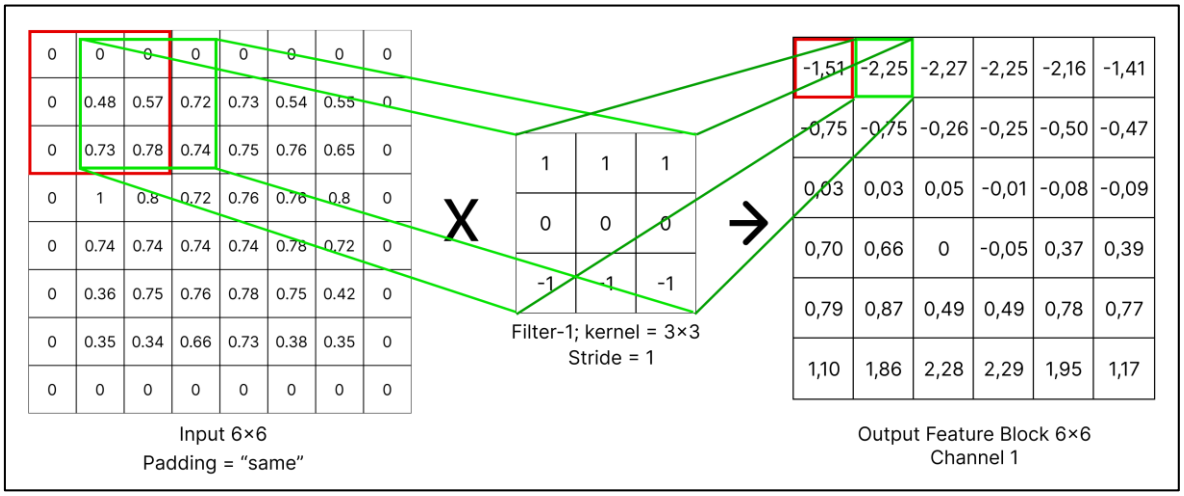
Convolution layer digunakan sebagai ekstraksi fitur dari citra, illustrasi proses *convolution* dapat dilihat pada Gambar 3.14. Dengan *padding* = "*same"* akan ada penambahan nilai 0 pada input untuk setiap awalan dan akhiran pada kolom dan barisnya dengan ukuran akan tetap genap, 32x64 menjadi 34x66. Dengan $S_{x,y}$ sebagai hasil per-pixel dari *feature block* dan x, y sebagai indeks pixel, $K_{i,j}$ sebagai matriks kernel dan i, j sebagai indeks matriks kernel, dengan $I_{x,y}$ sebagai matriks input pixel, maka perhitungan untuk *feature block* dengan *channel* warna *red* pada $S_{2,1}$ atau kotak warna hijau adalah sebagai berikut,

$$S_{2,1} = (I_{2,1} \times K_{1,1}) + (I_{3,1} \times K_{2,1}) + (I_{4,1} \times K_{3,1}) + (I_{2,2} \times K_{1,2}) + (I_{3,2} \times K_{2,2}) + (I_{4,2} \times K_{3,2}) + (I_{2,3} \times K_{1,3}) + (I_{3,3} \times K_{2,3}) + (I_{4,3} \times K_{3,3})$$

$$S_{2,1} = (0 \times 1) + (0 \times 1) + (0 \times 1) + (0{,}48 \times 0) + (0{,}57 \times 0) + (0{,}72 \times 0) + (0{,}73 \times -1) + (0{,}78 \times -1) + (0{,}74 \times -1)$$

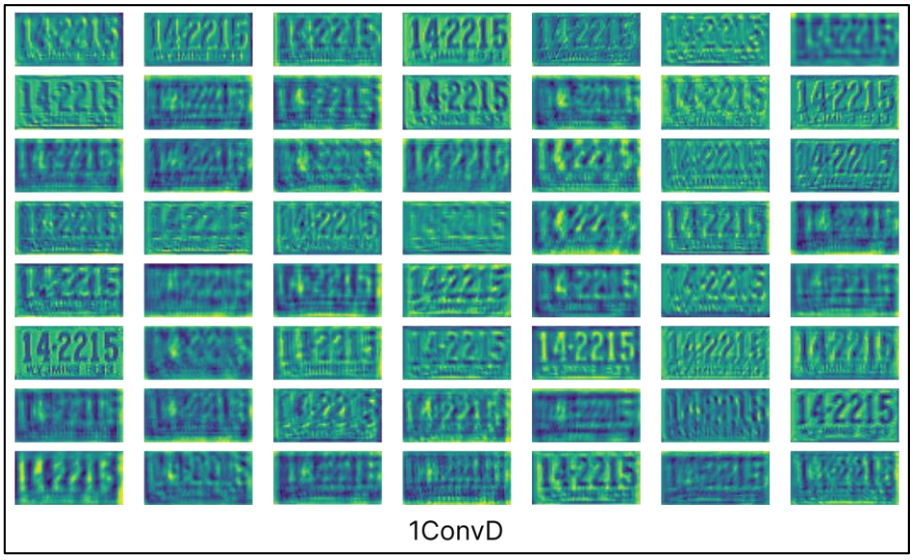
$$S_{2,1} = -2{,}25$$

Dihitung pula pixel-pixel selanjutnya dengan aturan dari kolom pojok kiri nilai input menujut kolom pojok kanan berjalan satu kotak seterusnya, contoh pada Gambar 3.14 dengan area warna merah dilanjutkan ke iterasi berikutnya pada area warna hijau, kemudian dilanjutkan untuk baris berikutnya untuk matriks $I_{1,2}$-$I_{1,4}$ sampai dengan menghasilkan output sebagaimana Gambar 3.14 output *feature block* 6x6. Perhitungan serupa akan terus dilanjutkan untuk *channel* warna selanjutnya, yaitu *green* dan *blue* yang selanjutnya akan dijumlah kan untuk membentuk *feature block* utuh.



Gambar 3.14. Ilustrasi hasil ektraksi fitur pada *convolution* layer

Dikarenakan menggunakan *padding* = "*same"* dan *stride* = 1 maka ukuran output fitur akan tetap sesuai dengan input yang diberikan. Contoh hasil proses ektraksi fitur pada layer *convolutional* pertama sebelum *residual block* dapat dilihat pada Gambar 3.15.



Gambar 3.15. Ilustrasi ekstraksi fitur pada layer *convolutional* pertama

Hasil dari fitur yang terekstraksi kemudian akan masuk pada layer aktivasi *parametric rectified linear units* (PReLU), fungsi aktivasi PReLU digunakan untuk mengubah nilai pixel dibawah nol menjadi y_i dengan nilai a_i atau *alpha* didapatkan secara berkala melalui

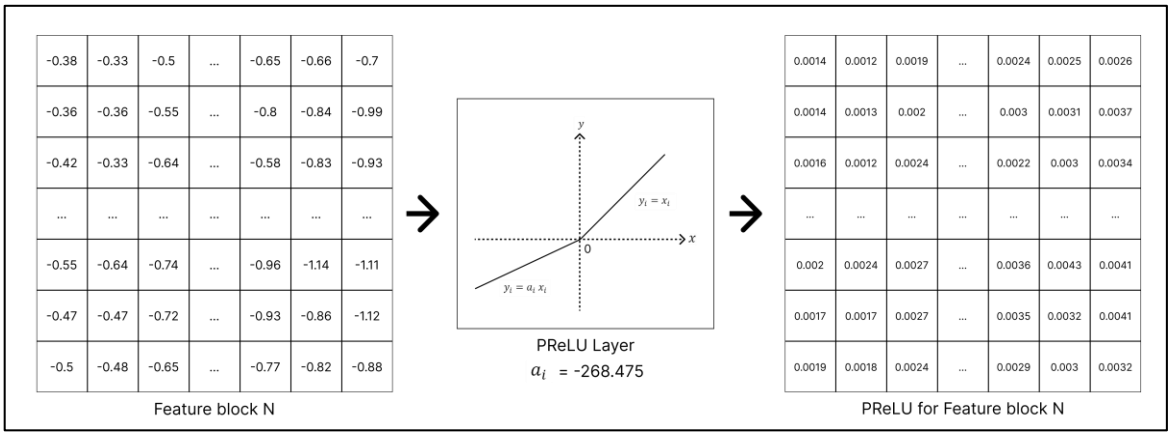
training back propagation, atau nilai y_i didapat sesuai dengan Persamaan 2.2 dengan *alpha* awal bernilai -268.475 maka untuk setiap pixelnya dapat diketahui nilainya sebagai berikut,

$$a_1 = -268.475$$

$$y_{1,1} = \frac{x_{1,1}}{a_1} = \frac{-0.38}{-268.475} = 0.0014$$

$$y_{1,3} = \frac{x_{1,3}}{a_1} = \frac{-0.42}{-268.475} = 0.0016$$

Dengan nilai-nilai selanjutnya didapat berdasarkan Persamaan 2.2, maka keseluruhan hasil layer aktivasi dapat dilihat pada Gambar 3.16, dengan input sebelah kiri dan hasil di sebelah kanan.



Gambar 3.16. Ilustrasi hasil PReLU

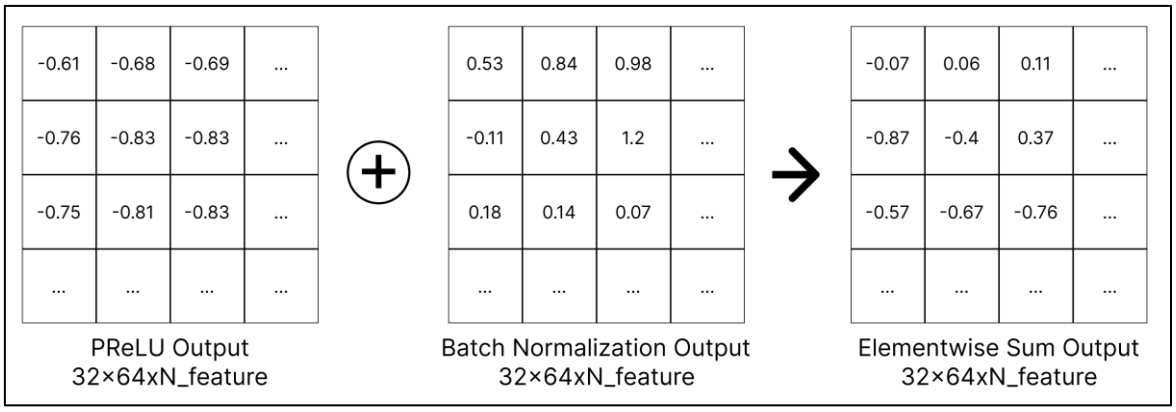
Proses *elementwise sum* dilakukan di layer terakhir untuk setiap *residual block* dan juga sebelum memasuki *upscale block*, *elementwise sum* adalah proses penggabungan atau penjumlahan dua output layer yang berbeda. Sebagai contoh pada perhitungan berikut, dengan $P_{i,j}$ sebagai hasil PReLU dan $BN_{i,j}$ sebagai input pixel hasil *batch normalization.*

$$Add_{i,j} = P_{i,j} + BN_{i,j}$$

$$Add_{1,1} = P_{1,1} + BN_{1,1} = -0.61 + 0.53 = -0.07$$

$$Add_{1,2} = P_{1,2} + BN_{1,2} = -0.76 + -0.11 = -0.87$$

Dengan ilustrasi hasil nilai pixel selanjutnya dapat dilihat pada Gambar 3.17, dengan matriks kanan sebagai hasil penjumlahan, matriks kiri sebagai input pertama dari output PReLU $P_{i,j}$ dan matriks tengah sebagai input ke dua dari output *batch normalization* $BN_{i,j}$

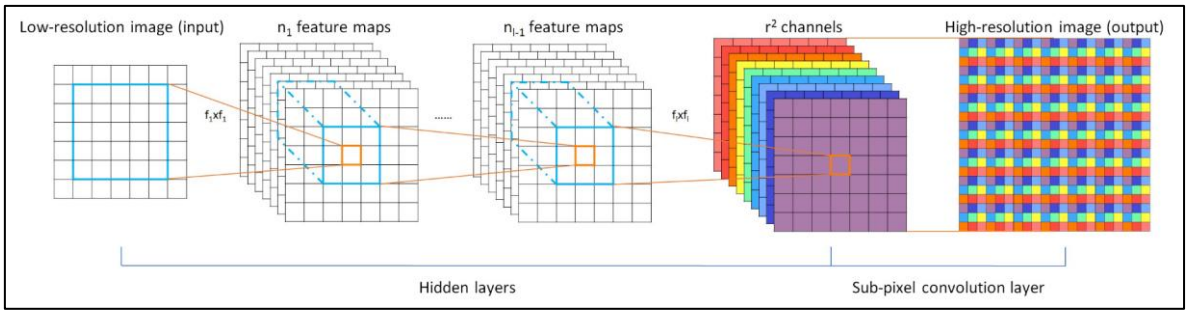


Gambar 3.17. Ilustrasi proses *elementwise sum*

Pada *upsampling* layer, input vector dilakukan peningkatan ukuran dua kali lipat dua kali dari ukuran semula dengan metode *pixel-shuffle* menurut (Shi et al., 2016) pada Gambar 3.18, berdasarkan hasil ektraksi fitur pada residual block, *pixel-shuffle* menyusun ulang elemen dari tensor (B × H × W × C) untuk membentuk tensor baru dengan rasio *scaling* r adalah (B × rH × rW × C/r²), dengan begitu terbentuk citra baru dengan kombinasi untuk setiap fitur yang didapat berdasarkan layer konvolusi dengan filter 256. Dengan B sebagai banyaknya *batch* input yaitu 1, H dan W secara berurutan sebagai *height* dan *width* input yaitu 32×64, C sebagai jumlah fitur dari layer convolution yaitu 256, maka didapati *tensor shape* hasil *pixel-shuffle* dengan $r = 2$ sebagai berikut,

(B × H × W × C) = (1, 32, 64, 256)
$H_{out} = H_{in} \times r = 32 \times 2 = 64$
$W_{out} = W_{in} \times r = 64 \times 2 = 128$
$C_{out} = C_{in} \div r^2 = 256 \div 4 = 64$
(B × rH × rW × C/r²) = (1, 64, 128, 64)

Proses dari *pixel-shuffle* akan diulang sekali lagi atau dengan total dua kali proses *pixel-shuffle* sehingga akan didapati output ukuran citra memiliki dimensi empat kali lebih besar daripada citra input, yaitu dengan hasil tensor (1, 128, 256, 3) setelah melewati layer konvolusi terakhir dengan 3 filter berukuran 9x9 dengan *padding* = ”*same*”.



Gambar 3.18. Ilustrasi pixel-shuffle
Sumber (Shi et al., 2016)

Pada *residual block*, setiap sebelum memasuki layer aktivasi, output layer convolution akan memasuki layer *batch normalization* terlebih dahulu, dengan nilai x_i selama seluruh *mini-batch* atau m sebagai input, berdasarkan output dari layer sebelumnya, dan $\mathrm{BN}_{\gamma,\beta}(x_i)$ sebagai hasil normalisasi, memiliki dua parameter, *gamma* (γ) dan *beta* (β) yang akan terus berubah dalam setiap iterasi *trainning*-nya. Batch normalization bertujuan untuk mempercepat proses konvergensi nilai covarians atau bisa disebut juga mempercepat proses *trainning* menurut (Ioffe and Szegedy, 2015) dan juga pada (Santurkar et al., 2018).

Perhitungan untuk salah satu *feature layer* dimulai dengan mencari *mean* (μ_B) dan *standart deviation* (σ_B^2) sesuai dengan persamaan pada Gambar 3.20, dengan input x_i sesuai pada Gambar 3.19, m adalah total jumlah input maka berdasar Gambar 3.19 akan bernilai 9, maka didapati nilai *mean* (μ_B) sebagai berikut,

-0.38	-0.33	-0.5
-0.36	-0.36	-0.55
-0.42	-0.33	-0.64

Gambar 3.19. Contoh matriks input pada *batch normalization*

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\mu_B = \frac{-0.38 + -0.33 + -0.5 + -0.36 + -0.36 + -0.55 + -0.42 + -0.33 + -0.64}{9}$$

$$\mu_B = \frac{-3.87}{9} = -0.43$$

Dengan menggunakan hasil perhitungan *mean* (μ_B), maka didapati nilai *standart deviation* (σ_B^2) sebagai berikut,

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$$

$$\sigma_B^2 = \frac{(-3.87 - (-0.43))^2}{9} = \frac{(-4.3)^2}{9} = \frac{18.49}{9} = 2.054444$$

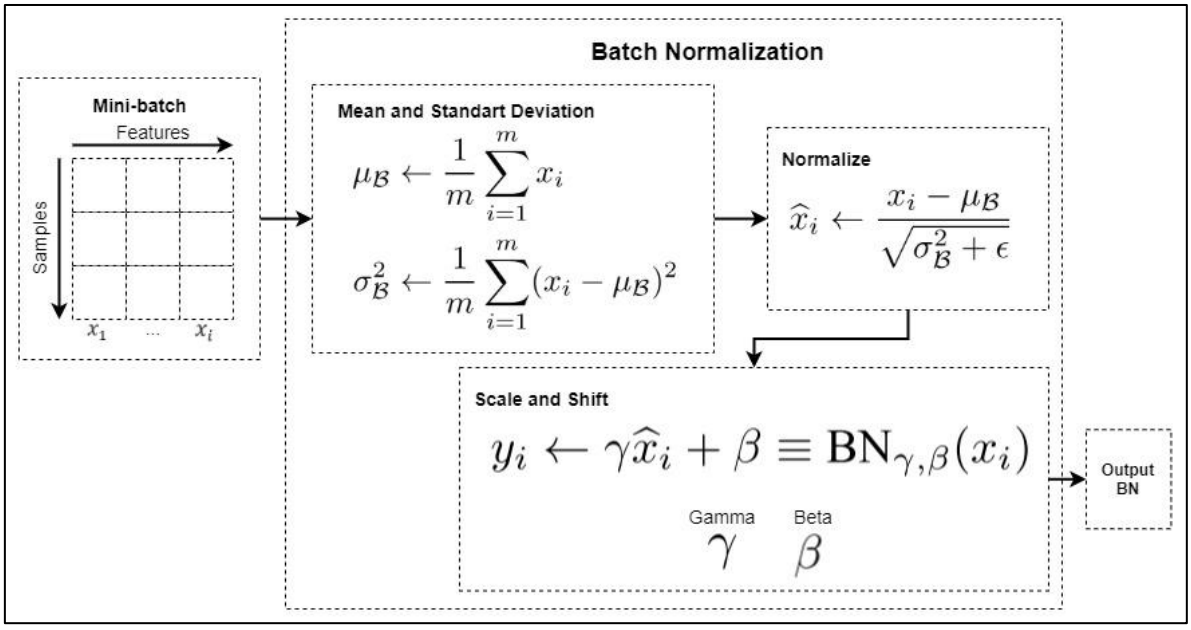
Berdasarkan perhitungan *mean* (μ_B) dan *standart deviation* (σ_B^2) tersebut maka dapat dihitung nilai normalisasi atau $\widehat{x_i}$, dengan nilai *epsilon* atau ϵ adalah nilai yang sangat kecil digunakan untuk menghindari pembagian dengan nol, atau pada perhitungan ini digunakan nilai $\epsilon = 0{,}000001$,

$$\widehat{x_i} = \frac{-3.87 - (-0.43)}{\sqrt{2.054444 + 0{,}000001}} = \frac{-4.3}{1.433334} = -2.9999986 \approx -3$$

Kemudian dapat dihitung nilai $\mathrm{BN}_{\gamma,\beta}(x_i)$ berdasar hasil nilai sebelumnya dengan nilai *gamma* (γ) = 0.8554 dan *beta* (β) = 0.2976 pada saat layer *training* tersebut, adalah sebagai berikut,

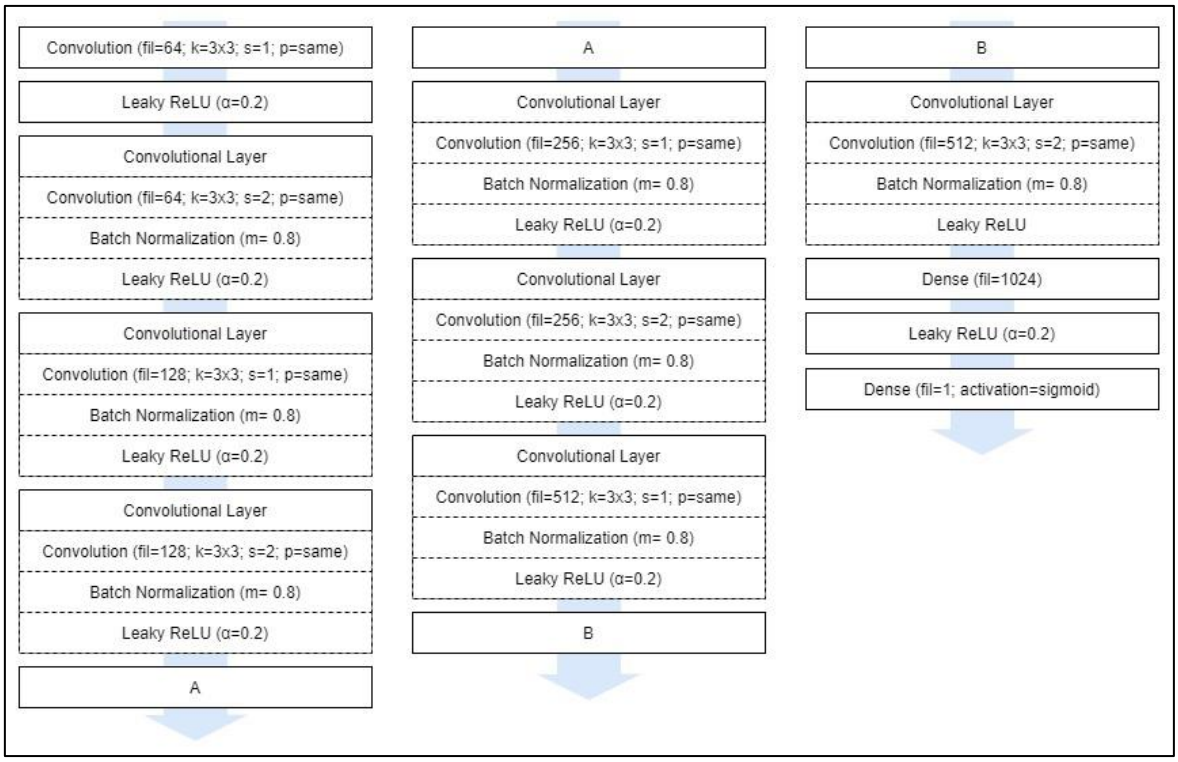
$$\mathrm{BN}_{\gamma,\beta}(x_i) = \gamma\, \widehat{x_i} + \beta = (0.8554)(-3) + 0.2976 = -2.2686$$

Maka hasil dari proses *batch normalization* untuk salah satu *feature layer* adalah $\mathrm{BN}_{\gamma,\beta}(x_i) = -2.2686$, kemudian dihitung untuk setiap *feature layer* sesuai dengan jumlah input pada layer *batch normalization* tersebut.



Gambar 3.20. Ilustrasi alur beserta rumus *batch-normalization*
Sumber (Ioffe and Szegedy, 2015)

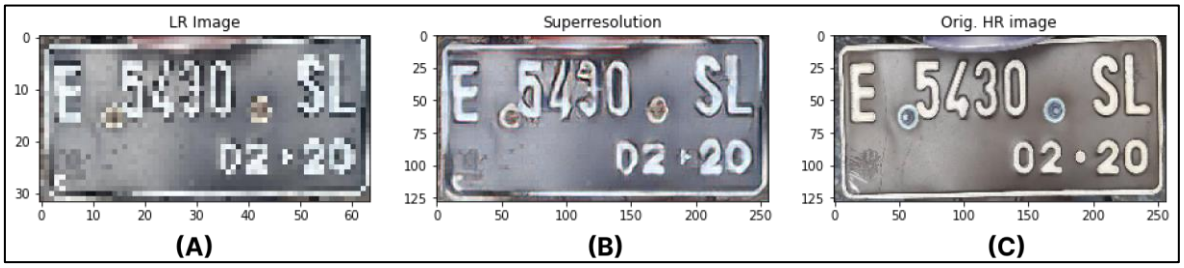
Dengan hasil generasi yaitu citra *super-resolution* sebagai input pada *discriminator model*, untuk kemudian dilakukan deteksi biner apakah citra termasuk atau dapat dianggap sangat mirip dengan citra *high-resolution* yaitu bernilai mendekati atau sama dengan 1, dan atau terdeteksi sebagai citra generasi atau *super-resolution* dengan nilai akhir mendekati atau sama dengan 0. Struktur dari discriminator dapat dilihat pada Gambar 3.21.



Gambar 3.21. Strukur model *discriminator*

Pada *dicriminator* aktivasi menggunakan Leaky ReLU, sedikit berbeda dengan PReLU, dimana nilai alpha pada Leaky ReLU di inisiasikan secara manual, pada *discriminator* menggunakan nilai *alpha* = 0.2, dan menggunakan *dense* layer dengan aktivasi *sigmoid* untuk mengklasifikasikan citra yang masuk sebagai citra *high-resolution* yaitu akan menghasilkan nilai mendekati atau sama dengan satu, dan citra *super-resolution* atau hasil generasi dengan menghasilakn nilai mendekati atau samadengan nol.

Dengan hasil akhir dari SRGAN dapat dilihat pada Gambar 3.22 (A) sebagai citra input atau citra *low-resolution*, Gambar 3.22 (B) adalah hasil citra *super-resolution*, dan Gambar 3.22 (C) adalah data citra asli.



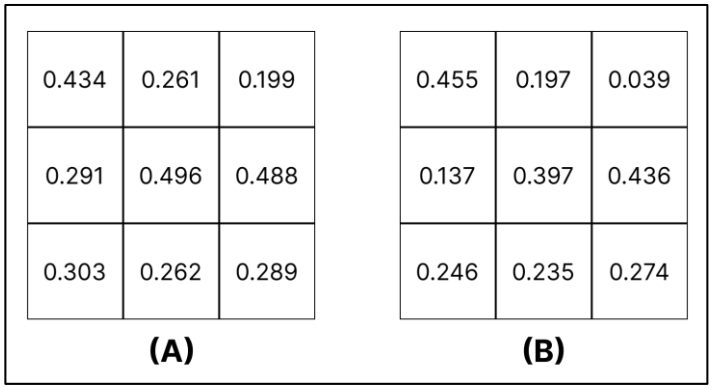
Gambar 3.22. Ilustrasi hasil Super-resolution dengan SRGAN

Hasil citra *super-resolution* kemudian digunakan sebagai bahan evaluasi fitur pada VGG19 untuk mengetahui *content loss* citra *super-resolution* ditimbang dengan citra *high-resolution*. Detail input output setiap layer VGG19 dapat dilihat pada Tabel 3.4 berdasarkan Gambar 2.10.

Tabel 3.4. Layer Output Shape $\mathbf{VGG19_{5.2}}$

Indeks	Nama Layer	*Output Shape*
0	input_1 (InputLayer)	(None, 128, 256, 3)
1	block1_conv1 (Conv2D)	(None, 128, 256, 64)
2	block1_conv2 (Conv2D)	(None, 128, 256, 64)
3	block1_pool (MaxPooling2D)	(None, 64, 128, 64)
4	block2_conv1 (Conv2D)	(None, 64, 128, 128)
5	block2_conv2 (Conv2D)	(None, 64, 128, 128)
6	block2_pool (MaxPooling2D)	(None, 32, 64, 128)
7	block3_conv1 (Conv2D)	(None, 32, 64, 256)
8	block3_conv2 (Conv2D)	(None, 32, 64, 256)
9	block3_conv3 (Conv2D)	(None, 32, 64, 256)
10	block3_conv4 (Conv2D)	(None, 32, 64, 256)
11	block3_pool (MaxPooling2D)	(None, 16, 32, 256)
12	block4_conv1 (Conv2D)	(None, 16, 32, 512)
13	block4_conv2 (Conv2D)	(None, 16, 32, 512)
14	block4_conv3 (Conv2D)	(None, 16, 32, 512)
15	block4_conv4 (Conv2D)	(None, 16, 32, 512)
16	block4_pool (MaxPooling2D)	(None, 8, 16, 512)
17	block5_conv1 (Conv2D)	(None, 8, 16, 512)
18	block5_conv2 (Conv2D)	(None, 8, 16, 512)
19	block5_conv3 (Conv2D)	(None, 8, 16, 512)
20	block5_conv4 (Conv2D)	(None, 8, 16, 512)

Berdasarkan Persamaan 2.4, dengan dimensi dari *feature map* adalah $W, H = 8, 16$ pada layer konvolusi ke empat pada blok konvolusi ke lima atau pada Tabel 3.4 layer dengan indeks 20, $\varphi_{i,j}(I^{HR})_{x,y}$ sebagai fitur yang terekstraksi dari citra *high-resolution* Gambar 3.23 (A), dan $\varphi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y}$ sebagai matriks fitur yang terektraksi dari citra generasi *super-resolution* Gambar 3.23 (B),



Gambar 3.23. Ilustrasi matriks ekstraksi fitur *super-resolution* dan *high-resolution*

Berdasarkan data matriks pada Gambar 3.23 maka akan didapati nilai *content loss* sebagai berikut,

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\varphi_{i,j}(I^{HR})_{x,y} - \varphi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y}\right)^2$$

$$l_{VGG/i.j}^{SR} = \frac{1}{(8)(16)}((0.434 - 0.455)^2 + (0.261 - 0.197)^2 + (0.199 - 0.039)^2 + (0.291 - 0.137)^2 + (0.496 - 0.397)^2 + (0.488 - 0.436)^2 + (0.303 - 0.246)^2 + (0.262 - 0.235)^2 + (0.289 - 0.274)^2)$$

$$l_{VGG/i.j}^{SR} = \frac{1}{128}((-0.021)^2 + (0.064)^2 + (0.16)^2 + (0.154)^2 + (0.099)^2 + (0.052)^2 + (0.057)^2 + (0.027)^2 + (0.015)^2)$$

$$l_{VGG/i.j}^{SR} = \frac{1}{128}(0.000441 + 0.004096 + 0.0256 + 0.023716 + 0.009801 + 0.002704 + 0.003249 + 0.000729 + 0.000225)$$

$$l_{VGG/i.j}^{SR} = \frac{0.047521}{128} = 0.00037125781$$

Hasil dari *discriminator* akan dihitung nilai dari *Adversarial loss* yaitu jumlah total nilai kemiripan citra *super-resolution* $D_{\theta_D}\left(G_{\theta_G}(I^{LR})\right)$ dengan citra *high-resolution* pada keseluruhan data *trainning*, meiliki rentang 0 sampai 1, dengan citra memiliki klasifikasi $D_{\theta_D}\left(G_{\theta_G}(I^{LR})\right) = 1.0$ sebagai citra *high-resolution*. Berdasarkan Persamaan 2.5 untuk jumlah *training* data sebanyak enam buah, akan didapati *Adversarial loss* sebagai berikut,

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -log\, D_{\theta_D}\left(G_{\theta_G}(I^{LR})\right)$$

$$l_{Gen}^{SR} = -\log(0.773) + (-\log(0.952)) + (-\log(1.0)) + (-\log(0.533)) + (-\log(0.731)) + (-\log(0.882))$$

$$l_{Gen}^{SR} = 0.11182 + 0.02136 + 0 + 0.27327 + 0.13608 + 0.05453$$

$$l_{Gen}^{SR} = 0.59706$$

VGG19 *content loss* $l_{VGG/i.j}^{SR}$ dan *adversarial loss* l_{Gen}^{SR} kemudian di jumlahkan sehingga mendapatkan *perceptual loss* yang akan digunakan untuk mengevaluasi dan memperbarui *gradient* dari *generator*, berdasarkan Persamaan 2.3 dan hasil dari perhitungan sebelumnya yaitu $l_{VGG/i.j}^{SR} = 0.00037125781$ dan $l_{Gen}^{SR} = 0.59706$, maka didapati *perceptual loss* sebagai berikut,

$$l^{SR} = l_{VGG/i.j}^{SR} + 10^{-3}(l_{Gen}^{SR})$$

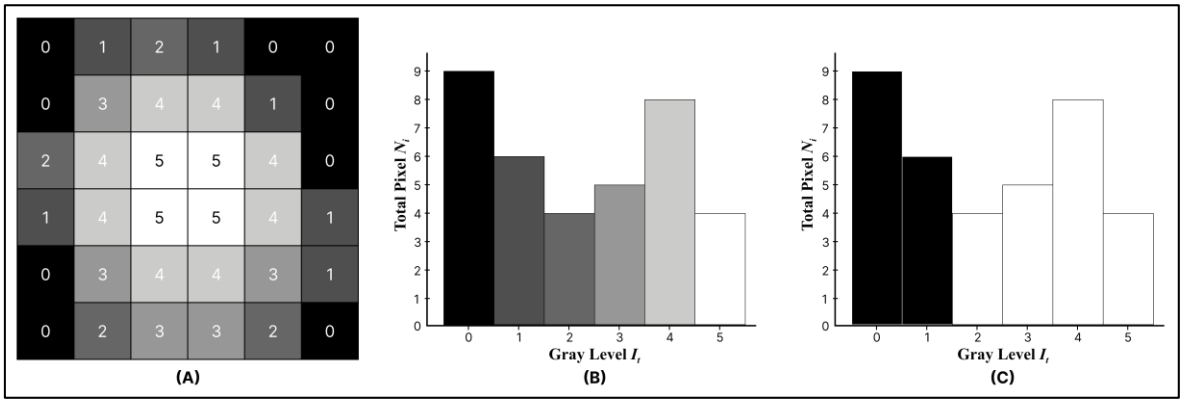
$$l^{SR} = 0.00037126 + (0.001 \times 0.59706) = 0.00037126 + 0.00059706$$

$$l^{SR} = 0.00096836$$

3.1.5. *Pre-Processing* II

Pada *pre-processing* ke dua, hasil generasi citra *super-resolution* dan hasil dari *cropping bounding-box* dilakukan proses *grayscaling*, yaitu mengubah warna citra dari RGB menjadi monokromatik dari hitam hingga putih 8-bit dengan tingkat nilai keabuan 256, dari 0 sampai 255. Dalam penelitian ini digunakan library cv2.cvtColor dengan parameter COLOR_BGR2GRAY.

Hasil citra *grayscale* kemudian dilakukan proses *otsu's thresholding,* yaitu untuk mengubah rentang warna 8-bit menjadi biner. *Thresholding* dilakukan dengan menggunakan library cv2.threshold, contoh matriks citra untuk proses *otsu's thresholding* dapat dilihat pada Gambar 3.24 (A).



Gambar 3.24. Illustrasi data thresholding

Pada Gambar 3.24 (A) adalah nilai matriks pixel menurut tingkat intensitas keabuan. Dengan grafik pada Gambar 3.24 (B) menggambarkan sebaran nilai keabuan. Pada grafik Gambar 3.24 (C) ditentukan nilai *threshold* atau I_t bernilai 2, dengan begitu dapat ditentukan batas bawah dan batas atas seperti pada illustrasi grafik Gambar 3.24 (C). Kemudian dapat ditentukan nilai-nilainya sesuai dengan Persamaan 2.7, dengan *gray level* sebagai I dan total pixel sebagai N, dengan hasil dapat dilihat pada Tabel 3.5 kolom indeks I_t bernilai 2.

$$\omega_0 = \frac{jumlah\ N_i < I_t}{total\ pixel\ citra} = \frac{9+6}{36} = 0.42$$

$$\omega_1 = \frac{jumlah\ N_i \geq I_t}{total\ pixel\ citra} = \frac{4+5+8+4}{36} = 0.58$$

$$\mu_0 = \frac{(I_0 \times N_0) + \cdots + (I_5 \times N_5) < I_t}{jumlah\ N_i < I_t} = \frac{(9 \times 0) + (6 \times 1)}{9+6} = 0.4$$

$$\mu_1 = \frac{(I_0 \times N_0) + \cdots + (I_5 \times N_5) \geq I_t}{jumlah\ N_i \geq I_t}$$

$$= \frac{(4 \times 2) + (5 \times 3) + (8 \times 4) + (4 \times 4)}{4+5+8+4} = 3.57$$

Sehingga didapati nilai varians antar kelas sesuai dengan Persamaan 2.8 dengan I_t bernilai 2 adalah sebagai berikut,

$$\sigma_b^2 = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2 = (0.42)(0.58)(0.4 - 3.57)^2 = 2.44$$

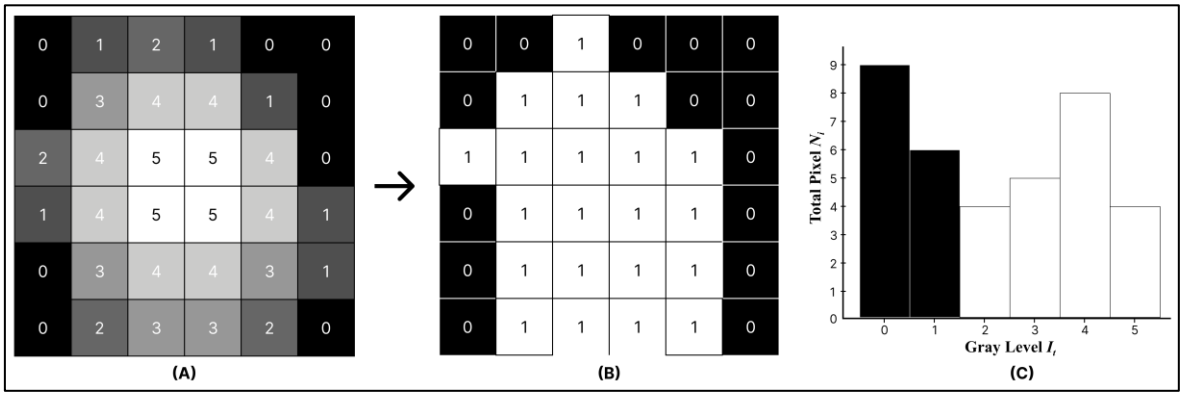
Untuk semua kemungkinan nilai I_t selanjutnya akan dihitung dengan hasil sebagaimana Tabel 3.5. Kemudian nilai threshold atau I_t akan ditentukan berdasarkan hasil

perhitungan menurut Persamaan 2.7 dengan nilai varians antar kelas atau σ_b^2 yang tertinggi, yaitu pada kolom I_t bernilai 3, dengan nilai varians $\boldsymbol{\sigma_b^2}$ antar kelas 2,56.

Tabel 3.5. Perhitungan nilai varians antar kelas untuk *thresholding*

I_t	**0**	**1**	**2**	**3**	**4**	**5**
ω_0	0	0,25	0,42	0,53	0,67	0,89
μ_0	0	0	0,4	0,74	1,21	1,91
ω_1	1	0,75	0,58	0,47	0,33	0,11
μ_1	2,25	3	3,57	3,94	4,33	5
σ_b^2	0	1,69	2,44	**2,56**	2,17	0,95

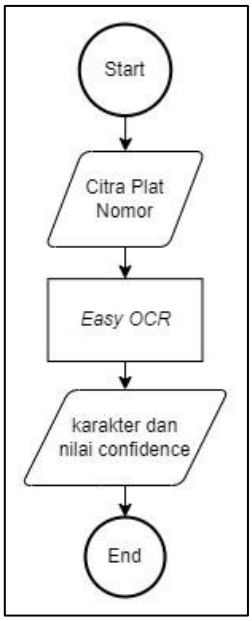
Nilai $I_t = 3$ kemudian digunakan untuk proses thresholding, akan mendapatkan citra baru sesuai dengan Gambar 3.25 (B). Dengan semua nilai *gray level* dibawah tiga akan bernilai biner 0 atau hitam, dan lebih atau sama dengan tiga akan bernilai biner 1 atau putih.



Gambar 3.25. Illustrasi hasil proses thresholding

3.1.6. *Optical Character Recognition*

Setelah melalui tahapan *pre-processing* yang ke dua, citra selanjutnya akan dilakukan proses pengenalan karakter, dengan alur proses sebagaimana Gambar 3.26. Hasil ekstraksi karakter kemudian akan digunakan untuk proses evaluasi dan analisis akurasi pengenalan karakter jika citra dilakukan proses SRGAN terlebih dahulu, dan dengan yang tidak melalui proses peningkatan kualitas citra.



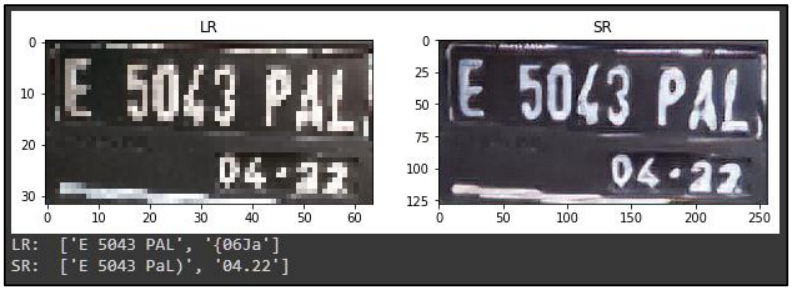
Gambar 3.26 Alur proses pengenalan karakter

Digunakan library *Easy* OCR milik (JaidedAI, 2021), dengan implementasi algoritma sebagaimana Algoritma 6. Menggunakan bahasa huruf "*English*" dengan memberikan input citra hasil *pre-processing II* dan akan mengembalikan output berupa karakter yang terdeteksi.

Algoritma 6 : Algoritma *Easy OCR*

```
INIT imagePath
BEGIN
   SET easyocr word language TO "English"
   IF imagePath IS VALID:
      OPEN file ON imagePath STORE INTO image
      USING easyocr.reader READ TEXT IN image STORE INTO character
      IF character IS NULL
         WRITE "Gagal Mendeteksi Karakter pada Citra!"
      ELIF
         WRITE "Karakter terdeteksi: " + character
      ENDIF
   ENDIF
END
```

Contoh hasil keluaran dari *Easy OCR* dapat dilihat pada Gambar 3.27, dengan input berupa citra file jpg atau jpeg dan memiliki output berupa nilai *string* dalam list berdasarkan hasil ekstraksi karakter pada citra.



Gambar 3.27. Ilustrasi hasil pengenalan karakter

3.1.7. Evaluasi Model

Hasil model dari YOLOv4 akan dievaluasi untuk mengetahui tingkat akurasi deteksi sebenarnya, dengan menggunakan *confussion matrix* berdasarkan citra pada Gambar 3.28

Gambar 3.28. Ilustrasi citra input dengan *bounding-box*

Pada Gambar 3.28 citra dengan *bounding-box* warna ungu hasil deteksi YOLOv4 maka didapati hasil sebagaimana Tabel 3.6.

Tabel 3.6. Ilustrasi Hasil *Confusion Matrix* berdasarkan Gambar 3.28

		Nilai Prediksi	
		***Positive* (Y)**	***Negative* (N)**
Nilai Asli	***Positive* (P)**	4 (**TP**)	0 (**FN**)
	***Negative* (N)**	0 (**FP**)	1 (**TN**)

Dengan nilai *True Positive* (TP) adalah prediksi benar dan terdapat pada nilai asli, yaitu terdeteksi 4 plat nomor kendaraan roda empat. Memiliki nilai *True Negatives* (TN) atau prediksi bukan termasuk plat kendaraan roda empat secara benar, yaitu terlihat pada Gambar 3.28 memiliki 1 plat kendaraan roda dua. Dengan nilai *False Positive* (FP) atau nilai prediksi bukan plat kendaraan roda empat namun terdeteksi dan dikategorikan, dan *False Negative* (FN) yaitu nilai prediksi termasuk plat kendaraan roda empat namun tidak terdeteksi dan tidak dikategorikan bernilai 0. Berdasarkan hasil *confussion matrix* tersebut kemudian didapati nilai *recall* berdasarkan Persamaan 2.15, nilai akurasi berdasarkan Persamaan 2.16, dan nilai F1 *Score* berdasarkan Persamaan 2.17 sebagai berikut,

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{4}{4+0} = 1$$

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} = \frac{4+1}{4+1+0+0} = 1$$

$$F1_{Score} = \frac{2TP}{2TP + FP + FN} = \frac{2(4)}{2(4)+0+0} = 1$$

Hasil citra *super-resolution* dari SRGAN akan diujikan pada proses pengenalan karakter yang kemudian akan dibandingkan hasilnya dengan citra tanpa melalui proses SRGAN terlebih dahulu. Dalam proses evaluasi model akan digunakan 30 citra hasil ektraksi atau *cropping* plat nomor berdasarkan deteksi YOLOv4 dan dilanjutkan untuk 20 citra uji sintetis, yaitu *high-resolution* dengan ukuran 256x128 pixel dan citra *low-resolution* berukuran 64x32 pixel. Dengan mencari nilai *confusion matrix* berdasarkan Gambar 3.27 (LR) didapati hasil pada Tabel 3.7.

Tabel 3.7. Ilustrasi Hasil *Confusion Matrix* berdasarkan Gambar 3.27 (LR)

		Nilai Prediksi : E 5043 PAL {06Ja	
	E 5043 PAL 04.22	***Positive* (Y)**	***Negative* (N)**
Nilai Asli	***Positive* (P)**	11 (**TP**)	1 (**FN**)
	***Negative* (N)**	4 (**FP**)	0 (**TN**)

Dengan nilai TP adalah prediksi benar dan terdapat pada nilai asli yaitu karakter "E 5043 PAL" dan "0" maka memiliki nilai 11 termasuk karakter spasi, nilai FN adalah prediksi salah yang sebenarnya ada pada nilai asli, yaitu bernilai 1, adalah karakter "." Titik yang gagal dikenali. Pada FP adalah nilai prediksi benar namun tidak ada pada nilai asli, memiliki nilai 4, adalah karakter "{6Ja", dan dengan TN adalah nilai prediksi salah dan tidak ada pada nilai asli. Berdasarkan Tabel 3.7 maka dapat diketahui nilai *recall* berdasarkan Persamaan 2.15, nilai akurasi berdasarkan Persamaan 2.16, dan nilai F1 *Score* berdasarkan Persamaan 2.17 sebagai berikut,

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{11}{11 + 1} = 0.9167$$

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} = \frac{11 + 0}{11 + 0 + 4 + 1} = 0.6875$$

$$F1_{Score} = \frac{2TP}{2TP + FP + FN} = \frac{2(11)}{2(11) + 4 + 1} = 0.8148$$

Dengan nilai recall 0.9167, akurasi 0.6875, dan F1-Score 0.8148 maka didapati rencana pengujian sebagaimana pada Tabel 3.8. Dengan klasifikasi semakin mendekati 1 maka ketiga nilai tersebut dianggap semakin baik (Powers, 2020).

Tabel 3.8. Ilustrasi rencana evaluasi OCR pada citra *low-resolution*

Citra ke-	**Karakter Asli**	**Karakter Dikenali**	***Recall* ↑**	**Akurasi ↑**	**F1 *Score* ↑**
Citra 1	E 5043 PAL 04.22	E 5043 PAL {06Ja	0.9167	0.6875	0.8148
…					
Citra 10					

Dengan citra asli sebagai input testing maka dapat diketahui terlebih dahulu karakter yang ada pada citram dan juga untuk dapat membantu mendapatkan citra *low-resolution* (LR) dan citra *high-resolution* (HR). Dengan menggunakan metode perhitungan yang sama maka didapati nilai *confusion matrix* untuk Gambar 3.27 (SR) sebagaimana Tabel 3.9.

Tabel 3.9. Ilustrasi Hasil *Confusion Matrix* berdasarkan Gambar 3.27 (LR)

		Nilai Prediksi : E 5043 PaL 04.22	
	E 5043 PAL 04.22	***Positive* (Y)**	***Negative* (N)**
Nilai Asli	***Positive* (P)**	13 (**TP**)	0 (**FN**)
	***Negative* (N)**	1 (**FP**)	1 (**TN**)

Berdasarkan Tabel 3.9 maka dapat diketahui nilai *recall* berdasarkan Persamaan 2.15, nilai akurasi berdasarkan Persamaan 2.16, dan nilai F1 *Score* berdasarkan Persamaan 2.17 sebagai berikut,

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{13}{13 + 0} = 1$$

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} = \frac{13 + 1}{13 + 1 + 1 + 0} = 0.933$$

$$F1_{Score} = \frac{2TP}{2TP + FP + FN} = \frac{2(13)}{2(13) + 1 + 0} = 0.963$$

Dengan nilai tersebut maka didapati rencana pengujian sebagaimana pada Tabel 3.10. Dengan kedua hasil pada Tabel 3.8 dan Tabel 3.10 dapat digunakan dalam mengukur tingkat akurasi pada setiap pengenalan karakternya, apakah mengalami peningkatan atau bahkan penurunan.

Tabel 3.10. Ilustrasi rencana evaluasi OCR pada citra dikenai *super-resolution*

Citra Ke-	**Karakter Asli**	**Karakter Dikenali**	***Recall* ↑**	**Akurasi ↑**	**F1 *Score* ↑**
Citra 1	E 5043 PAL 04.22	E 5043 PaL 04.22	1	0.933	0.963
…					
Citra 10					

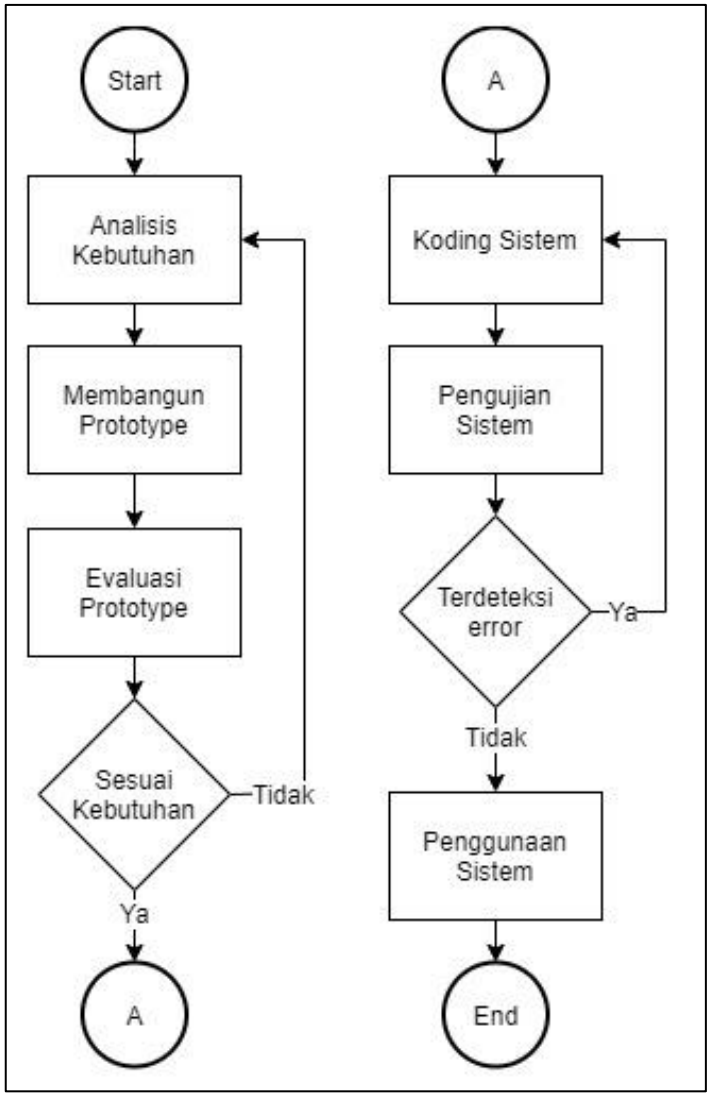
Dengan 20 citra uji *low-resolution* sintetis didapat berdasarkan proses *resizing* dengan hasil ukuran menjadi 64x32 pixel dan citra *high-resolution* sintetis menjadi 256x128 pixel. Juga dilakukan penambahan *noise* pada citra uji *low-resolution* sintetis menggunakan fungsi tensorflow yaitu random_jpeg_quality, hasil generasi citra *super-resolution* akan dihitung nilai *Peak Signal-to-Noise Ratio* (PSNR) dan *Structural Similarity Index* (SSIM) antara citra generasi *super-resolution* dengan citra *high-resolution* sintetis, dengan semakin besar nilai PSNR maupun SSIM maka semakin baik hasil dari citra generasi *super-resolution*. Dilakukan juga evaluasi *confussion matrix* maupun pengukuran terhadap ukuran file citra sebelum dan sesudah diperlakukan proses SRGAN. Untuk selanjutnya didapati rencana pengujian sebagaimana Tabel 3.11.

Tabel 3.11. Ilustrasi rencana evaluasi citra *super-resolution*

No	**Citra**	**Ukuran File**		**PSNR↑**	**SSIM↑**
		Citra *low-resolution*	**Citra *super-resolution***		
1	Citra 1	123 KB	456 KB	12,34 db	0,123
…	…				
10	Citra 10				

3.2. Metodologi Pengembangan Sistem

Metode peengembangan sistem pada penelitian ini akan mengacu pada metode *prototying* menurut (Pressman, 2014), ilustrasi alur dapat dilihat pada Gambar 3.29.



Gambar 3.29. Illustrasi alur metode pengembangan sistem prototyping

3.2.1. Analisis Kebutuhan

Dalam tahap analisis kebutuhan, peneliti akan menganalisa kebutuhan untuk pengembangan perangkat lunak sistem, yaitu berupa kebutuhan funsional maupun non-fungsional. Hasil analisa kebutuhan fungsional sistem adalah,

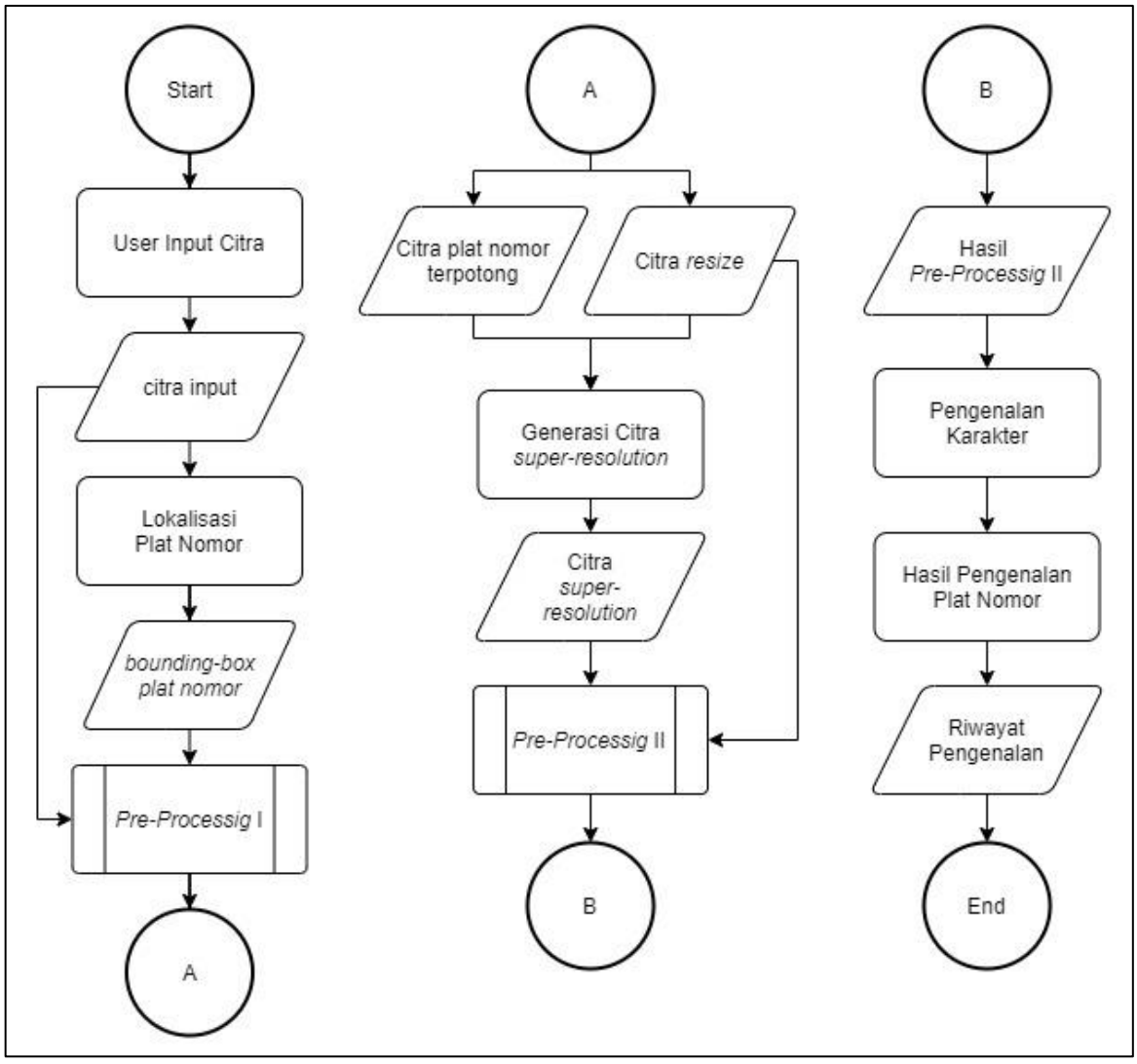
1. Sistem dapat menerima input citra.
2. Sistem dapat melakukan lokalisasi plat nomor kendaraan roda empat.
3. Sistem dapat melakukan ekstraksi atau *cropping* berdasarkan hasil lokalisasi.
4. Sistem dapat menghasilkan citra *super-resolution* berdasarkan hasil *cropping*.
5. Sistem dapat mengenali dan mengekstraksi karakter berdasarkan hasil *cropping* dari lokalisasi maupun generasi *super-resolution*.

Dengan kebutuhan non-fungsional berupa kebutuhan perangkat keras dan perangkat lunak sebagai berikut,

1. Komputer pribadi dengan spesifikasi,
    a. CPU Ryzen 5 1600 @ 3.20 GHz
    b. RAM 16 GB
    c. Sistem Operasi Windows 10 64-bit
2. Visual Studio Code IDE dan Web IDE Google Collab.
3. Bahasa pemrograman Python, dengan library tensorflow, keras, cv2, pillow, matplotlib, darknet, easyOcr, numpy, dan Fast API.
4. Bahasa pemrograman JavaScript dengan framework ReactJS.
5. Database management dengan SQLite.

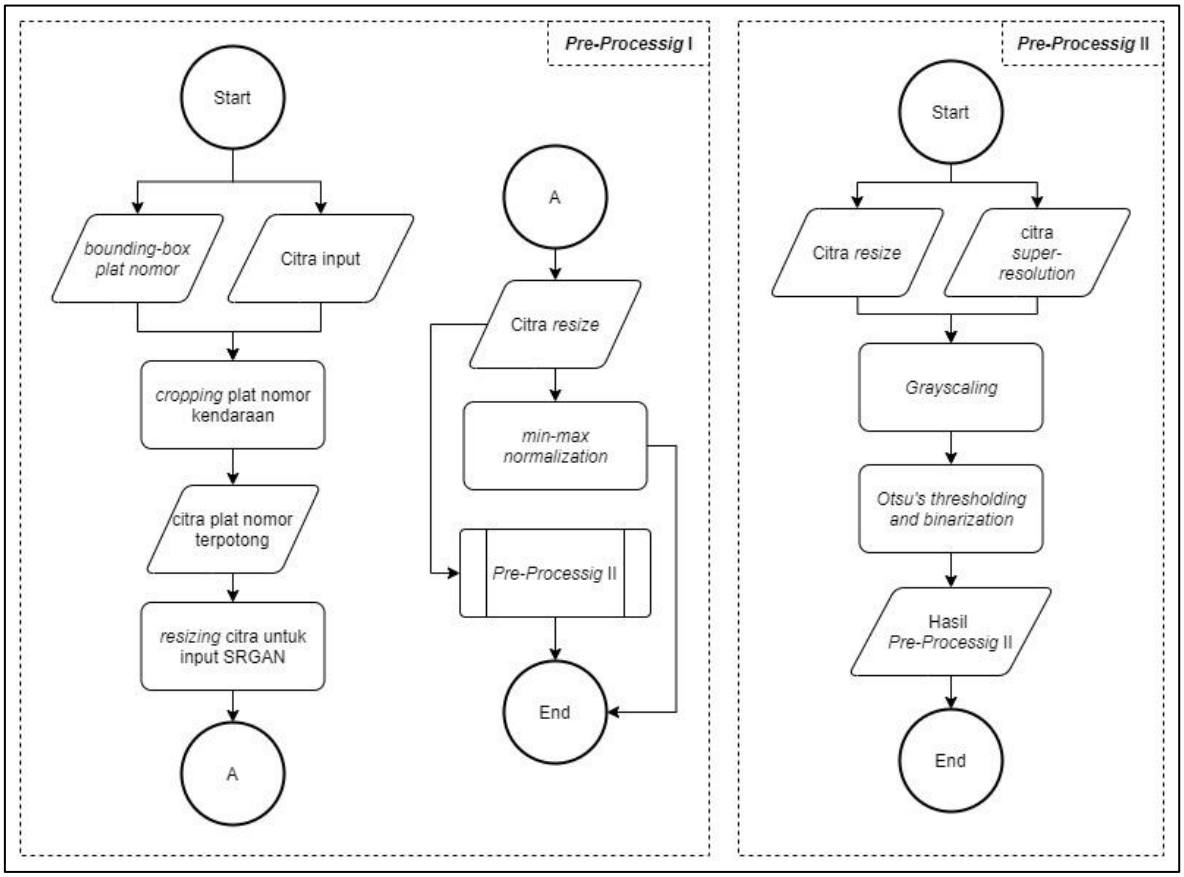
3.2.2. Rancangan *Prototype*

Dalam membangun prototype agar hasil implementasi sesuai dengan kebutuhan fungsional sistem maka dibutuhkan perancangan dalam membangun sistem. Adapun rancangan alur utama pada sistem digambarkan pada Gambar 3.30.



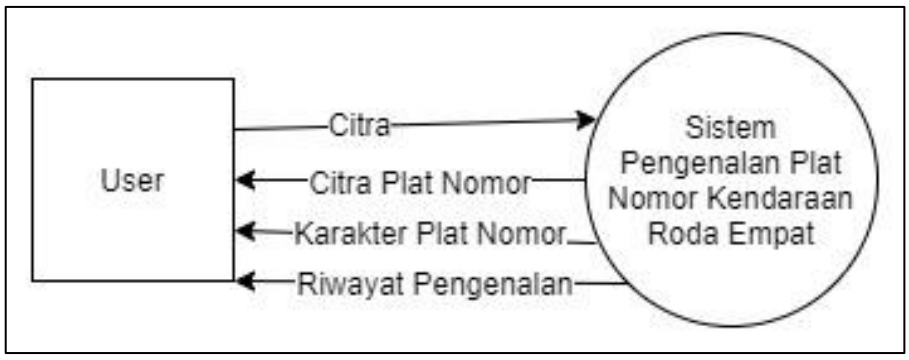
Gambar 3.30. Alur umum sistem

Berdasarkan Gambar 3.30 memiliki dua sub proses, yaitu *pre-processig I* dan *pre-processig II* dengan detail alur dapat dilihat pada Gambar 3.31.



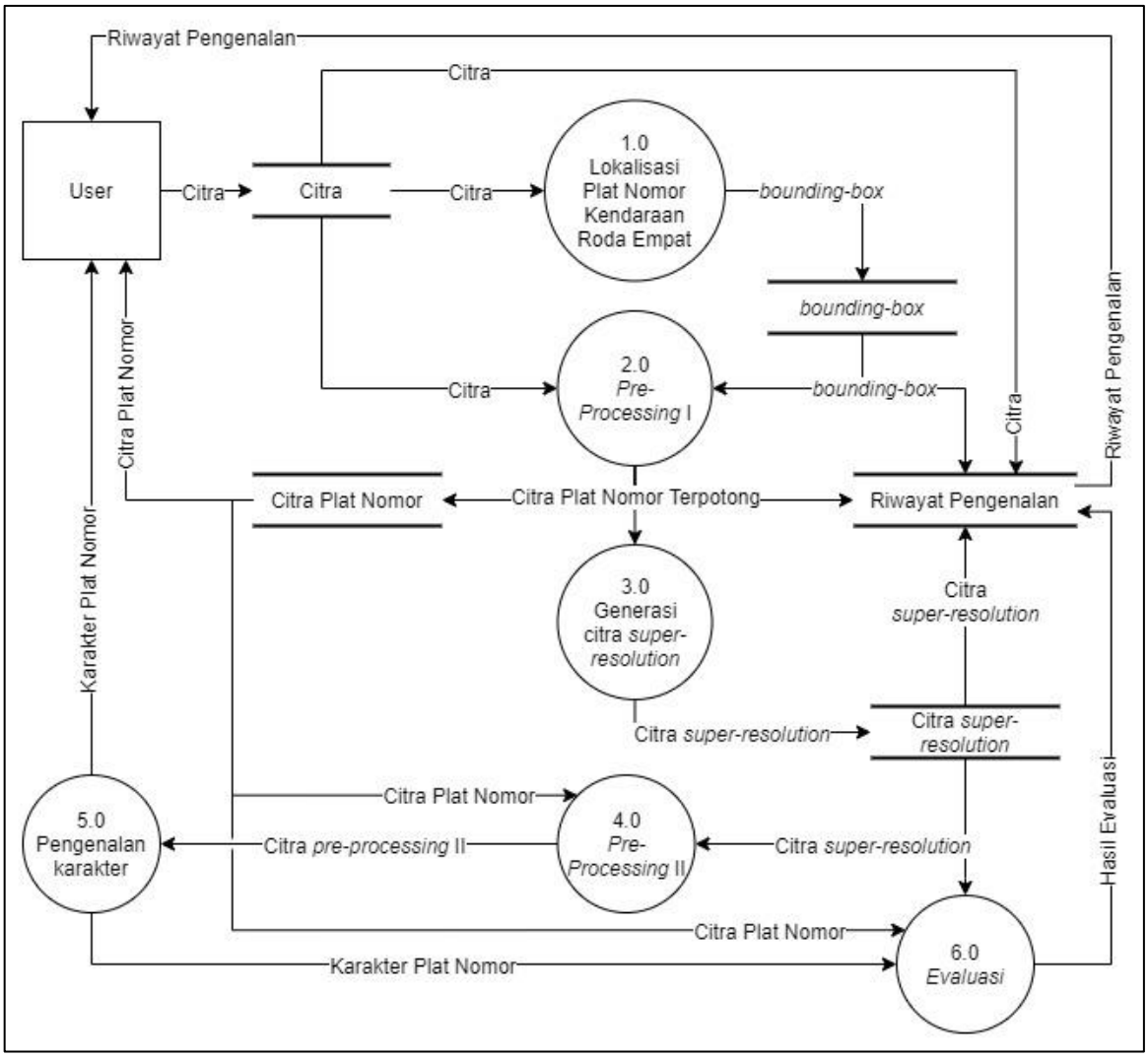
Gambar 3.31. Alur *pre-processig I* dan *pre-processig II*

Dengan rancangan alur data pada sistem atau *Data Flow Diagram* (DFD) dapat dilihat pada Gambar 3.32 untuk kemudian disebut sebagai DFD *level* 0.



Gambar 3.32. *Data flow diagram* sistem *level* 0

Secara gambaran umum DFD *level* 0 adalah sebagai berikut, user memberikan masukan citra kepada sistem, untuk kemudian sistem akan mengembalikan keluaran pada user berupa citra dan karakter plat nomor yang terdeteksi, dan juga riwayat hasil pengenalan yang telah dilakukan. Kemudian pada Gambar 3.33 alur dari data dijelaskan lebih lanjut untuk kemudian disebut sebagai DFD *level* 1.



Gambar 3.33. *Data flow diagram* sistem *level* 1

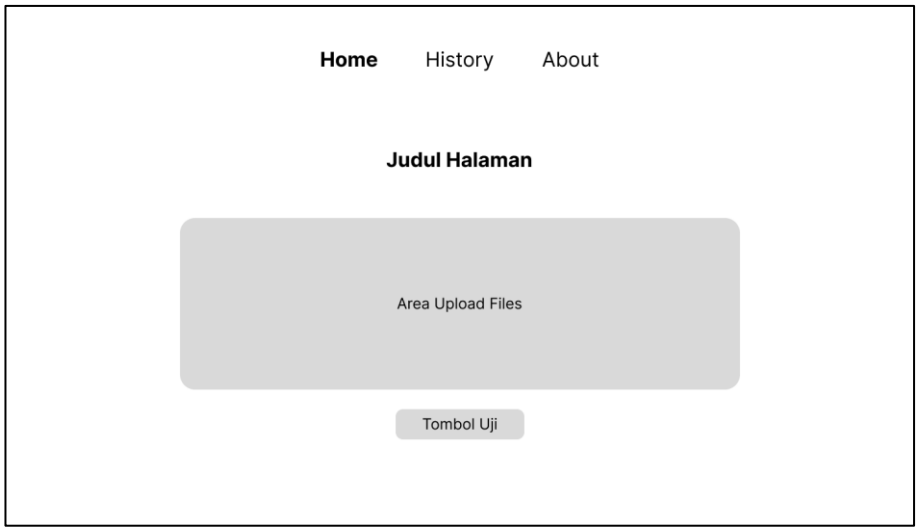
Pada DFD *level* 1, dijelaskan secara lebih lanjut bagaimana alur data akan diolah dalam sistem. Sistem akan memiliki enam proses utama untuk dapat memproses citra kemudian mengetahui hasil pengenalan dan evaluasi hasil.

3.2.3. Membangun *Prototype*

Prototype akan dibangun dengan desain tampilan antarmuka sebagai berikut,

1. Halaman Utama

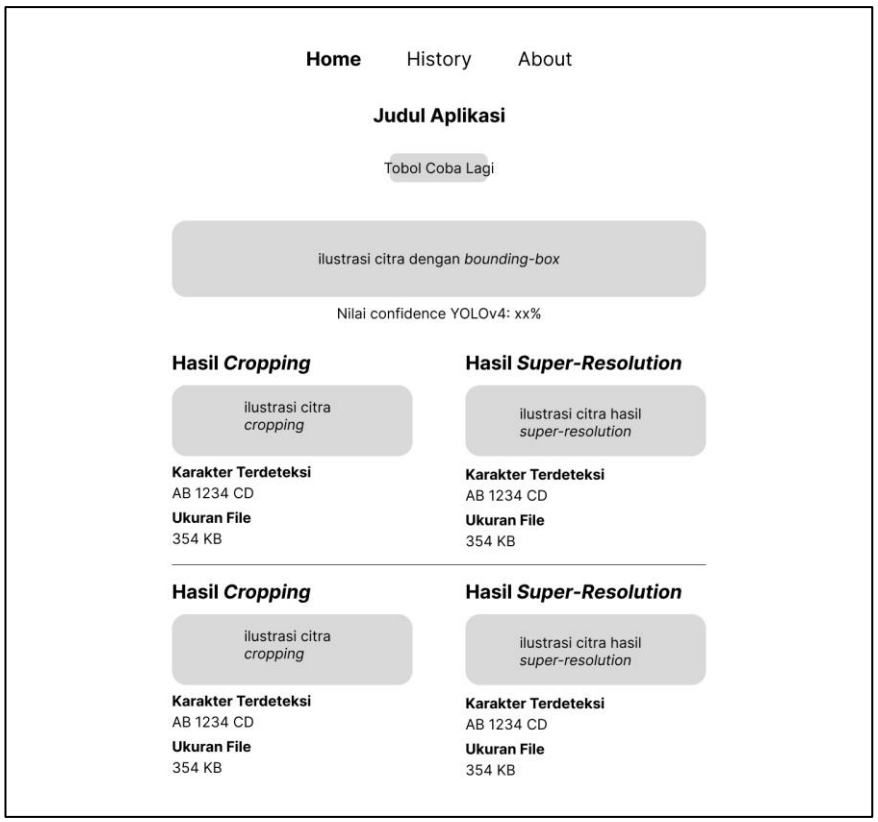
   Halaman utama atau halaman uji coba akan menampilkan formulir untuk memilih file citra yang akan dilakukan proses pengenalan karakter, dengan tampilan antar muka dapat dilihat pada Gambar 3.34.



Gambar 3.34. Tampilan antar muka halaman utama

2. Halaman Hasil Pengenalan

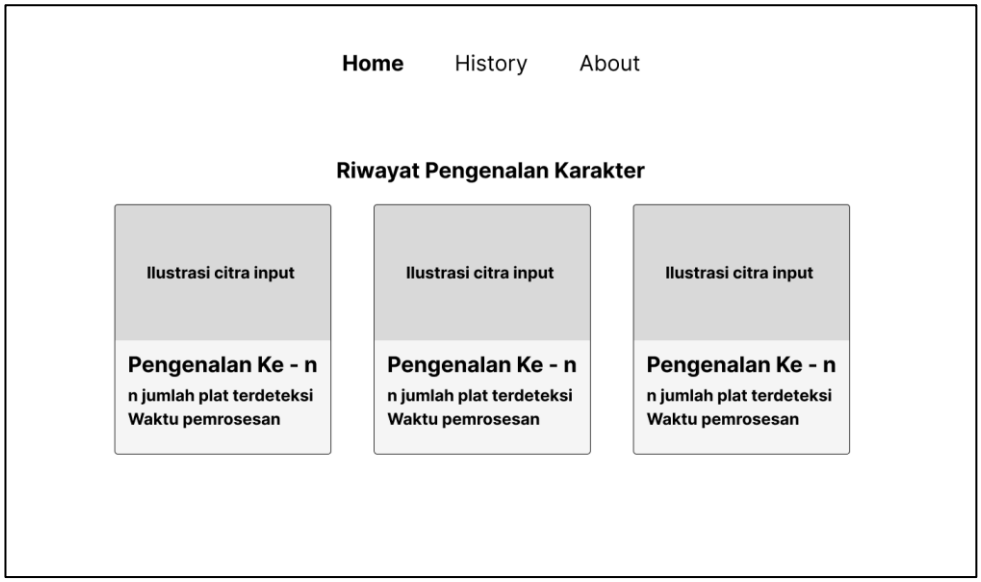
User akan diarahkan kedalam halaman hasil pengenalan setelah melakukan *upload* file citra pada halaman utama atau halaman uji coba. Pada halaman ini akan ditampilkan citra asli, lokalisasi citra plat nomor, dan citra generasi citra *super-resolution*. Beberapa data pendukung lain juga akan ditampilkan, yaitu berupa nilai *confidence* dari lokalisasi YOLOv4, hasil karakter dari proses OCR untuk masing-masing lokalisasi citra plat nomor dan hasil generasi citra *super-resolution*. Tampilan antar muka halaman hasil pengenalan dapat dilihat pada Gambar 3.35.



Gambar 3.35. Tampilan antar muka halaman hasil pengenalan

3. Halaman Riwayat

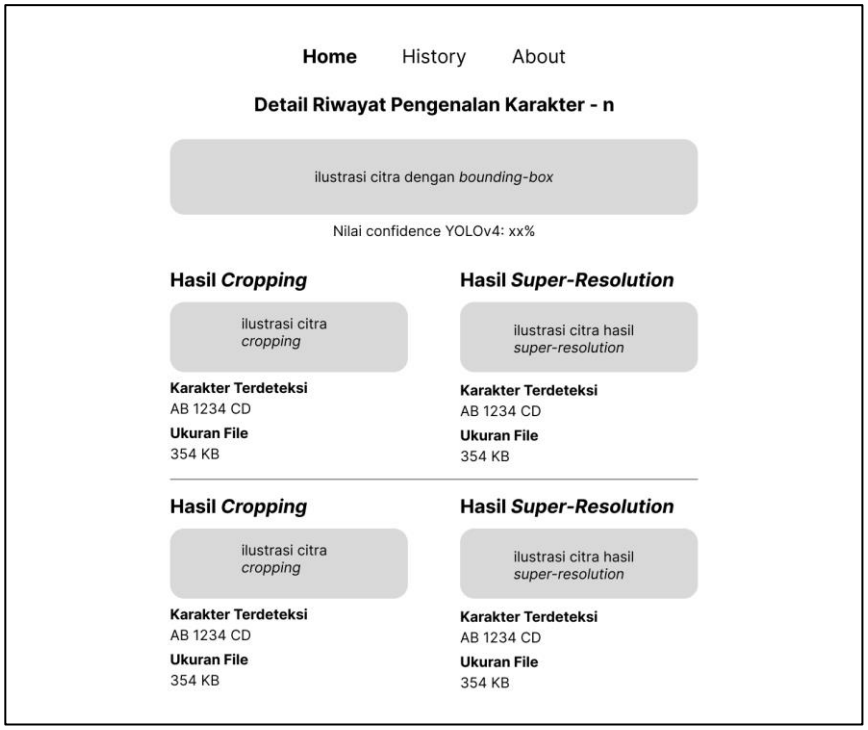
Untuk setiap hasil dari proses pengenalan akan disimpan dan ditampilkan riwayatnya pada halaman ini, tampilan antarmuka halaman riwayat dapat dilihat pada Gambar 3.36.



Gambar 3.36. Tampilan antarmuka halaman riwayat

4. Halaman Detail Riwayat

Pada halaman detail riwayat memiliki antarmuka yang serupa dengan halaman hasil pengenalan. Antarmuka halaman detail riwayat dapat dilihat pada Gambar 3.37.



Gambar 3.37. Tampilan antar muka detail riwayat

3.2.4. Pengujian Sistem

Pengujian sistem dilakukan dengan metode *blackbox testing* untuk mengetahui apakah seluruh fungsi sistem dapat berjalan dengan baik, dan juga dapat mengetahui adanya *bug* saat menjalankan sistem, yang kemudian dapat menjadi bahan evaluasi untuk memperbaiki sistem. Adapun pengujian yang akan dilakukan dapat dilihat pada Tabel 3.12.

Tabel 3.12. Parameter pengujian sistem

No	Halaman	Parameter Uji	Hasil	Keterangan
1	Halaman Utama	Dapat menerima citra berformat jpg, jpeg.	Sukses/ Gagal	
2	Halaman Hasil Pengenalan	Dapat memberikan keluaran citra yang diunggah, plat nomor ter-*cropping*, generasi citra *super-resolution*, hasil karakter dikenali, beserta ukuran file dari citra *super-resolution*.	Sukses/ Gagal	
3	Halaman Riwayat Pengenalan	Dapat menampilkan daftar seluruh riwayat pengenalan plat nomor sebelumnya.	Sukses/ Gagal	
4	Halaman Detail Riwayat Pengenalan	Dapat menampilkan detail dari salah satu riwayat pengenalan terpilih.	Sukses/ Gagal	

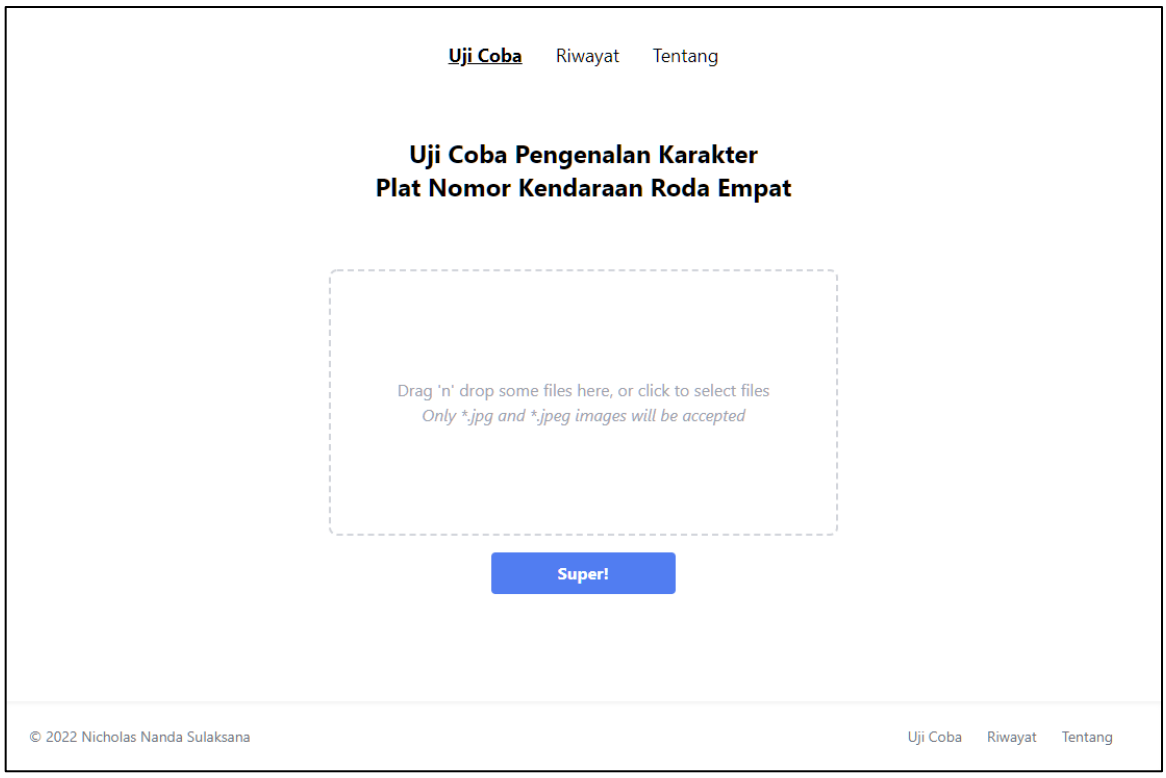
BAB IV
IMPLEMENTASI DAN HASIL

4.1. Implementasi

Implementasi rancangan *prototype* dari sitem yang telah dibuat merupakan tampilan antar muka fungsional berdasarkan rancangan yang telah dibuat.

4.1.1. Halaman Utama

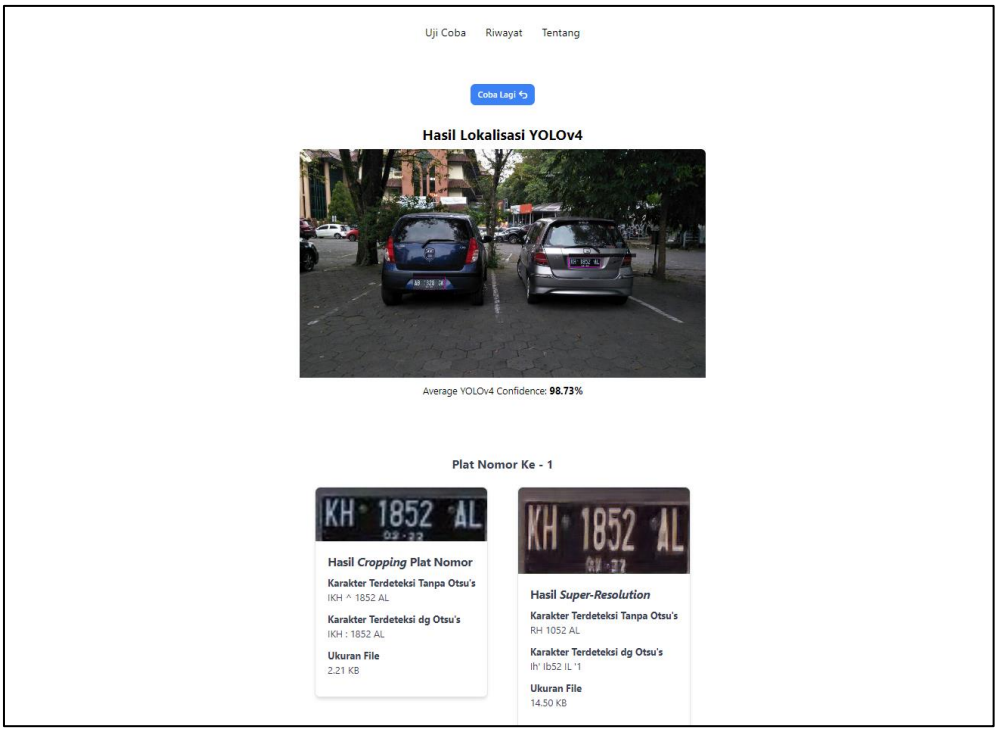
Berupa tampilan antar muka bagi user untuk dapat mengunggah citra yang akan dilakukan proses *license plate recognition*, dengan hasil antar muka dapat dilihat pada Gambar 4.1.



Gambar 4.1. Antar muka halaman utama

4.1.2. Halaman Hasil Pengenalan

Merupakan halaman lanjutan dari halaman utama, dimana akan ditampilkan citra yang telah diunggah sebelumnya dengan prediksi *bounding-box* dan nilai rata-rata *confidence* dari plat nomor yang dideteksi. Ditampilkan juga hasil ekstraksi plat nomor atau *cropping* plat nomor yang terdeteksi dan hasil generasi citra *super-resolution* berdasarkan citra *cropping*. Dengan data tambahan yaitu hasil karakter dikenali dan ukuran dari citra *cropping* sebelum dan sesudah dikenai SRGAN. Tampilan antar muka halaman hasil pengenalan selanjutnya dapat dilihat pada Gambar 4.2.

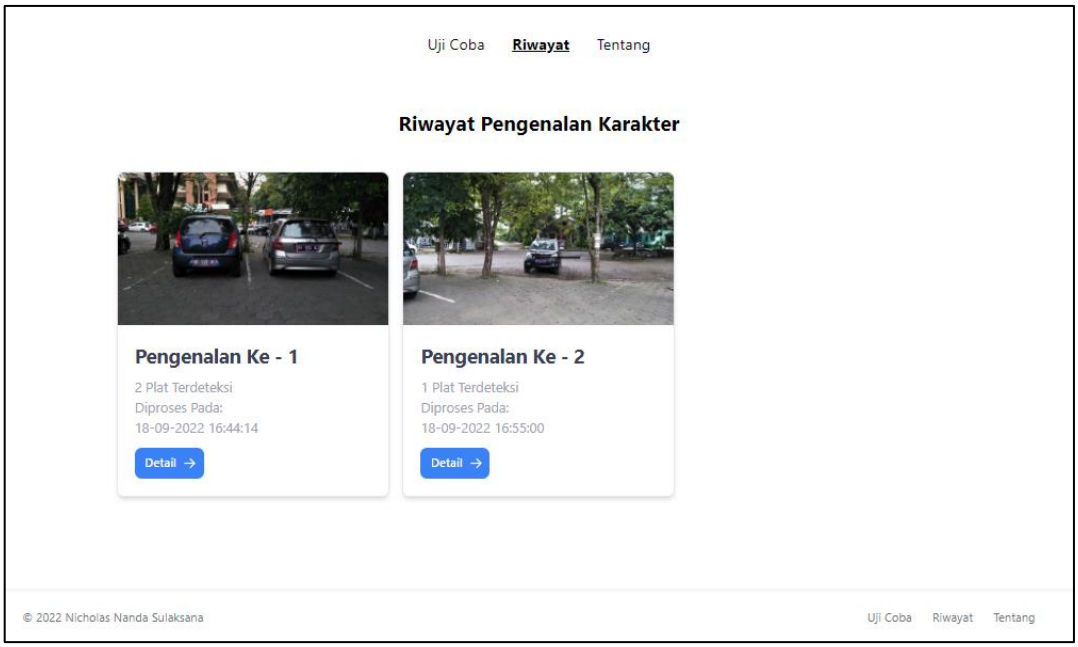


Gambar 4.2. Antar muka halaman hasil pengenalan

Hasil citra *cropping* akan berdasarkan seluruh hasil deteksi YOLOv4, menurun kebawah dengan informasi data tambahan yang serupa.

4.1.3. Halaman riwayat pengenalan

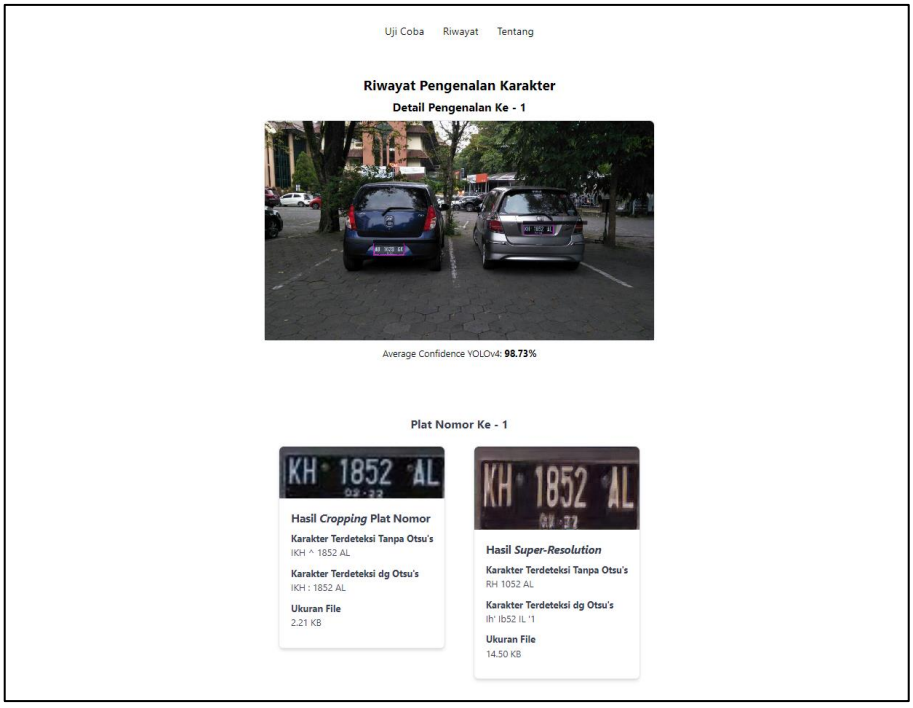
Dalam halaman riwayat, akan ditampilkan ringkasan umum dari seluruh riwayat proses *license plate recognition* yang pernah dilakukan, beserta halaman rujukan untuk mengetahui detail dari setiap pengenalan yang ada. Tampilan antar muka halaman riwayat selanjutnya dapat dilihat pada Gambar 4.3.



Gambar 4.3. Antar muka halaman riwayat pengenalan

4.1.4. Halaman detail riwayat pengenalan

Merupakan halaman lanjutan dari halaman riwayat pengenalan, memiliki tampilan antar muka serupa dengan halaman hasil pengenalan yang dapat dilihat pada Gambar 4.4.



Gambar 4.4. Antar muka halaman detail riwayat pengenalan

4.2. Hasil

Hasil penelitian akan dijelaskan secara kronologis berdasarkan rancangan alur metodologi, dengan hasil evaluasi dan analisis pada tahapan terakhir.

4.2.1. Pengumpulan data

Pengumpulan data latih untuk proses *transfer learning* YOLOv4 menggunakan bantuan OIDv4 ToolKit milik (Vittorio, 2018) untuk dapat juga digunakan dalam mengunduh data pada versi 6. Proses pengunduhan data latih atau *trainning* sebanyak 5368 citra, dengan menggunakan perintah sebagaimana Gambar 4.5.

```
!python main.py downloader --classes 'Vehicle registration plate' --type_csv train -y

    [INFO] | Downloading Vehicle registration plate.
   [ERROR] | Missing the class-descriptions-boxable.csv file.
[DOWNLOAD] | Automatic download.
...145%, 0 MB, 17340 KB/s, 0 seconds passed
[DOWNLOAD] | File class-descriptions-boxable.csv downloaded into OID/csv_folder/class-description
s-boxable.csv.
   [ERROR] | Missing the train-annotations-bbox.csv file.
[DOWNLOAD] | Automatic download.
...43%, 493 MB, 171932 KB/s, 2 seconds passed
...97%, 1105 MB, 174061 KB/s, 6 seconds passed
---------------------------Vehicle registration plate---------------------------
    [INFO] | Downloading train images.
    [INFO] | [INFO] Found 5368 online images for train.
    [INFO] | Download of 5368 images in train.
100%|████████████████████████████████████| 5368/5368 [19:12<00:00,  4.66it/s]
    [INFO] | Done!
    [INFO] | Creating labels for Vehicle registration plate of train.
    [INFO] | Labels creation completed.
```

Gambar 4.5. Mengunduh data *trainning* YOLOv4

Data *validation* diunduh sebanyak 386 citra dengan menggunakan perintah yang serupa , dapat dilihat pada Gambar 4.6. Ilustrasi citra yang didapat dari dataset OIDv6 dapat dilihat pada Gambar 3.2.

```
!python main.py downloader --classes 'Vehicle registration plate' --type_csv validation -y

    [INFO] | Downloading Vehicle registration plate.
   [ERROR] | Missing the validation-annotations-bbox.csv file.
[DOWNLOAD] | Automatic download.
...100%, 16 MB, 118430 KB/s, 0 seconds passed
[DOWNLOAD] | File validation-annotations-bbox.csv downloaded into OID/csv_folder/validation-annot
ations-bbox.csv.

---------------------------Vehicle registration plate---------------------------
    [INFO] | Downloading validation images.
    [INFO] | [INFO] Found 386 online images for validation.
    [INFO] | Download of 386 images in validation.
100%|████████████████████████████████████| 386/386 [01:22<00:00,  4.70it/s]
    [INFO] | Done!
    [INFO] | Creating labels for Vehicle registration plate of validation.
    [INFO] | Labels creation completed.
```

Gambar 4.6. Mengunduh data *validation* YOLOv4

Dataset untuk proses *trainning* SRGAN diperoleh dari (Agustsson and Timofte, 2017) yaitu DIV2K dataset. Digunakan tipe dataset bicubic downscaling x4 *low-resolution image* bersamaan pasangan citra *high-resolution* untuk masing-masing citra *low-resolution*. Ilustrasi dataset DIV2K dapat dilihat pada Gambar 3.3.

4.2.2. *Transfer learning* YOLOv4

Untuk dapat melakukan lokalisasi plat nomor menggunakan library darknet YOLOv4 (Bochkovskiy, Wang and Liao, 2020), dilakukan konfigurasi *hyperparameters* sesuai dengan Tabel 3.2 pada file ‘yolov4-obj.cfg’, dengan hasil dapat dilihat pada Gambar 4.7.

```
 6  # Training
 7  batch=64
 8  subdivisions=16
 9  width=416
10  height=416
21  max_batches = 6000
23  steps=4800,5400
```

Gambar 4.7. Konfigurasi *hyperparameter* YOLOv4

Kemudian menyesuaikan nilai *filters* dan *classes* pada setiap layer *convolutional*, yaitu pada baris 964 dan 971 untuk layer pertama, baris 1052 dan 1059 untuk layer *convolutional* kedua, dan terakhir pada baris 1140 dan 1147. Hasil konfigurasi layer dapat dilihat pada Gambar 4.8, sebagai hasil pengubahan pada baris pertama.

```
960  [convolutional]
964  filters=18
968  [yolo]
971  classes=1
```

Gambar 4.8. Konfigurasi *hyperparameter* pada layer *convolutional* YOLOv4

Membuat file ‘obj.names’ untuk menyimpan nama *class* yang akan digunakan dan sesuai dengan klasifikasi *class* pada OIDv6 yaitu “*Vehicle registration plate*”, dengan hasil dapat dilihat pada Gambar 4.9.

```
1  Vehicle registration plate
2
```

Gambar 4.9. Konfigurasi *class* pada file ‘obj.names’

Konfigurasi pada file ‘obj.data’ untuk menyimpan alamat dari daftar citra *trainning* dan *validation*, konfigurasi jumlah *class* beserta alamat file ‘obj.names’ dan juga konfigurasi alamat file untuk menyimpan hasil *trainning weights*. Hasil konfigurasi file ‘obj.data’ dapat dilihat pada Gambar 4.10.

```
1  classes = 1
2  train = OIDv4_ToolKit/OID/Dataset/train/train.txt
3  valid = OIDv4_ToolKit/OID/Dataset/validation/validation.txt
4  names = obj.names
5  backup = darket_backup/
6
```

Gambar 4.10. Konfigurasi file ‘obj.data’

Mengunduh *pre-trainned weights* YOLOv4 milik (Bochkovskiy, Wang and Liao, 2020) menggunakan perintah terminal sebagaimana pada Gambar 4.11. *Pre-trainned weights* akan digunakan untuk proses *transfer learning* agar YOLOv4 dapat melakukan lokalisasi atau deteksi plat nomor kendaraan roda empat.

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
                                                                                    Shell Script
--2022-10-13 10:33:24--
https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
Resolving github.com (github.com)... 192.30.255.112
Connecting to github.com (github.com)|192.30.255.112|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/75388965/ba4b6380-
HTTP request sent, awaiting response... 200 OK
Length: 257717640 (246M) [application/octet-stream]
Saving to: 'yolov4.weights'
yolov4.weights      100%[===================>] 245.78M  7.39MB/s    in 20s
2022-10-13 10:33:45 (12.0 MB/s) - 'yolov4.weights' saved [257717640/257717640]
```

Gambar 4.11. Perintah untuk mengunduh *pre-trainned weights YOLOv4*

Konfigurasi *makefile* dengan menggunakan perintah ‘`!sed`’ atau *stream editor* untuk mengubah *makefile* agar dapat mengalokasikan *Graphics Processing Unit* (GPU) sebagai sumber daya proses *trainning* YOLOv4, dengan mengubah nilai OPENCV, GPU, dan CUDNN bernilai 1 atau *True*, dilanjutkan dengan eksekusi *makefile* agar library darknet dapat digunakan sebagai perintah terminal. Hasil eksekusi dapat dilihat pada Gambar 4.12.

```
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!make
```

Shell Script

```
Output exceeds the size limit. Open the full output data in a text editor
chmod +x *.sh
g++ -std=c++11 -std=c++11 -Iinclude/ -I3rdparty/stb/include -DOPENCV `pkg-config --cflags opencv4 2>
/dev/null || pkg-config --cflags opencv` -DGPU -I/usr/local/cuda/include/ -DCUDNN -DCUDNN_HALF -Wall -
```

Gambar 4.12. Konfigurasi *makefile* YOLOv4

Proses *trainning* atau *transfer learning* dari *pre-trainned weights* dilakukan dengan menggunakan perintah, dan sebagian output sebagaimana Gambar 4.13.

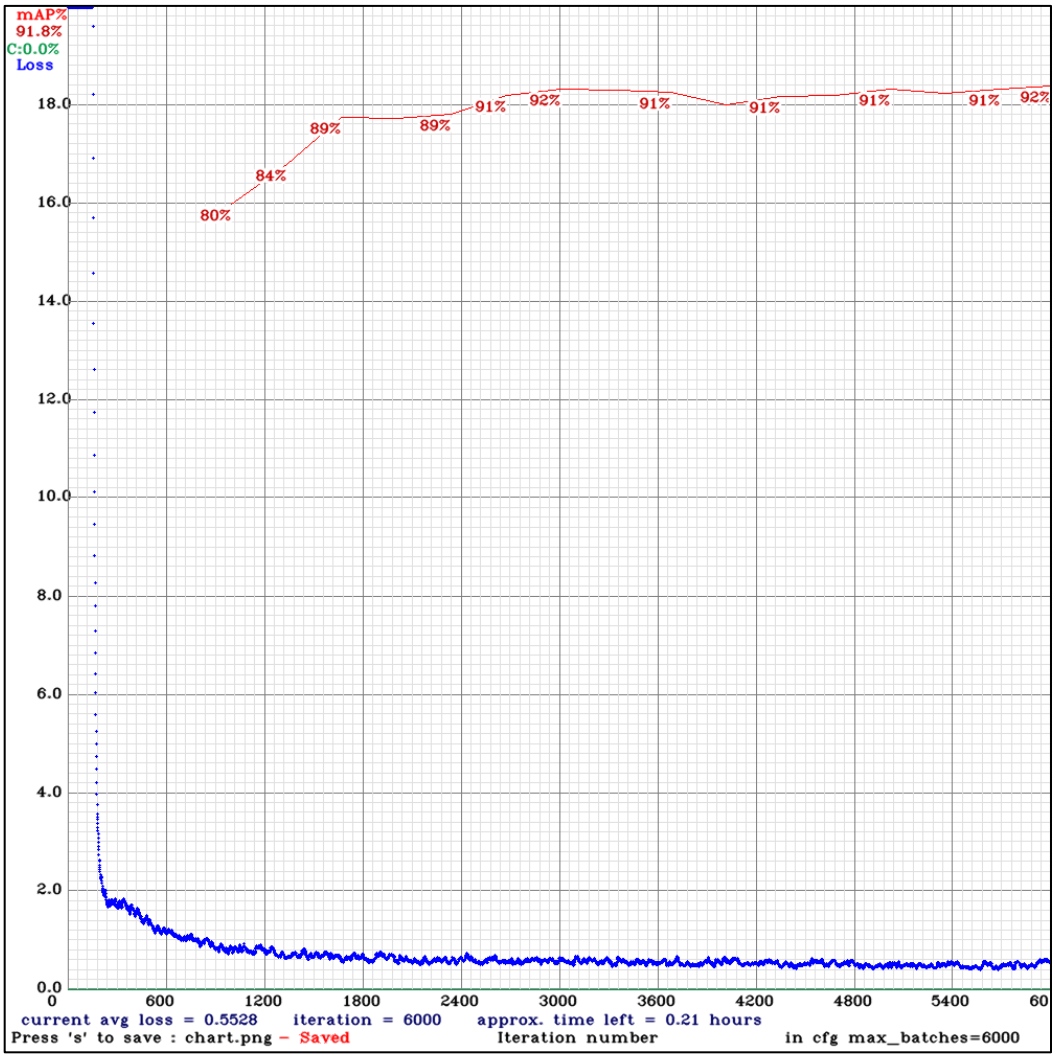
```
!./darknet detector train obj.data yolov4-obj.cfg yolov4.weights -dont_show -map
```

Python

```
Output exceeds the size limit. Open the full output data in a text editor
 CUDA-version: 11030 (11060), cuDNN: 8.2.0, CUDNN_HALF=1, GPU count: 1
 CUDNN_HALF=1
 OpenCV version: 3.2.0
 Prepare additional network for mAP calculation...
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU:
0.634909), count: 4, class_loss = 0.400673, iou_loss = 12.920578, total_loss = 13.321251
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU:
0.553438), count: 5, class_loss = 0.249429, iou_loss = 1.131992, total_loss = 1.381421
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU:
0.370741), count: 4, class_loss = 0.466068, iou_loss = 0.064089, total_loss = 0.530157
 total_bbox = 206481, rewritten_bbox = 0.042135 %
```

Gambar 4.13. *Trainning* darknet YOLOv4

Dengan format perintah adalah '`!./darknet detector train <alamat_file_obj.data> <alamat_file_cfg> <alamat_pre_trained_weights_YOLOv4> -dont_show -map`', dengan *flag* atau parameter '`-dont_show`' untuk tidak mengeluarkan nilai loss pada output terminal saat trainning berjalan, dan flag '`-map`' untuk menghasilkan keluaran grafik hasil mean average precision trainning dan grafik nilai average loss, hasil grafik dapat dilihat pada Gambar 4.14.



Gambar 4.14. Hasil grafik *trainning* YOLOv4

Berdasarkan grafik pada Gambar 4.14 dapat diketahui bahwa,

1. Sumbu X pada grafik menunjukan jumlah nilai *trainning epoch*,
2. Sumbu Y pada grafik sebagai acuan nilai desimal,
3. Garis biru adalah grafik nilai *mean average loss* (mAP) untuk setiap *epoch*, dengan semakin kecil nilainya akan semakin baik,
4. Garis merah pada grafik adalah nilai *mean average precision* untuk setiap 300 epoch dimulai dari epoch ke 1000 pertama, dengan semakin besar presentasenya semakin baik,
5. Model tidak mengalami *underfitting* maupun *overfitting* berdasarkan beberapa kondisi yaitu, grafik biru atau *average loss* mengalami penurunan drastis pada 300 *epoch* awal dan penurunan secara konstan dan stabil pada *epoch* setelahnya. Nilai mAP mengalami peningkatan konstan dan stabil pada rata-rata nilai 91%.

4.2.3. Lokalisasi plat nomor dengan YOLOv4

Hasil model dari darknet YOLOv4 akan disimpan pada file berekstensi '.weights' dengan model terbaik akan disimpan dengan nama file 'yolov4-obj_best.weights'. Model terbaik akan digunakan dalam lokalisasi plat nomor dengan menggunakan perintah yang dapat dilihat pada Gambar 4.15.

```
!./darknet detector test obj.data yolov4-obj.cfg yolov4-obj_best.weights -ext_output testLokalisasi.jpg -dont_show -out predictResult.txt

 CUDA-version: 11020 (11020), cuDNN: 8.1.1, CUDNN_HALF=1, GPU count: 1
 CUDNN_HALF=1
 OpenCV version: 3.2.0
 0 : compute_capability = 750, cudnn_half = 1, GPU: Tesla T4
net.optimized_memory = 0
mini_batch = 1, batch = 32, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
Done! Loaded 162 layers from weights-file
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
/content/drive/Shareddrives/UPNYK/LPR/testingLP/testLokalisasi.jpg: Predicted in 53.118000 milli-seconds.
license_plate: 94%      (left_x:  310   top_y:  896   width:  536   height:  216)
```

Gambar 4.15. Lokalisasi plat nomor menggunakan best.weights

Dengan format perintah adalah '`!./darknet detector test <alamat_file_obj.data> <alamat_file_cfg> <alamat_pre_trained_weights_YOLOv4> -ext_ouput <alamat_input_citra> -dont_show -out <alamat_file_Bbox.txt>`', dengan *flag* atau parameter '`-ext_ouput`' untuk menyimpan hasil citra bersama dengan *bounding-box* plat nomor untuk selanjutnya dapat dilihat pada Gambar 4.16. Parameter '`-dont_show`' untuk tidak mengeluarkan nilai loss pada output terminal. Parameter '`-out`' untuk menyimpan nilai koordinat relatif *bounding-box* dalam file berekstensi '.txt', keluaran file txt dapat dilihat pada Gambar 4.16.

```
3     "frame_id":1,
4     "filename":"testLokalisasi.jpg",
5       "objects":
6       [
7         {
8          "class_id":0,
9          "name":"license_plate",
10         "relative_coordinates":
11          {
12           "center_x":0.390324,
13           "center_y":0.668923,
14           "width":0.276156,
15           "height":0.108176
16          },
17        "confidence":0.993564
18        }
```

Gambar 4.16. File predictedResult.txt

Berdasarkan koordinat yang didapat pada file "predictedResult.txt" maka dapat diketahui dan dibuat visualisasi *bounding-box* seperti pada Gambar 4.17.

Gambar 4.17. Hasil visualisasi *bounding-box*

4.2.4. *Pre-processing* I

Hasil file predictionResult.txt dilakukan ekstraksi informasi dari koordinat relatif menurut deteksi setiap objek menggunakan Algoritma 7.

Algoritma 7 : Ektraksi informasi file txt

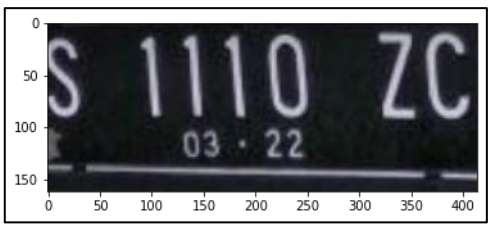
```
INIT frameId, classId, center_x, center_y, yoloWidth, yoloHeight, confidence
SET output_coord TO EMPTY LIST
BEGIN
   FOR EVERY frame DO:
      FOR EVRY objects DO:
         frameId = frame_id
         classId = objects['class_id']
         center_x = objects['center_x']
         center_y = objects['center_y']
         yoloWidth = objects['width']
         yoloHeight = objects['height']
         confidence = objects['confidence']
         [frameId, classId, center_x, center_y, yoloWidth, yoloHeight,
          confidence] APPEND TO output_coord
      ENDFOR
   ENDFOR
END
```

Hasil ekstraksi informasi koordinat relatif *bounding-box* berdasarkan file .txt pada Gambar 4.16 digunakan untuk mengekstraksi plat nomor dari input citra atau proses *cropping* menggunakan Algoritma 4. Dengan nilai imgHeight adalah 1500, imgWidth adalah 1500, center_x adalah 0.390324, center_y adalah 0.668923, yoloWidth adalah 0.276156, dan yoloHeight adalah 0.108176. Maka didapati hasil koordinat absolut x1 = 378, y1 = 922, x2 = 792, dan y2 = 1084. Berdasarkan koordinat absolut kemudian dilakukan proses *cropping* yaitu menggunakan Algoritma 8.

Algoritma 8 : Cropping citra

```
INIT citra_arry, x1, x2, y1, y2
BEGIN
   SET citra_arry INTO OPEN file_citra
   SET citra_arry = citra_arry[y1:y2, x1:x2]
   SAVE citra_arry INTO file_citra_plat
END
```

Hasil dari Algoritma 8 adalah hasil *cropping* citra input yang sebelumnya telah diketahui koordinat *bounding-box* dengan visualisasi pada Gambar 4.17 yaitu citra dengan plat nomor saja, dapat dilihat pada Gambar 4.18.



Gambar 4.18. Citra plat nomor hasil *cropping*

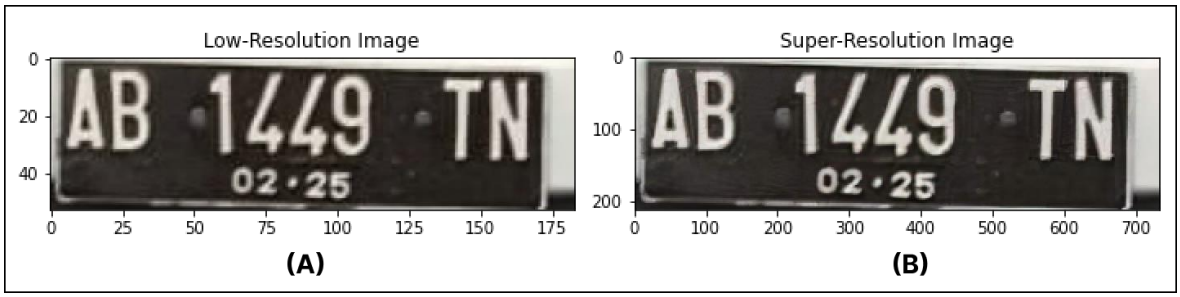
Beberapa citra plat nomor didapat secara manual atau tidak melalui proses deteksi YOLO akan dilakukan proses *resizing* untuk mensimulasikan citra *low-resolution*, yaitu 32x64 pixel, dan *high-resolution* berukuran 256x128 pixel. Proses *resizing* dilakukan menggunakan algoritma pillow image resize, dengan hasil dapat dilihat pada Gambar 4.19.

Gambar 4.19. Citra hasil *resizing*

Dapat dilihat pada Gambar 4.19 adalah citra hasil *resize*, dengan Gambar 4.19 (A) adalah citra sintetis *low-resolution* berukuran 64x32 pixel, Gambar 4.19 (B) adalah hasil citra sintetis *high-resolution* berukuran 256x128 pixel. Hasil citra *resize low-resolution* akan dilakukan penambahan *noise* dan diteruskan langsung untuk dikenali karakternya, maupun dilakukan *scaling* atau *normalisasi* nilai matriksnya untuk kemudian dapat diproses oleh SRGAN terlebih dahulu sebelum dilakukan proses pengenalan karakter.

4.2.5. Generasi citra *super-resolution*

Sebelum model digunakan untuk generasi citra *super-resolution*, SRGAN dilakukan proses *trainning* berdasarkan Algoritma 2 sesuai dengan alur SRGAN pada Gambar 3.12. Proses trainning menghasilkan model yang berekstensi .h5 yang digunakan untuk proses generasi citra *super-resolution*, dengan hasil proses generasi citra dapat dilihat pada Gambar 4.20 (B). Dengan perbandingan citra *low-resolution* pada Gambar 4.20 (A).

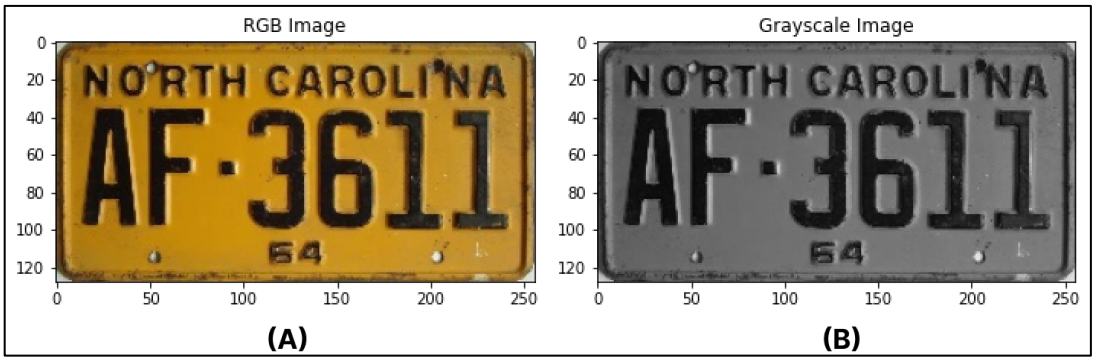


Gambar 4.20. Hasil generasi citra *super-resolution*

Dapat diketahui berdasarkan Gambar 4.20, hasil citra *super-resolution* Gambar 4.20 (B) memiliki kepadatan atau ukuran citra empat kali lebih yaitu 732x212 pixel, dari pada citra input *low-resolution* Gambar 4.20 (A) yaitu 183x53 pixel.

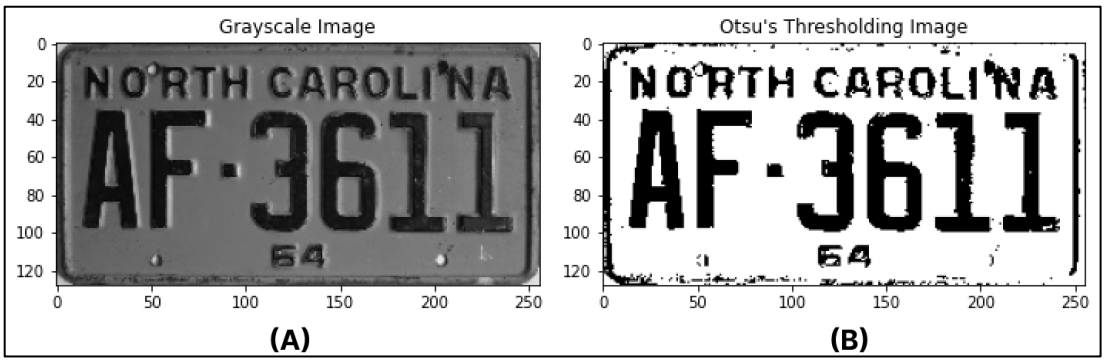
4.2.6. *Pre-processing* II

Hasil citra *super-resolution* dan juga *resizing* pada citra *cropping* dilakukan proses *grayscaling*, yaitu mengubah nilai warna citra yang sebelumnya 3 *channel* yaitu *Red Green Blue* (RGB) menjadi 1 *channel* dengan menggunakan Persamaan 2.6 atau dengan bantuan fungsi cv2.cvtColor untuk mengolah citra atau cv2.IMREAD_GRAYSCALE saat membaca citra input. Hasil dari proses *grayscaling* dapat dilihat pada Gambar 4.21 (B).



Gambar 4.21. Citra hasil *grayscaling*

Hasil citra *grayscaling* tersebut dilanjutkan pada proses *otsu's thresholding and binarization*, dengan ilustrasi proses sesuai pada Gambar 3.25. Menggunakan bantuan fungsi cv2.THRESH_BINARY+cv2.THRESH_OTSU yaitu berdasarkan Persamaan 2.7 didapati hasil citra pada Gambar 4.22 (B).



Gambar 4.22. Citra hasil *otsu's thresholding and binarization*

4.2.7. Hasil evaluasi model

Hasil dari model YOLOv4 untuk deteksi atau lokalisasi plat nomor kendaraan roda empat berdasarkan rancangan evaluasi pada Tabel 3.6 dilakukan evaluasi dengan menghitung nilai *recall* berdasarkan Persamaan 2.15, nilai akurasi berdasarkan Persamaan 2.16, dan nilai F1 *Score* berdasarkan Persamaan 2.17. Dengan hasil evaluasi pada Tabel 4.1.

Tabel 4.1. Hasil *confussion matrix* deteksi YOLOv4

Citra Ke-	Jumlah Plat Nomor Kendaraan		TP	FP	TN	FN	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
	Roda Empat	Bukan Roda Empat							
1	3	0	3	0	0	0	100%	100%	1,00
2	4	0	3	0	0	1	75%	75%	0,86
3	1	0	1	0	0	0	100%	100%	1,00
4	1	0	1	0	0	0	100%	100%	1,00
5	1	0	1	0	0	0	100%	100%	1,00
6	1	0	1	0	0	0	100%	100%	1,00
7	2	0	1	0	0	1	50%	50%	0,67
8	2	0	1	0	0	1	50%	50%	0,67
9	1	0	1	0	0	0	100%	100%	1,00

Tabel 4.2. Hasil *confussion matrix* deteksi YOLOv4 (lanjutan)

Citra Ke-	Jumlah Plat Nomor Kendaraan		TP	FP	TN	FN	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
	Roda Empat	Bukan Roda Empat							
10	7	5	5	0	5	2	71%	83%	0,83
11	2	0	1	0	0	1	50%	50%	0,67
12	4	1	4	0	1	0	100%	100%	1,00
13	3	3	1	0	3	2	33%	67%	0,50
14	2	0	2	0	0	0	100%	100%	1,00
15	1	1	1	0	1	0	100%	100%	1,00
16	13	0	10	0	0	3	77%	77%	0,87
17	23	0	13	0	0	10	57%	57%	0,72
18	2	1	2	0	1	0	100%	100%	1,00
19	13	22	7	1	21	6	54%	80%	0,67
20	12	0	8	0	0	0	100%	100%	1,00
Rata-rata							**81%**	**84%**	**0,87**

Hasil dari proses pengenalan plat nomor dilakukan evaluasi berdasarkan rancangan pada Tabel 3.8 dengan nilai *recall* berdasarkan Persamaan 2.15, nilai akurasi berdasarkan Persamaan 2.16, dan nilai F1 *Score* berdasarkan Persamaan 2.17 dan nilai matriks didapat berdasarkan Tabel 3.7. Maka didapati hasil sebagaimana Tabel 4.3 dan Tabel 4.5 untuk citra hasil *cropping* deteksi YOLOv4 sebelum dikenai proses SRGAN.

Dengan masing-masing hasil citra dapat dilihat pada Lampiran A, dengan urutan citra pada seluruh tabel hasil sama dengan urutan citra pada Lampiran A.

Tabel 4.3. Hasil *confussion matrix* citra *cropping* YOLOv4 tanpa proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
1	R 8585 CW	0585 CUI	5	1	0	4
2	H 311 CI	311	3	0	0	5
3	AB 1449 TN 02.25	AB 1449 TM 02.25	15	0	1	1
4	AB 1111 YE 08.27	AB VV YE 0,8<27	11	1	0	5
5	AB 1823 DF 02.20	AB 1823 DF 02.20	15	0	1	0
6	AB 1623 AY 01.26	AB 1623 AYI 01.26	15	1	1	0
7	AB1382LX 09.24	[B1382LX	7	0	0	7
8	B 1599 TYC 02.24	B 1599 TYC 02.22	13	0	1	1
9	AB 1832 MH 08.26	AB 1832 MH 00 < 26	14	2	0	1
10	N 1226 BJ	"1226" BJ	7	2	0	2
11	KH 1852 AL 02.22	IKH 1852 AL}	10	2	0	6
12	AB 1628 GK 10.21	AB 1628 GK	10	0	0	6
13	B 2532 PFN	8 2532 PFN	9	0	0	1
14	B 9147 BCT	B 9107 BCT	9	0	0	1
15	9212 M	19212	4	1	0	2
16	B 9357 KAS	AIN	0	2	0	9
17	B 8899 AAN 08.24	B 8899 AAN 08 24	15	0	0	1
18	BL 1614 C	BBL 7612	5	1	0	4

Tabel 4.4. Hasil *confussion matrix* citra *cropping* YOLOv4 tanpa proses otsu's (lanjutan)

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
19	B 2655 SRY 07.25	3 2655 SRY DIAE	10	0	0	6
20	1095 F 03.22	1095	4	0	0	8
21	B 2641 SIT 11.24	B 2641 SIT	10	0	0	6
22	2099 PFY	2099 PFV	8	0	0	0
23	B 4675 TTY	-	0	0	0	10
24	B 1553 UOY	8 1553 UBY	8	0	0	2
25	B 2655 KOP 12.24	3 2655 KOP	9	0	0	6
26	A 1893 YA	1093	3	0	0	6
27	B 2849 KKD	'2849 KKD	8	0	0	2
28	B 1732 KRY	7732 KRL	6	0	0	4
29	B 1628 JI 04.25	B 1628	6	0	0	9
30	PA 61 KK 08.21	PA 6} KK	7	0	0	7

Didapati temuan, yaitu ketika evaluasi pengenalan karakter jika tidak melalui tahapan *otsu's thresholding* pada *pre-processing II* dapat dilihat pada Tabel 4.3 dan ketika pengenalan karakter terlebih dahulu melalui tahapan *otsu's thresholding* dengan hasil pada Tabel 4.5 menunjukan hasil karakter dikenali yang berbeda.

Tabel 4.5. Hasil confussion matrix citra cropping YOLOv4 dengan proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
1	R 8585 CW	RA Q	1	1	0	8
2	H 311 CI	# 341 CI	6	0	0	2
3	AB 1449 TN 02.25	AB 1L49 T 02.25	14	0	0	2
4	AB 1111 YE 08.27	AB: M YE	6	1	0	10
5	AB 1823 DF 02.20	AB 1823 DF 0A'40	13	0	0	3
6	AB 1623 AY 01.26	AB 1623 AYI 01-38	13	1	0	3
7	AB1382LX 09.24	B1382LX	7	0	0	7
8	B 1599 TYC 02.24	B 1599 TYC 22	13	0	0	4
9	AB 1832 MH 08.26	AB 1832 HH	9	0	0	7
10	N 1226 BJ	"1226" 8J	6	2	0	3
11	KH 1852 AL 02.22	IH 1852 AL;	9	1	0	2
12	AB 1628 GK 10.21	AB 1628 GK	10	0	0	6
13	B 2532 PFN	B337AM	2	0	0	8
14	B 9147 BCT	8 %187 ECT	5	0	0	6
15	9212 M	9212	4	0	0	2
16	B 9357 KAS	9137 L|R	2	0	0	5
17	B 8899 AAN 08.24	B 8899 AAN 08 2 4	16	1	0	1
18	BL 1614 C	BL 7614 €	7	0	0	2
19	B 2655 SRY 07.25	3 2655 SRY 07.18	12	0	1	3
20	1095 F 03.22	10G5 F Aea	4	0	0	6
21	B 2641 SIT 11.24	B 26L1 SIT	9	0	0	7
22	2099 PFY	2098 PFY	7	0	0	1
23	B 4675 TTY	-	0	0	0	10
24	B 1553 UOY	9 1553 U8Y	8	0	0	2
25	B 2655 KOP 12.24	3 2655 KEP	8	0	0	8
26	A 1893 YA	1 * Y^	1	1	0	5

Tabel 4.6. Hasil confussion matrix citra cropping YOLOv4 dengan proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
27	B 2849 KKD	2849 KKD	8	0	0	1
28	B 1732 KRY	1732 KRI	7	0	0	3
29	B 1628 JI 04.25	9 1628 JI	8	0	0	7
30	PA 61 KK 08.21	PA 6} KK	7	0	0	7

Berdasarkan Tabel 4.3 maka didapati masing-masing nilai *recall*, *akurasi*, dan F1 *score* sebagaimana Tabel 4.7 pada citra *cropping* YOLOv4 tanpa proses otsu's dan sebelum dikenai proses SRGAN. Dengan masing-masing hasil citra dapat dilihat pada Lampiran A, dengan urutan citra pada seluruh tabel hasil sama dengan urutan citra pada Lampiran A.

Tabel 4.7. Hasil perhitungan *confussion matrix* citra *cropping* tanpa proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
1	R 8585 CW	0585 CUI	56%	50%	0,67
2	H 311 CI	311	38%	38%	0,55
3	AB 1449 TN 02.25	AB 1449 TM 02.25	94%	94%	0,97
4	AB 1111 YE 08.27	AB VV YE 0,8<27	69%	65%	0,79
5	AB 1823 DF 02.20	AB 1823 DF 02.20	100%	100%	1,00
6	AB 1623 AY 01.26	AB 1623 AYI 01.26	100%	94%	0,97
7	AB1382LX 09.24	[B1382LX	50%	50%	0,67
8	B 1599 TYC 02.24	B 1599 TYC 02.22	93%	93%	0,96
9	AB 1832 MH 08.26	AB 1832 MH 00 < 26	93%	82%	0,90
10	N 1226 BJ	"1226" BJ	78%	64%	0,78
11	KH 1852 AL 02.22	IKH 1852 AL}	63%	56%	0,71
12	AB 1628 GK 10.21	AB 1628 GK	63%	63%	0,77
13	B 2532 PFN	8 2532 PFN	90%	90%	0,95
14	B 9147 BCT	B 9107 BCT	90%	90%	0,95
15	9212 M	19212	67%	57%	0,73
16	B 9357 KAS	AIN	0%	0%	0,00
17	B 8899 AAN 08.24	B 8899 AAN 08 24	94%	94%	0,97
18	BL 1614 C	BBL 7612	56%	50%	0,67
19	B 2655 SRY 07.25	3 2655 SRY DIAE	63%	63%	0,77
20	1095 F 03.22	1095	33%	33%	0,50
21	B 2641 SIT 11.24	B 2641 SIT	63%	63%	0,77
22	2099 PFY	2099 PFV	100%	100%	1,00
23	B 4675 TTY	-	0%	0%	0,00
24	B 1553 UOY	8 1553 UBY	80%	80%	0,89
25	B 2655 KOP 12.24	3 2655 KOP	60%	60%	0,75
26	A 1893 YA	1093	33%	33%	0,50
27	B 2849 KKD	'2849 KKD	80%	80%	0,89
28	B 1732 KRY	7732 KRL	60%	60%	0,75
29	B 1628 JI 04.25	B 1628	40%	40%	0,57
30	PA 61 KK 08.21	PA 6} KK	50%	50%	0,67
		Rata-rata	**65%**	**63%**	**0,73**

Untuk Tabel 4.5 didapati masing-masing nilai *recall*, *akurasi*, dan F1 *score* sebagaimana Tabel 4.8 pada citra *cropping* YOLOv4 dengan proses otsu's dan sebelum dikenai proses SRGAN.

Tabel 4.8. Hasil perhitungan *confussion matrix* citra *cropping* dengan proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
1	R 8585 CW	RA Q	11%	10%	0,18
2	H 311 CI	# 341 CI	75%	75%	0,86
3	AB 1449 TN 02.25	AB 1L49 T 02.25	88%	88%	0,93
4	AB 1111 YE 08.27	AB: M YE	38%	35%	0,52
5	AB 1823 DF 02.20	AB 1823 DF 0A'40	81%	81%	0,90
6	AB 1623 AY 01.26	AB 1623 AYI 01-38	81%	76%	0,87
7	AB1382LX 09.24	B1382LX	50%	50%	0,67
8	B 1599 TYC 02.24	B 1599 TYC 22	76%	76%	0,87
9	AB 1832 MH 08.26	AB 1832 HH	56%	56%	0,72
10	N 1226 BJ	"1226" 8J	67%	55%	0,71
11	KH 1852 AL 02.22	IH 1852 AL;	82%	75%	0,86
12	AB 1628 GK 10.21	AB 1628 GK	63%	63%	0,77
13	B 2532 PFN	B337AM	20%	20%	0,33
14	B 9147 BCT	8 %187 ECT	45%	45%	0,63
15	9212 M	9212	67%	67%	0,80
16	B 9357 KAS	9137 L|R	29%	29%	0,44
17	B 8899 AAN 08.24	B 8899 AAN 08 2 4	94%	89%	0,94
18	BL 1614 C	BL 7614 €	78%	78%	0,88
19	B 2655 SRY 07.25	3 2655 SRY 07.18	80%	81%	0,89
20	1095 F 03.22	10G5 F Aea	40%	40%	0,57
21	B 2641 SIT 11.24	B 26L1 SIT	56%	56%	0,72
22	2099 PFY	2098 PFY	88%	88%	0,93
23	B 4675 TTY	-	0%	0%	0,00
24	B 1553 UOY	9 1553 U8Y	80%	80%	0,89
25	B 2655 KOP 12.24	3 2655 KEP	50%	50%	0,67
26	A 1893 YA	1 * y^	17%	14%	0,25
27	B 2849 KKD	2849 KKD	89%	89%	0,94
28	B 1732 KRY	1732 KRI	70%	70%	0,82
29	B 1628 JI 04.25	9 1628 JI	53%	53%	0,70
30	PA 61 KK 08.21	PA 6} KK	50%	50%	0,67
		Rata-rata	**59%**	**58%**	**0,70**

Dilakukan evaluasi pula pada citra hasil *cropping* YOLOv4 yang telah dikenai proses SRGAN, dengan hasil *confussion matrix* untuk masing-masing citra pada Tabel 4.9 dan Tabel 4.10 dan masing masing citra dapat dilihat pada Lampiran A pada kolom Hasil Super-Resolution, dengan urutan citra pada seluruh tabel hasil sama dengan urutan citra pada Lampiran A.

.

Tabel 4.9. Hasil *confussion matrix* citra *cropping* setelah SRGAN tanpa proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
1	R 8585 CW	IR 8505 QM]	7	2	0	2
2	H 311 CI	J11	2	0	0	6
3	AB 1449 TN 02.25	AB 1LZ9 TN 02.25	13	0	1	2
4	AB 1111 YE 08.27	AR M YE 08'27	10	0	0	5
5	AB 1823 DF 02.20	AB1823 DF 02 0	13	1	0	2
6	AB 1623 AY 01.26	AB 1623 AY 01 .26	15	1	1	0
7	AB1382LX 09.24	AB1382LX '09 24	13	1	0	1
8	B 1599 TYC 02.24	B 599 TYC 02.22	14	0	1	2
9	AB 1832 MH 08.26	AB 1832 MH 4 26	13	0	0	3
10	N 1226 BJ	1226 8J	6	0	0	3
11	KH 1852 AL 02.22	KH 1852 ALI 03 72	13	1	0	3
12	AB 1628 GK 10.21	AB 1628 GK 10,71	14	0	0	2
13	B 2532 PFN	877802 PFM	5	0	0	6
14	B 9147 BCT	8 9147 6C1	7	0	0	3
15	9212 M	92122	4	1	0	2
16	B 9357 KAS	7 0797 !KAS	6	1	0	4
17	B 8899 AAN 08.24	B 8899 AAN 0 8 2L	13	1	0	2
18	BL 1614 C	BL 161	6	0	0	3
19	B 2655 SRY 07.25	J 2655 SRY DEE	9	3	0	3
20	1095 F 03.22	1095 F 03 F	9	0	0	3
21	B 2641 SIT 11.24	B 2621 SIT	10	0	0	6
22	2099 PFY	2099 PFYI	8	1	0	0
23	B 4675 TTY	7675 TTY	7	0	0	3
24	B 1553 UOY	D 1553 UBY	8	0	0	2
25	B 2655 KOP 12.24	3 2655 KOP	9	0	0	7
26	A 1893 YA	1893 MA	6	0	0	3
27	B 2849 KKD	8_2849 KKL	7	0	0	3
28	B 1732 KRY	3 1732 KRL	7	0	0	2
29	B 1628 JI 04.25	B 1628 02 7SE	8	2	0	5
30	PA 61 KK 08.21	PA 6} KK	7	0	0	7

Ditemukan pula hasil yang berbeda pada pengenalan karakter jika tidak melalui tahapan *otsu's thresholding* pada *pre-processing II* dapat dilihat pada Tabel 4.9 dan ketika pengenalan karakter terlebih dahulu melalui tahapan *otsu's thresholding* dengan hasil pada Tabel 4.10 menunjukan hasil yang berbeda pada karakter dikenali.

Tabel 4.10 Hasil *confussion matrix* citra *cropping* setelah SRGAN dengan proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
1	R 8585 CW	IP 8585 QHI	6	2	0	3
2	H 311 CI	H J11CI	6	0	0	2
3	AB 1449 TN 02.25	AB 1L49 TN 02'25	14	0	0	2
4	AB 1111 YE 08.27	AP I YE 08'27	10	0	0	5
5	AB 1823 DF 02.20	AB 1823 DF 02+20	15	0	0	1
6	AB 1623 AY 01.26	|AB 1623 AYT 0 ! + 26	13	5	0	2
7	AB1382LX 09.24	AB1382LX	8	0	0	6
8	B 1599 TYC 02.24	B 1599 TYC 02.22	14	0	1	1

Tabel 4.11 Hasil *confussion matrix* citra *cropping* setelah SRGAN dengan proses otsu's (lanjutan)

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
9	AB 1832 MH 08.26	AB 1832 MH 4	11	0	0	5
10	N 1226 BJ	'1226 ' BJ	6	2	0	2
11	KH 1852 AL 02.22	IKH 1852 AL] 02	13	2	0	0
12	AB 1628 GK 10.21	AB 1628 GK| '10,71	15	3	0	0
13	B 2532 PFN	8 7372 AM	1	0	0	9
14	B 9147 BCT	0 9107_ECT	8	0	0	3
15	9212 M	9212	4	0	0	2
16	B 9357 KAS	J 9747 KAS	7	0	0	3
17	B 8899 AAN 08.24	B8899 AAN 08 . 26	13	2	1	1
18	BL 1614 C	BL 767	4	0	0	4
19	B 2655 SRY 07.25	3 2655 SRY 77.76	11	0	1	4
20	1095 F 03.22	1095 F MV	7	2	0	3
21	B 2641 SIT 11.24	'B 2641 SIT 11.26	15	1	1	1
22	2099 PFY	2099 PFY	8	0	0	0
23	B 4675 TTY	-	0	0	0	10
24	B 1553 UOY	3 2655 KOP	2	0	0	8
25	B 2655 KOP 12.24	3 2655 KOP 17}	11	0	0	5
26	A 1893 YA	18J YN	4	0	0	6
27	B 2849 KKD	-	0	0	0	10
28	B 1732 KRY	9 1732 KRY	9	0	0	1
29	B 1628 JI 04.25	B 1628 JL 3 2 G	9	0	0	6
30	PA 61 KK 08.21	PA 6} KK	7	0	0	7

Berdasarkan Tabel 4.9 maka didapati masing-masing nilai *recall*, *akurasi*, dan F1 *score* sebagaimana Tabel 4.12 pada citra *cropping* YOLOv4 tanpa proses otsu's dan sesudah dikenai proses SRGAN. Dengan masing-masing hasil citra dapat dilihat pada Lampiran A, dengan urutan citra pada seluruh tabel hasil sama dengan urutan citra pada Lampiran A.

Tabel 4.12. Hasil perhitungan *confussion matrix* citra *cropping* tanpa proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
1	R 8585 CW	IR 8505 QM]	78%	64%	0,78
2	H 311 CI	J11	25%	25%	0,40
3	AB 1449 TN 02.25	AB 1LZ9 TN 02.25	87%	88%	0,93
4	AB 1111 YE 08.27	AR M YE 08'27	67%	67%	0,80
5	AB 1823 DF 02.20	AB1823 DF 02 0	87%	81%	0,90
6	AB 1623 AY 01.26	AB 1623 AY 01 .26	100%	94%	0,97
7	AB1382LX 09.24	AB1382LX '09 24	93%	87%	0,93
8	B 1599 TYC 02.24	B 599 TYC 02.22	88%	88%	0,93
9	AB 1832 MH 08.26	AB 1832 MH 4 26	81%	81%	0,90
10	N 1226 BJ	1226 8J	67%	67%	0,80
11	KH 1852 AL 02.22	KH 1852 ALI 03 72	81%	76%	0,87
12	AB 1628 GK 10.21	AB 1628 GK 10,71	88%	88%	0,93
13	B 2532 PFN	877802 PFM	45%	45%	0,63
14	B 9147 BCT	8 9147 6C1	70%	70%	0,82
15	9212 M	92122	67%	57%	0,73
16	B 9357 KAS	7 0797 !KAS	60%	55%	0,71

Tabel 4.13. Hasil perhitungan *confussion matrix* citra *cropping* tanpa proses otsu's (lanjutan)

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
17	B 8899 AAN 08.24	B 8899 AAN 0 8 2L	87%	81%	0,90
18	BL 1614 C	BL 161	67%	67%	0,80
19	B 2655 SRY 07.25	J 2655 SRY DEE	75%	60%	0,75
20	1095 F 03.22	1095 F 03 F	75%	75%	0,86
21	B 2641 SIT 11.24	B 2621 SIT	63%	63%	0,77
22	2099 PFY	2099 PFYI	100%	89%	0,94
23	B 4675 TTY	7675 TTY	70%	70%	0,82
24	B 1553 UOY	D 1553 UBY	80%	80%	0,89
25	B 2655 KOP 12.24	3 2655 KOP	56%	56%	0,72
26	A 1893 YA	1893 MA	67%	67%	0,80
27	B 2849 KKD	8_2849 KKL	70%	70%	0,82
28	B 1732 KRY	3 1732 KRL	78%	78%	0,88
29	B 1628 JI 04.25	B 1628 02 7SE	62%	53%	0,70
30	PA 61 KK 08.21	PA 6} KK	50%	50%	0,67
		Rata-rata	**73%**	**70%**	**0,81**

Untuk Tabel 4.9 didapati masing-masing nilai *recall*, *akurasi*, dan F1 *score* sebagaimana Tabel 4.14 pada citra *cropping* YOLOv4 dengan proses otsu's dan sesudah dikenai proses SRGAN. Dengan masing-masing hasil citra dapat dilihat pada Lampiran A, dengan urutan citra pada seluruh tabel hasil sama dengan urutan citra pada Lampiran A.

Tabel 4.14. Hasil perhitungan *confussion matrix* citra *cropping* tanpa proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
1	R 8585 CW	IP 8585 QHI	67%	55%	0,71
2	H 311 CI	H J11CI	75%	75%	0,86
3	AB 1449 TN 02.25	AB 1L49 TN 02'25	88%	88%	0,93
4	AB 1111 YE 08.27	AP I YE 08'27	67%	67%	0,80
5	AB 1823 DF 02.20	AB 1823 DF 02+20	94%	94%	0,97
6	AB 1623 AY 01.26	|AB 1623 AYT 0 ! + 26	87%	65%	0,79
7	AB1382LX 09.24	AB1382LX	57%	57%	0,73
8	B 1599 TYC 02.24	B 1599 TYC 02.22	93%	94%	0,97
9	AB 1832 MH 08.26	AB 1832 MH 4	69%	69%	0,81
10	N 1226 BJ	'1226 ' BJ	75%	60%	0,75
11	KH 1852 AL 02.22	IKH 1852 AL] 02	100%	87%	0,93
12	AB 1628 GK 10.21	AB 1628 GK| '10,71	100%	83%	0,91
13	B 2532 PFN	8 7372 AM	10%	10%	0,18
14	B 9147 BCT	0 9107_ECT	73%	73%	0,84
15	9212 M	9212	67%	67%	0,80
16	B 9357 KAS	J 9747 KAS	70%	70%	0,82
17	B 8899 AAN 08.24	B8899 AAN 08 . 26	93%	82%	0,90
18	BL 1614 C	BL 767	50%	50%	0,67
19	B 2655 SRY 07.25	3 2655 SRY 77.76	73%	75%	0,85
20	1095 F 03.22	1095 F MV	70%	58%	0,74
21	B 2641 SIT 11.24	'B 2641 SIT 11.26	94%	89%	0,94
22	2099 PFY	2099 PFY	100%	100%	1,00
23	B 4675 TTY	-	0%	0%	0,00
24	B 1553 UOY	3 2655 KOP	20%	20%	0,33

Tabel 4.15. Hasil perhitungan *confussion matrix* citra *cropping* tanpa proses otsu's (lanjutan)

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
25	B 2655 KOP 12.24	3 2655 KOP 17}	69%	69%	0,81
26	A 1893 YA	18J YN	40%	40%	0,57
27	B 2849 KKD	-	0%	0%	0,00
28	B 1732 KRY	9 1732 KRY	90%	90%	0,95
29	B 1628 JI 04.25	B 1628 JL 3 2 G	60%	60%	0,75
30	PA 61 KK 08.21	PA 6} KK	50%	50%	0,67
Rata-rata			**67%**	**63%**	**0,73**

Berdasarkan hasil data uji, yaitu hasil perhitungan nilai *recall*, akurasi, dan F1 *score* pada 30 citra uji hasil *cropping bounding-box* YOLOv4 pada Tabel 4.7 untuk citra sebelum dikenai SRGAN dan *otsu's thresholding*, Tabel 4.8 untuk citra sebelum dikenai SRGAN namun telah melalui proses *otsu's thresholding*. Dengan Tabel 4.12 adalah hasil perhitungan untuk citra telah dikenai SRGAN atau sebagai citra *super-resolution* namun tidak dikenai proses *otsu's thresholding*, dan pada Tabel 4.14 adalah citra *super-resolution* namun telah melalui proses *otsu's thresholding*. Dapat dilihat pada data uji citra hasil *cropping bounding-box* YOLOv4 terjadi peningkatan rata-rata untuk *recall*, akurasi, maupun F1 *score* dari setiap pasangan rancangan uji setelah dikenai proses SRGAN. Dapat dilihat pada Tabel 4.7 dibandingkan dengan Tabel 4.12 untuk citra sebelum dikenai proses *otsu's thresholding*, dan Tabel 4.8 dibandingkan dengan rata-rata hasil pada Tabel 4.14 untuk citra telah dikenai *otsu's thresholding*. Diketahui pula bahwa terjadi penurunan rata-rata hasil pada citra sebelum melalui proses *otsu's thresholding* dan citra setelah melalui proses *otsu's thresholding* yang dapat dilihat pada Tabel 4.7 dibandingkan dengan Tabel 4.8 untuk citra sebelum dikenai SRGAN, dan pada Tabel 4.12 dibandingkan dengan Tabel 4.14 untuk citra telah dikenai SRGAN.

Pada citra *low-resolution* sintetis hasil juga dilakukan evaluasi yang serupa, yaitu dilakukan perhitungan nilai *confussion matrix* sebagaimana Tabel 4.16 untuk citra sintetis sebelum dikenai proses SRGAN dan *otsu's thresholding*. Dengan masing-masing citra sintetis dapat dilihat pada LAMPIRAN B. Evaluasi citra sintetis, dengan urutan sebagaimana label dan sesuai pada seluruh tabel hasil.

Tabel 4.16. Hasil *confussion matrix* citra sintetis sebelum SRGAN tanpa proses *otsu's*

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
1	MARYLAND AC.32.93 l55l	AC 32.93	6	0	1	15
2	CORNHUSKER STATE II.F506 NEBRASKA 69	LLE506	3	0	0	33
3	NEVADA CCA222	CCAZZZ	3	0	0	10
4	AX.3838 NY WORLD'S FAIR 64	AX 3838	7	0	0	19
5	NORTH CAROLINA AD-49373 .MULTI YEAR.	AD-49373	7	0	1	28
6	NF J 077 JUL OREGON Oregon 8252526 88	NFJ 077	7	0	0	30
7	SG13993 SOUTH CAROLINA 64	[5G13993	6	1	0	19
8	TEXAS 57 KR.7195	KR*7195	6	0	0	10
9	UTAH 59 AH 2023	AH 2023	7	0	0	8
10	TEXAS 60 BJ.4432	BJ*4432	6	0	0	10

Tabel 4.17. Hasil *confussion matrix* citra sintetis sebelum SRGAN tanpa proses otsu's (lanjutan)

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
11	P 1396 HE 06.25	1396 HE	7	0	0	8
12	B 5347 BBQ 01 .26	B 5347 BBQ 01 .26	17	0	0	0
13	B 1174 BES 08.26	B 1174 BES	10	0	0	6
14	KT 5050 BNI 05.25	5050 D	4	0	0	13
15	AB 3174 KT 03.27	AB 3174 KI 03-27	9	0	0	7
16	E 5053 RG 09.20	5053_ 09*20	8	1	0	7
17	E 3193 PS 08.17	3193	4	0	0	11
18	E 4670 TL 05.22	4670	4	0	0	11
19	E 3916 P 05.21	40916	3	0	0	11
20	E 3063 RS 07.22	7063RS	5	0	0	10

Dengan hasil nilai *confussion matrix* citra *low-resolution* sintetis sebelum dikenai SRGAN dan telah dikenai proses *otsu's thresholding* dapat dilihat pada Tabel 4.18.

Tabel 4.18. Hasil *confussion matrix* citra sintetis sebelum SRGAN setelah proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
1	MARYLAND AC.32.93 l55l	LC 3393	4	0	0	18
2	CORNHUSKER STATE II.F506 NEBRASKA 69	7308	1	0	0	35
3	NEVADA CCA222	CCAZZZ	3	0	0	10
4	AX.3838 NY WORLD'S FAIR 64	0X3838	5	0	0	21
5	NORTH CAROLINA AD-49373 MULTI YEAR	R-1977	2	0	1	31
6	NF J 077 JUL OREGON Oregon 8252526 88	REJO7 QAFSOK	2	0	0	37
7	SG13993 SOUTH CAROLINA 64	15G13993|	6	1	0	19
8	TEXAS 57 KR.7195	KR'7195	6	0	0	10
9	UTAH 59 AH 2023	AH 2023	7	0	0	8
10	TEXAS 60 BJ.4432	0J*4432	5	0	0	11
11	P 1396 HE 06.25	1396 HE	7	0	0	8
12	B 5347 BBQ 01 .26	5347 BBU 01 +26	13	0	0	4
13	B 1174 BES 08.26	1194 BES	7	0	0	9
14	KT 5050 BNI 05.25	K7 5U50 BN] 62G	8	0	0	9
15	AB 3174 KT 03.27	7	1	0	0	15
16	E 5053 RG 09.20	5053 . RG 08*80	10	2	0	5
17	E 3193 PS 08.17	1193	2	0	0	12
18	E 4670 TL 05.22	4670	4	0	0	11
19	E 3916 P 05.21	7916 08011	4	0	0	10
20	E 3063 RS 07.22	J06?	2	0	0	13

Sehingga didapati masing-masing nilai *recall*, *akurasi*, dan F1 *score* sebagaimana Tabel 4.19 berdasarkan pada hasil *confussion matrix* Tabel 4.16 pada citra *low-resolution* sintetis sebelum SRGAN dan tanpa proses *otsu's thresholding*. Dan pada citra *low-resolution* sintetis sebelum SRGAN dan sesudah dikenai proses *otsu's thresholding* sebagaimana Tabel 4.20 berdasarkan hasil *confussion matrix* Tabel 4.18.

Tabel 4.19. Perhitungan *confussion matrix* citra sintetis sebelum SRGAN tanpa proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
1	MARYLAND AC.32.93 l55l	AC 32.93	29%	32%	0,44
2	CORNHUSKER STATE II.F506 NEBRASKA 69	LLE506	8%	8%	0,15
3	NEVADA CCA222	CCAZZZ	23%	23%	0,38
4	AX.3838 NY WORLD'S FAIR 64	AX 3838	27%	27%	0,42
5	NORTH CAROLINA AD-49373 .MULTI YEAR.	AD-49373	20%	22%	0,33
6	NF J 077 JUL OREGON Oregon 8252526 88	NFJ 077	19%	19%	0,32
7	SG13993 SOUTH CAROLINA 64	[5G13993	24%	23%	0,38
8	TEXAS 57 KR.7195	KR*7195	38%	38%	0,55
9	UTAH 59 AH 2023	AH 2023	47%	47%	0,64
10	TEXAS 60 BJ.4432	BJ*4432	38%	38%	0,55
11	P 1396 HE 06.25	1396 HE	47%	47%	0,64
12	B 5347 BBQ 01 .26	B 5347 BBQ 01 .26	100%	100%	1,00
13	B 1174 BES 08.26	B 1174 BES	63%	63%	0,77
14	KT 5050 BNI 05.25	5050 D	24%	24%	0,38
15	AB 3174 KT 03.27	AB 3174 KI 03-27	56%	56%	0,72
16	E 5053 RG 09.20	5053_ 09*20	53%	50%	0,67
17	E 3193 PS 08.17	3193	27%	27%	0,42
18	E 4670 TL 05.22	4670	27%	27%	0,42
19	E 3916 P 05.21	40916	21%	21%	0,35
20	E 3063 RS 07.22	7063RS	33%	33%	0,50
		Rata-rata	**36%**	**36%**	**0,50**

Tabel 4.20. Perhitungan *confussion matrix* citra sintetis sebelum SRGAN setelah proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
1	MARYLAND AC.32.93 l55l	LC 3393	18%	18%	0,31
2	CORNHUSKER STATE II.F506 NEBRASKA 69	7308	3%	3%	0,05
3	NEVADA CCA222	CCAZZZ	23%	23%	0,38
4	AX.3838 NY WORLD'S FAIR 64	0X3838	19%	19%	0,32
5	NORTH CAROLINA AD-49373 .MULTI YEAR.	R-1977	6%	9%	0,11
6	NF J 077 JUL OREGON Oregon 8252526 88	REJO7 QAFSOK	5%	5%	0,10
7	SG13993 SOUTH CAROLINA 64	15G13993|	24%	23%	0,38
8	TEXAS 57 KR.7195	KR'7195	38%	38%	0,55
9	UTAH 59 AH 2023	AH 2023	47%	47%	0,64
10	TEXAS 60 BJ.4432	0J*4432	31%	31%	0,48
11	P 1396 HE 06.25	1396 HE	47%	47%	0,64
12	B 5347 BBQ 01 .26	5347 BBU 01 +26	76%	76%	0,87
13	B 1174 BES 08.26	1194 BES	44%	44%	0,61

Tabel 4.21. Perhitungan *confussion matrix* citra sintetis sebelum SRGAN setelah proses otsu's (lanjutan)

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
14	KT 5050 BNI 05.25	K7 5U50 BN] 62G	47%	47%	0,64
15	AB 3174 KT 03.27	7	6%	6%	0,12
16	E 5053 RG 09.20	5053 . RG, 08*80	67%	59%	0,74
17	E 3193 PS 08.17	1193	14%	14%	0,25
18	E 4670 TL 05.22	4670	27%	27%	0,42
19	E 3916 P 05.21	7916 08011	29%	29%	0,44
20	E 3063 RS 07.22	J06?	13%	13%	0,24
		Rata-rata	**29%**	**29%**	**0,41**

Dengan hasil perhitungan nilai *confussion matrix* pada citra generasi *super-resolution* sebelum dikenai proses *otsu's thresholding* yaitu pada Tabel 4.22. Pada citra generasi *super-resolution* setelah dikenai proses *otsu's thresholding* yaitu pada Tabel 4.23.

Tabel 4.22. Hasil *confussion matrix* citra sintetis setelah SRGAN dan sebelum proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
1	MARYLAND AC.32.93 l55l	LILD AC 32 93	9	0	0	13
2	CORNHUSKER STATE II.F506 NEBRASKA 69	FF506 JONI	5	4	0	33
3	NEVADA CCA222	KEMDA CCA2ZZ	8	0	0	5
4	AX.3838 NY WORLD'S FAIR 64	AX 3838 NI_#ADE1E HNT	7	1	0	19
5	NORTH CAROLINA AD-49373 .MULTI YEAR.	RORTR CAROLNA AD-49373 ~UL'	23	0	0	13
6	NF J 077 JUL OREGON Oregon 8252526 88	NFJ 077/ OREGOM	12	1	0	25
7	SG13993 SOUTH CAROLINA 64	(613993| I47I COROL S] 5A	9	0	0	16
8	TEXAS 57 KR.7195	{TIXAG KR:7195	10	0	0	6
9	UTAH 59 AH 2023	WIMI AH 2023	7	0	0	8
10	TEXAS 60 BJ.4432	TC5 00 BJ*4432	10	0	0	6
11	P 1396 HE 06.25	P 1396 HE} 6'2	11	1	0	4
12	B 5347 BBQ 01 .26	B 5347 BBQ 01 .26	17	0	0	0
13	B 1174 BES 08.26	B 7174 BEES 0E 26	13	1	0	3
14	KT 5050 BNI 05.25	KT 5050 BHI 06.25	14	0	1	2
15	AB 3174 KT 03.27	AB 3176 K 03.27	13	0	1	2
16	E 5053 RG 09.20	IE 5053 IBO 09*201	12	2	0	3
17	E 3193 PS 08.17	13193 PS 00 $	9	0	0	6
18	E 4670 TL 05.22	E 4670 TL 05 23	13	0	0	2
19	E 3916 P 05.21	F 2916 PI 05.21	11	1	1	2
20	E 3063 RS 07.22	IE 3063 RST 07 '32	13	3	0	2

Tabel 4.23. Hasil *confussion matrix* citra sintetis sebelum SRGAN setelah proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
1	MARYLAND AC.32.93 l55l	IRC 2 93	5	0	0	17
2	CORNHUSKER STATE II.F506 NEBRASKA 69	WEATI 7506] URQWINA 0	8	0	0	28
3	NEVADA CCA222	NEVADA CCA2ZZ	11	0	0	2
4	AX.3838 NY WORLD'S FAIR 64	AX 3838 MI MVIM 5 FAEA H	10	0	0	16

Tabel 4.24. Hasil *confussion matrix* citra sintetis sebelum SRGAN setelah proses otsu's (lanjutan)

Citra Ke-	Karakter Asli	Karakter Dikenali	TP	FP	TN	FN
5	NORTH CAROLINA AD-49373 .MULTI YEAR.	NORTH CAROLNA AD-49373 ~LTE	24	0	0	12
6	NF J 077 JUL OREGON Oregon 8252526 88	NFJ 077/ QREQQMN_E	10	0	0	27
7	SG13993 SOUTH CAROLINA 64	SG139937 34701 CXROLAOY 6A	16	1	0	9
8	TEXAS 57 KR.7195	"TEXAE 47 KF:7195	12	1	0	4
9	UTAH 59 AH 2023	WI AH 2023	8	0	0	7
10	TEXAS 60 BJ.4432	TC5 00 HJ*4432	9	0	0	7
11	P 1396 HE 06.25	P 1396 HE] M %	9	1	0	6
12	B 5347 BBQ 01 .26	B 5347 BBQ 01 .26	17	0	0	0
13	B 1174 BES 08.26	B 1174 BES 68 26	14	0	0	2
14	KT 5050 BNI 05.25	KT 5050 B#I 0625	14	0	0	3
15	AB 3174 KT 03.27	AB 3174 03 F	10	0	0	6
16	E 5053 RG 09.20	JE5053 RG 201	10	2	0	5
17	E 3193 PS 08.17	3193 PS 00	9	0	0	6
18	E 4670 TL 05.22	E 4670 TL 05 .23	13	1	1	1
19	E 3916 P 05.21	9916 P 05'2	9	0	0	5
20	E 3063 RS 07.22	E 3063 RSI 07 22	14	1	0	1

Didapati hasil perhitungan nilai *recall*, *akurasi*, dan F1 *score* sebagaimana Tabel 4.25 berdasarkan pada hasil *confussion matrix* Tabel 4.22 pada citra *low-resolution* sintetis setelah dikenai SRGAN dan tanpa proses *otsu's thresholding*.

Tabel 4.25. Perhitungan *confussion matrix* citra sintetis setelah SRGAN tanpa proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
1	MARYLAND AC.32.93 l55l	LILD AC 32 93	41%	41%	0,58
2	CORNHUSKER STATE II.F506 NEBRASKA 69	FF506 JONI	13%	12%	0,21
3	NEVADA CCA222	KEMDA CCA2ZZ	62%	62%	0,76
4	AX.3838 NY WORLD'S FAIR_64	AX 3838 NI_#ADE1E HNT	27%	26%	0,41
5	NORTH CAROLINA AD-49373 .MULTI YEAR.	RORTR CAROLNA AD-49373 ~UL'	64%	64%	0,78
6	NF J 077 JUL OREGON Oregon 8252526 88	NFJ 077/ OREGOM	32%	32%	0,48
7	SG13993 SOUTH CAROLINA 64	(613993| I47I COROL S] 5A	36%	36%	0,53
8	TEXAS 57 KR.7195	{TIXAG KR:7195	63%	63%	0,77
9	UTAH 59 AH 2023	WIMI AH 2023	47%	47%	0,64
10	TEXAS 60 BJ.4432	TC5 00 BJ*4432	63%	63%	0,77
11	P 1396 HE 06.25	P 1396 HE} 6'2	73%	69%	0,81
12	B 5347 BBQ 01 .26	B 5347 BBQ 01 .26	100%	100%	1,00
13	B 1174 BES 08.26	B 7174 BEES 0E, 26	81%	76%	0,87
14	KT 5050 BNI 05.25	KT 5050 BHI 06.25	88%	88%	0,93
15	AB 3174 KT 03.27	AB 3176 K 03.27	87%	88%	0,93
16	E 5053 RG 09.20	IE 5053, IBO 09*201	80%	71%	0,83
17	E 3193 PS 08.17	13193 PS 00 $	60%	60%	0,75

Tabel 4.26. Perhitungan *confussion matrix* citra sintetis setelah SRGAN tanpa proses otsu's (lanjutan)

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
18	E 4670 TL 05.22	E 4670 TL 05 23	87%	87%	0,93
19	E 3916 P 05.21	F 2916 PI 05.21	85%	80%	0,88
20	E 3063 RS 07.22	IE 3063 RST 07 '32	87%	72%	0,84
		Rata-rata	**64%**	**62%**	**0,73**

Didapati pula pada citra *low-resolution* sintetis setelah dikenai SRGAN dan proses *otsu's thresholding* sebagaimana Tabel 4.27 berdasarkan hasil *confussion matrix* Tabel 4.23.

Tabel 4.27. Perhitungan *confussion matrix* citra sintetis setelah SRGAN setelah proses otsu's

Citra Ke-	Karakter Asli	Karakter Dikenali	*Recall* ↑	Akurasi ↑	F1 *Score* ↑
1	MARYLAND AC.32.93 l55l	IRC 2 93	23%	23%	0,37
2	CORNHUSKER STATE II.F506 NEBRASKA 69	WEATI 7506] URQWINA 0	22%	22%	0,36
3	NEVADA CCA222	NEVADA CCA2ZZ	85%	85%	0,92
4	AX.3838 NY WORLD'S FAIR_64	AX 3838 MI MVIM 5 FAEA H	38%	38%	0,56
5	NORTH CAROLINA AD-49373 .MULTI YEAR.	NORTH CAROLNA AD-49373 ~LTE	67%	67%	0,80
6	NF J 077 JUL OREGON Oregon 8252526 88	NFJ 077/ QREQQMN_E	27%	27%	0,43
7	SG13993 SOUTH CAROLINA 64	SG139937 34701 CXROLAOY 6A	64%	62%	0,76
8	TEXAS 57 KR.7195	"TEXAE 47 KF:7195	75%	71%	0,83
9	UTAH 59 AH 2023	WI AH 2023	53%	53%	0,70
10	TEXAS 60 BJ.4432	TC5 00 HJ*4432	56%	56%	0,72
11	P 1396 HE 06.25	P 1396 HE] M %	60%	56%	0,72
12	B 5347 BBQ 01 .26	B 5347 BBQ 01 .26	100%	100%	1,00
13	B 1174 BES 08.26	B 1174 BES 68 26	88%	88%	0,93
14	KT 5050 BNI 05.25	KT 5050 B#I 0625	82%	82%	0,90
15	AB 3174 KT 03.27	AB 3174 03 F	63%	63%	0,77
16	E 5053 RG 09.20	JE5053 RG 201	67%	59%	0,74
17	E 3193 PS 08.17	3193 PS 00	60%	60%	0,75
18	E 4670 TL 05.22	E 4670 TL 05 .23	93%	88%	0,93
19	E 3916 P 05.21	9916 P 05'2	64%	64%	0,78
20	E 3063 RS 07.22	E 3063 RSI 07 22	93%	88%	0,93
		Rata-rata	**64%**	**63%**	**0,74**

Dilanjutkan pada proses evaluasi nilai *Peak Signal to Noise Ratio* (PSNR) dan juga *Structural Similarity Index* (SSIM), dengan semakin besar nilai PSNR maupun SSIM maka semakin baik hasil dari citra sintetis generasi *super-resolution*. Dilakukan pula evaluasi terhadap ukuran file citra sintetis sebelum dan setelah dikenai proses SRGAN sebagaimana pada Tabel 4.28.

Tabel 4.28. Evaluasi ukuran file, PSNR, dan SSIM citra *super-resolution*

Citra Ke-	Ukuran File		PSNR↑	SSIM↑
	Citra *low-resolution*	Citra *super-resolution*		
1	1,45 KB	9,92 KB	16,33	0,62
2	1,39 KB	10,11 KB	17,48	0,83
3	1,45 KB	7,65 KB	20,16	0,82
4	1,52 KB	9,75 KB	16,26	0,64
5	1,60 KB	11,97 KB	11,96	0,86
6	1,44 KB	8,63 KB	16,68	0,70
7	1,53 KB	9,22 KB	17,41	0,87
8	1,41 KB	8,09 KB	17,78	0,89
9	1,38 KB	6,44 KB	20,15	0,76
10	1,43 KB	7,00 KB	20,18	0,86
11	1,29 KB	6,40 KB	22,52	0,94
12	1,33 KB	6,86 KB	21,48	0,93
13	1,34 KB	7,29 KB	21,90	0,91
14	1,33 KB	9,18 KB	18,14	0,84
15	1,43 KB	8,80 KB	17,88	0,87
16	1,40 KB	8,90 KB	18,14	0,84
17	1,47 KB	9,84 KB	17,00	0,77
18	1,32 KB	7,00 KB	20,87	0,87
19	1,27 KB	6,26 KB	19,52	0,84
20	1,30 KB	6,73 KB	19,80	0,90
Rata-rata			**18,58**	**0,83**

Berdasarkan hasil yang ada pada Tabel 4.28 didapati bahwa hasil evaluasi citra *low-resolution* sintetis mendapatkan rata-rata nilai PSNR sebesar 18,58 db. Dengan memiliki rata-rata hasil nilai SSIM sebesar 0,83. Diketahui pula hasil citra *super-resolution* rata-rata mengalami peningkatan ukuran file sebesar 4,8 kali dari ukuran awal.

Berdasarkan hasil data uji lanjutan, yaitu hasil perhitungan nilai *recall*, akurasi, dan F1 *score* pada 20 citra uji hasil hasil generasi data pasangan citra *low-resolution* dan *high-resolution* sintetis, menghasilkan sebagaimana Tabel 4.16 untuk citra sintetis sebelum dikenai SRGAN dan *otsu's thresholding*. Tabel 4.18 untuk citra sintetis sebelum dikenai SRGAN namun telah melalui proses *otsu's thresholding*. Dengan Tabel 4.22 adalah hasil perhitungan untuk citra sintetis telah dikenai SRGAN atau sebagai citra *super-resolution* namun tidak dikenai proses *otsu's thresholding*. Terakhir pada Tabel 4.23 adalah citra *super-resolution* namun telah melalui proses *otsu's thresholding*.

Dapat dilihat pada data uji citra sintetis mengalami peningkatan rata-rata untuk *recall*, akurasi, maupun F1 *score* dari setiap pasangan rancangan uji setelah dikenai proses SRGAN. Dengan Tabel 4.16 dibandingkan dengan Tabel 4.22 untuk citra sebelum dikenai proses *otsu's thresholding*, dan Tabel 4.18 dibandingkan dengan rata-rata hasil pada Tabel 4.22 untuk citra telah dikenai *otsu's thresholding*. Diketahui pula bahwa terjadi penurunan rata-rata hasil pada citra sebelum melalui proses *otsu's thresholding* dan citra setelah melalui proses *otsu's thresholding* yang dapat dilihat pada Tabel 4.16 dibandingkan dengan Tabel 4.18 untuk citra sebelum dikenai SRGAN. Sedangkan pada Tabel 4.22 dibandingkan dengan Tabel 4.23 untuk citra telah dikenai SRGAN didapati kemiripan hasil.

4.2.8. Pembahasan hasil evaluasi model

Pada penelitian ini dilakukan dengan beberapa kombinasi pengujian yaitu, tanpa dan dengan proses *pre-processing Otsu's Thresholding and Binzarization*, dan masing-masing dilakukan proses SRGAN dengan evaluasi sebelum dan setelah dikenai SRGAN. Pada lingkup pengujian pertama, yaitu menggunakan 20 input citra dengan keadaan sebenarnya, berisikan kendaraan roda empat dan bukan roda empat yang diproses dengan YOLOv4, dengan berhasil melakukan lokalisasi pada 30 plat nomor yang kemudian dilakukan ektraksi citra hasil *bounding-box* dengan proses berupa *cropping*.

Pada hasil ekstraksi citra plat nomor menggunakan YOLOv4, yang telah dikenai proses SRGAN pada Tabel 4.12 untuk citra sebelum proses *otsu's* dan Tabel 4.14 sesudah proses *otsu's*, didapati peningkatan akurasi pengenalan karakter pada setiap plat nomor dari pada citra yang tidak melalui SRGAN dengan hasil pada Tabel 4.7 untuk citra sebelum proses *otsu's* dan Tabel 4.8 sesudah proses *otsu's*. Didapati peningkatan akurasi pada proses *license plate recognition* rata-rata sebesar 12% untuk nilai *recall*, 10% pada akurasi, dan 8% pada nilai F1 *score*.

Pada lingkup pengujian kedua digunakan citra sintetis yaitu dilakukan *pre-processing resize* pada citra plat nomor saja, dengan salah satu hasil dapat dilihat pada Lampiran B, dengan citra paling kiri adalah hasil *resize* citra menjadi *low-resolution* berukuran 32x64 pixel, dengan citra ke dua dari kiri adalah citra *high-resolution* hasil *resize* yaitu berukuran 128x256, dengan hasil citra *super-resolution* pada citra ketiga dari kiri yang memiliki ukuran sama dengan citra *high-resolution* atau empat kali lebih besar dari pada citra *low-resolution*.

Digunakan citra sintetis sebanyak 20 citra dengan masing masing citra dapat dilihat pada Lampiran B dengan urutan label sebagaimana tabel hasil pengujian citra sintetis. Memiliki kombinasi yang serupa dengan pengujian pada citra keadaan sebenarnya, pada Tabel 4.19 untuk citra sebelum proses *otsu's* dan Tabel 4.20 sesudah proses *otsu's* untuk citra sebelum dikenai proses SRGAN. Didapati proses SRGAN memberikan peningkatan pada proses pengenalan karakter, dengan hasil dapat dilihat pada Tabel 4.25 untuk citra sebelum proses *otsu's* dan Tabel 4.27 sesudah proses *otsu's*, dengan rata-rata peningkatan sebesar 98% untuk nilai *recall*, 94% pada akurasi, dan 0,63 pada nilai F1 *score*. Dilakukan evaluasi nilai PSNR dan SSIM terhadap hasil evaluasi citra sintetis, dengan menghasilkan rata-rata nilai sesuai pada Tabel 4.28 yaitu memiliki rata-rata nilai PSNR sebesar 18,58 dB dan SSIM sebesar 0,83. Evaluasi nilai PSNR dan SSIM hanya dilakukan pada lingkup pengujian data sintetis dikarenakan dibutuhkan ukuran citra yang sama, yaitu pada citra *high-resolution* dalam hal ini digunakan untuk mensimulasikan citra asli harus memiliki ukuran yang sama dengan hasil citra *super-resolution*.

Pada setiap lingkup pengujian yaitu pada citra ekstraksi YOLOv4 dan citra sintetis ditemukan bahwa citra yang melalui proses *pre-processing Otsu's Thresholding and Binzarization* memiliki hasil pengenalan karakter yang lebih buruk dikarenakan kontras pencahayaan yang kurang pada citra *low-resoluion* mengakibatkan buruknya prediksi nilai *threshold* sehingga hasil *binarization* menjadi kurang baik.

Terlepas dari hasil yang buruk dikarenakan proses *pre-processing Otsu's Thresholding and Binzarization*, proses pengenalan karakter juga dipengaruhi oleh hasil tangkapan citra oleh CCTV analog, semakin jauh posisi CCTV dengan objek yang ditangkap maka hasil citra memiliki kualitas yang kurang baik, atau pixel tangkapan CCTV menjadi sangat kecil yang mengakibatkan rendahnya kontras pencahayaan. Posisi CCTV juga dapat berpengaruh pada sudut tangkapan, beberapa kondisi mengakibatkan beberapa plat nomor menjadi tertutup oleh kendaraan lain dikarenakan penempatan CCTV yang kurang optimal.

4.2.9. Hasil evaluasi sistem

Berdasarkan rancangan pengujian pada Tabel 3.12 didapati hasil pengujian sitem sebagaimana Tabel 4.29.

Tabel 4.29. Hasil pengujian sistem

No	Halaman	Parameter Uji	Hasil	Keterangan
1	Halaman Utama	Dapat menerima citra berformat jpg, jpeg.	Sukses	Berhasil melakukan pengungahan citra dan mengharakan pada halaman tunggu pemrosesan.
2	Halaman Hasil Pengenalan	Dapat memberikan keluaran citra yang diunggah, plat nomor ter-*cropping*, generasi citra *super-resolution*, hasil karakter dikenali, beserta ukuran file dari citra *super-resolution*.	Sukses	Berhasil menunjukan citra input, nilai akurasi, hasil citra *cropping* dan citra *super-resolution*, dan masing-masing karakter pada citra *cropping* dan citra *super-resolution*, beserta ukuran citra.
3	Halaman Riwayat Pengenalan	Dapat menampilkan daftar seluruh riwayat pengenalan plat nomor sebelumnya.	Sukses	Berhasil menunjukan riwayat pengenalan yang pernah dilakukan secara historis, beserta ringkasan hasil pengenalan.
4	Halaman Detail Riwayat Pengenalan	Dapat menampilkan detail dari salah satu riwayat pengenalan terpilih.	Sukses Parsial	Berhasil menunjukan detail dari riwayat terpilih dengan informasi identik dengan Halaman Hasil Pengenalan. Pada kondisi citra input tidak terdeteksi plat nomor sama sekali maka hanya diarahkan pada halaman kosong.

Berdasarkan hasil pengujian pada Tabel 4.29, hampir seluruh parameter uji yang ada dapat dilakukan pada sistem yang telah dibuat. Dengan hasil pada masing-masing halamannya dapat dilihat pada Gambar 4.1 untuk Halaman Utama, Gambar 4.2 untuk Halaman Hasil Pengenalan, dengan Halaman Riwayat Pengenalan pada Gambar 4.3, dan Halaman Detail Riwayat Pengenalan pada Gambar 4.4.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Proses pengenalan plat nomor kendaraan roda empat masih memiliki beberapa kelemahan, yaitu salah satunya citra yang berkualitas rendah dikarenakan kemampuan CCTV Analog dalam menangkap citra masih memiliki banyak noise. Hal tersebut mengakibatkan proses pengenalan plat nomor kendaraan mengalami penurunan akurasi.

Pada penelitian ini dilakukan proses peningkatan kualitas citra satu kali menggunakan *Super-Resolution Generative Adversarial Network* (SRGAN). Citra diproses setelah ditemukannya plat nomor atau setelah proses lokalisasi plat nomor menggunakan arsitektur YOLOv4 sehingga mendapatkan citra dengan plat nomor saja. Penggunaan proses peningkatan citra tersebut didapati menghasilkan peningkatan akurasi pada proses *license plate recognition* rata-rata sebesar 12% untuk nilai *recall*, 10% pada akurasi, dan 8% pada nilai F1 *score* pada citra hasil *cropping bounding-box* YOLOv4. Pada citra uji sintetis menghasilkan rata-rata peningkatan sebesar 98% untuk nilai *recall*, 94% pada akurasi, dan 0,63 pada nilai F1 *score*. Dengan memiliki rata-rata nilai PSNR sebesar 18,58 dB dan SSIM sebesar 0,83.

Berdasarkan evaluasi yang ada dalam berbagai jenis, diketahui tak hanya berdasarkan kondisi kualitas citra namun posisi dan jarak penempatan kamera terhadap kendaraan maupun plat nomor akan mempengaruhi hasil dari kualitas tangkapan citra plat nomor kendaraan. Ditemui pula bahwa penerapan *pre-processing otsu's thresholding* memiliki hasil yang lebih buruk daripada citra yang tidak melalui tahapan *otsu's thresholding*.

Dibandingan dengan penelitian sebelumnya dalam proses pengenalan plat nomor kendaraan atau *license plate recognition*, pada penelitian ini didapati dengan penerapan proses peningkatan citra sebelumnya akan dapat mengahasilkan akurasi yang lebih baik pada tangkapan citra CCTV analog yang memiliki kualitas rendah.

5.2. Saran

Terlepas dari keberhasilan penggunaan *Super-Resolution Generative Adversarial Network* untuk meningkatkan akurasi pada pengenalan plat nomor roda empat, penelitian ini masih dapat dikembangkan lebih baik lagi. Adapun saran yang dapat diberikan adalah,

1. Penggunaan SRGAN atau algoritma peningkatan kualitas citra lainnya dapat pula dilanjutkan untuk diterapkan sebelum proses deteksi atau lokalisasi plat nomor untuk menambah akurasi lokalisasi, maupun dapat dilakukan secara berulang untuk mendapatkan hasil yang lebih baik.
2. Proses *pre-processing yaitu Otsu's Tresholding and Binarization* diketahui dapat meningkatkan akurasi dalam proses pengenalan karakter atau *Optical Character Recognition* (OCR), namun pada beberapa kondisi yaitu pada rendahnya kontras pencahayaan, proses *Otsu's Tresholding* dapat menyebabkan ketidak akuratan dalam proses pengenalan, hal ini dapat menjadi bahan kajian lebih lanjut pada pemilihan *pre-processing* yang tepat untuk proses pengenalan karakter.
3. Penambahan dataset untuk dapat mendeteksi plat nomor kendaraan roda dua, roda tiga, maupun kendaraan besar lainnya seperti bus maupun truk dapat dilakukan untuk menambah kemampuan lokalisasi *license plate recognition*.

DAFTAR PUSTAKA

Adams, W.J.L. and Saaty, D.L., 2006. CTC end2end.

Agarwal, P., Chopra, K., Kashif, M. and Kumari, V., 2018. Implementing ALPR for detection of traffic violations: A step towards sustainability. *Procedia Computer Science*, [online] 132, pp.738–743. Available at: <https://doi.org/10.1016/j.procs.2018.05.085>.

Agustsson, E. and Timofte, R., 2017. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017-July, pp.1122–1131.

Angel, T.P., Aishwarya, M.J., Varghese, F. and George, L., 2017. Traffic violation detection system. 4(03), pp.92–95.

Anon 2022. *Badan Pusat Statistik*. [online] Available at: <https://www.bps.go.id/indicator/17/57/1/perkembangan-jumlah-kendaraan-bermotor-menurut-jenis.html> [Accessed 8 Mar. 2022].

Asif, M.R., Qi, C., Wang, T., Fareed, M.S. and Raza, S.A., 2019. License plate detection for multi-national vehicles: An illumination invariant approach in multi-lane environment. *Computers and Electrical Engineering*, [online] 78, pp.132–147. Available at: <https://doi.org/10.1016/j.compeleceng.2019.07.012>.

Baek, Y., Lee, B., Han, D., Yun, S. and Lee, H., 2019. Character Region Awareness for Text Detection. *CoRR*, [online] abs/1904.0, p.12. Available at: <https://doi.org/10.48550/arXiv.1904.01941>.

Bakurov, I., Buzzelli, M., Schettini, R., Castelli, M. and Vanneschi, L., 2022. Structural similarity index (SSIM) revisited: A data-driven approach. *Expert Systems with Applications*, 189(October).

Bashir, S.M.A., Wang, Y., Khan, M. and Niu, Y., 2021. A comprehensive review of deep learningbased single image super-resolution. *PeerJ Computer Science*, 7, pp.1–56.

Bhat, A.T., Anupama, Akshatha, Rao, M.S. and Pai, D.G., 2021. Traffic violation detection in India using genetic algorithm. *Global Transitions Proceedings*, [online] 2(2), pp.309–314. Available at: <https://doi.org/10.1016/j.gltp.2021.08.056>.

Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y.M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. [online] Available at: <http://arxiv.org/abs/2004.10934>.

Chu, M., Xie, Y., Mayer, J., Leal-Taixé, L. and Thuerey, N., 2020. Learning temporal coherence via self-supervision for GAN-based video generation. *ACM Transactions on Graphics*, 39(4).

Dong, C., Loy, C.C., He, K. and Tang, X., 2014. Image Super-Resolution Using Deep Convolutional Networks. [online] Available at: <http://arxiv.org/abs/1501.00092>.

Dong, C., Zhu, X., Deng, Y., Loy, C.C. and Qiao, Y., 2015. Boosting Optical Character Recognition: A Super-Resolution Approach. [online] pp.1–5. Available at: <http://arxiv.org/abs/1506.02211>.

Fawcett, T., 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), pp.861–874.

Feng, X., Jiang, Y., Yang, X., Du, M. and Li, X., 2019. Computer vision algorithms and

hardware implementations: A survey. *Integration*, [online] 69(August), pp.309–320. Available at: <https://doi.org/10.1016/j.vlsi.2019.07.005>.

Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative Adversarial Networks. *Communications of the ACM*, [online] 63(11), pp.139–144. Available at: <http://arxiv.org/abs/1406.2661>.

Gunawan, D., Rohimah, W. and Rahmat, R.F., 2019. Automatic Number Plate Recognition for Indonesian License Plate by Using K-Nearest Neighbor Algorithm. *IOP Conference Series: Materials Science and Engineering*, 648(1).

Gunawan, I.K., Bayupati, I.P.A., Wibawa, K.S., Sukarsa, I.M. and Kurniawan, L.A., 2021. Indonesian Plate Number Identification Using YOLACT and Mobilenetv2 in the Parking Management System. *JUITA: Jurnal Informatika*, [online] 9(1), p.69. Available at: <10.30595/juita.v9i1.9230>.

Hamdi, A., Chan, Y.K. and Koo, V.C., 2021. A New Image Enhancement and Super Resolution technique for license plate recognition. *Heliyon*, 7(11).

Han, J., Kamber, M. and Pei, J., 2012. Data Preprocessing. In: J.P. Jiawei Han, Micheline Kamber, ed. *Data Mining*, Third Edit. [online] Elsevier.pp.83–124. Available at: <https://linkinghub.elsevier.com/retrieve/pii/B9780123814791000034>.

He, K., Zhang, X., Ren, S. and Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter, pp.1026–1034.

Hochreiter, S. and Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Computation*, 9(8), pp.1735–1780.

Imaduddin, H., Anwar, M.K., Perdana, M.I., Sulistijono, I.A. and Risnumawan, A., 2019. Indonesian vehicle license plate number detection using deep convolutional neural network. *International Electronics Symposium on Knowledge Creation and Intelligent Computing, IES-KCIC 2018 - Proceedings*, (October), pp.158–163.

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*, 1, pp.448–456.

JaidedAI, 2021. *GitHub - JaidedAI/EasyOCR*. [online] Available at: <https://github.com/jaidedai/easyocr> [Accessed 4 May 2022].

Jay, F., Renou, J.-P., Voinnet, O. and Navarro, L., 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks Jun-Yan. *Proceedings of the IEEE international conference on computer vision*, [online] pp.183–202. Available at: <http://link.springer.com/10.1007/978-1-60327-005-2_13>.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding. *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, pp.675–678.

Kangune, K., Kulkarni, V. and Kosamkar, P., 2019. Grapes Ripeness Estimation using Convolutional Neural network and Support Vector Machine. *2019 Global Conference for Advancement in Technology, GCAT 2019*, pp.1–5.

Kumar, K., Mishra, R.K. and Nandan, D., 2020. Efficient Hardware of RGB to Gray

Conversion Realized on FPGA and ASIC. *Procedia Computer Science*, [online] 171(2019), pp.2008–2015. Available at: <https://doi.org/10.1016/j.procs.2020.04.215>.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z. and Shi, W., 2016. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. [online] Available at: <http://arxiv.org/abs/1609.04802>.

Lin, C.H. and Li, Y., 2019. A License Plate Recognition System for Severe Tilt Angles Using Mask R-CNN. *International Conference on Advanced Mechatronic Systems, ICAMechS*, 2019-Augus, pp.229–234.

Powers, D.M.W., 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. [online] (May). Available at: <http://arxiv.org/abs/2010.16061>.

Pressman, R.S., 2014. *Software Quality Engineering*. SEVENTH ed. [online] *Software Quality Engineering: A Practitioner's Approach*, Hoboken, NJ, USA: John Wiley & Sons, Inc. Available at: <http://doi.wiley.com/10.1002/9781118830208>.

Priyanka C. Dighe, S.K.G., 2014. Survey on Image Resizing Techniques. *International Journal of Science and Research (IJSR)*, [online] 3(12), pp.1444–1448. Available at: <https://www.ijsr.net/archive/v3i12/U1VCMTQ3MjI=.pdf>.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem, pp.779–788.

Redmon, J. and Farhadi, A., 2018. YOLOv3: An Incremental Improvement. [online] Available at: <http://arxiv.org/abs/1804.02767>.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A. and Chen, X., 2016. Improved techniques for training GANs. *Advances in Neural Information Processing Systems*, pp.2234–2242.

Sanjeevi, M., 2019. *Ch:14 Generative Adversarial Networks (GAN's) with Math*. [online] Available at: <https://medium.com/deep-math-machine-learning-ai/ch-14-general-adversarial-networks-gans-with-math-1318faf46b43> [Accessed 25 Feb. 2022].

Santurkar, S., Tsipras, D., Ilyas, A. and Madry, A., 2018. How does batch normalization help optimization? *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS), pp.2483–2493.

Sham, A.S.D., Pandey, P., Jain, S. and Kalaivani, S., 2021. Automatic License Plate Recognition Using Yolov4 and Tesseract Ocr. *International Journal of Electrical Engineering and Technology*, 12(5), pp.58–67.

Shi, B., Bai, X. and Yao, C., 2017. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11), pp.2298–2304.

Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D. and Wang, Z., 2016. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem, pp.1874–

1883.

Simonyan, K. and Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp.1–14.

Smith, R. and Podobny, Z., 2021. *Tesseract Open Source OCR Engine*. [online] Available at: <https://github.com/tesseract-ocr/tesseract> [Accessed 3 Mar. 2022].

Stehman, S. V., 1997. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1), pp.77–89.

Symolon, W. and Dagli, C., 2021. Single-Image Super Resolution Using Convolutional Neural Network. In: *Procedia Computer Science*. Elsevier B.V.pp.213–222.

Venkata Sudhakar, M., Anoora Reddy, A.V., Mounika, K., Sai Kumar, M.V. and Bharani, T., 2021. Development of smart parking management system. *Materials Today: Proceedings*, [online] (xxxx). Available at: <https://doi.org/10.1016/j.matpr.2021.07.040>.

Vittorio, A., 2018. *Toolkit to download and visualize single or multiple classes from the huge Open Images v4 dataset*. *GitHub repository*. Available at: <https://github.com/EscVM/OIDv4_ToolKit>.

Wang, C.Y., Mark Liao, H.Y., Wu, Y.H., Chen, P.Y., Hsieh, J.W. and Yeh, I.H., 2020a. CSPNet: A new backbone that can enhance learning capability of CNN. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2020-June, pp.1571–1580.

Wang, W., Xie, E., Liu, X., Wang, W., Liang, D., Shen, C. and Bai, X., 2020b. Scene Text Image Super-Resolution in the Wild. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12355 LNCS, pp.650–666.

Xu, B., Wang, N., Chen, T. and Li, M., 2015. Empirical Evaluation of Rectified Activations in Convolutional Network. [online] Available at: <http://arxiv.org/abs/1505.00853>.

Yang, W., Zhang, X., Tian, Y., Wang, W., Xue, J.H. and Liao, Q., 2019. *Deep Learning for Single Image Super-Resolution: A Brief Review*. *IEEE Transactions on Multimedia*, .

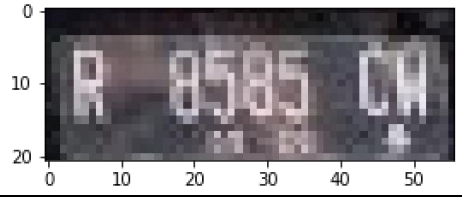
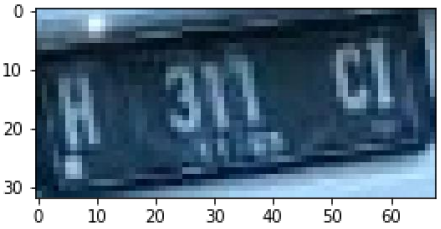
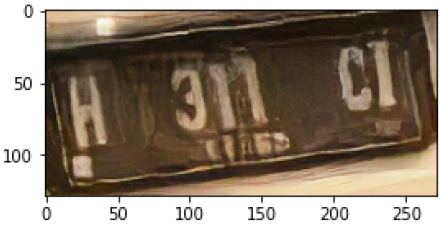
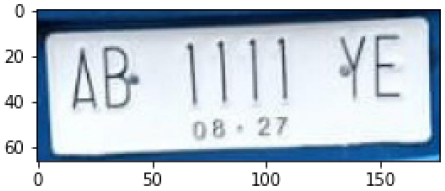
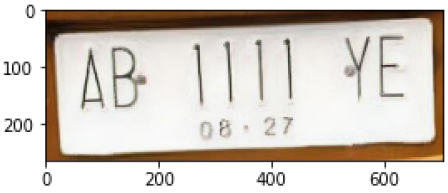
Zakria, Z., Deng, J., Kumar, R., Khokhar, M.S., Cai, J. and Kumar, J., 2022. Multiscale and Direction Target Detecting in Remote Sensing Images via Modified YOLO-v4. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15, pp.1039–1048.

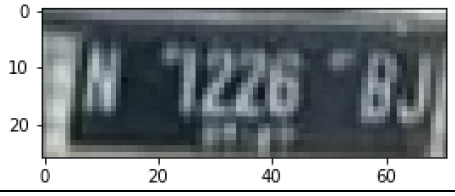
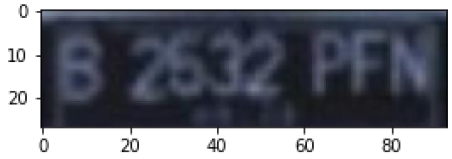
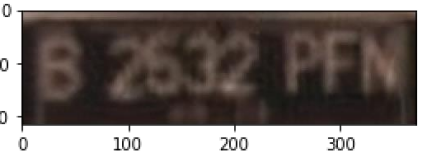
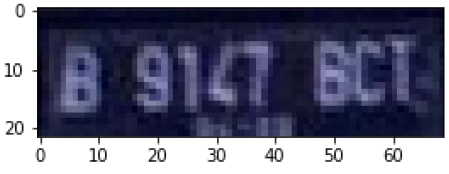
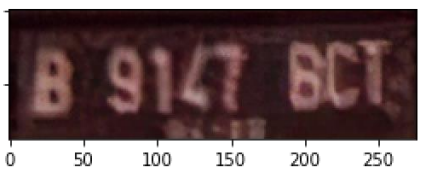
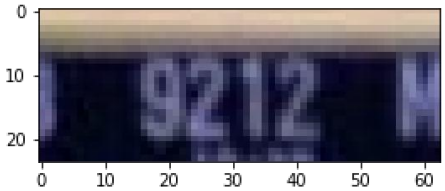
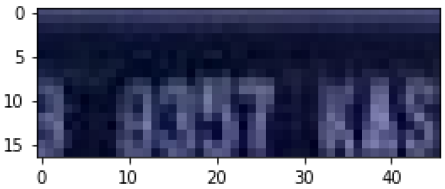
Zhang, C., Wang, Q. and Li, X., 2021. V-LPDR: Towards a unified framework for license plate detection, tracking, and recognition in real-world traffic videos. *Neurocomputing*, [online] 449, pp.189–206. Available at: <https://doi.org/10.1016/j.neucom.2021.03.103>.

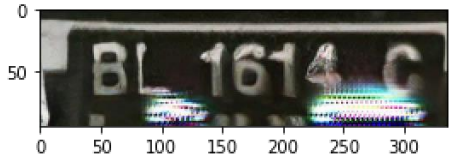
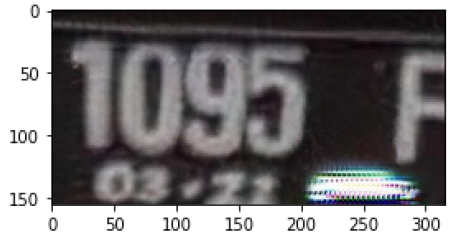
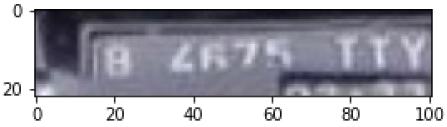
Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R. and Ren, D., 2020. Distance-IoU loss: Faster and better learning for bounding box regression. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, (2), pp.12993–13000.

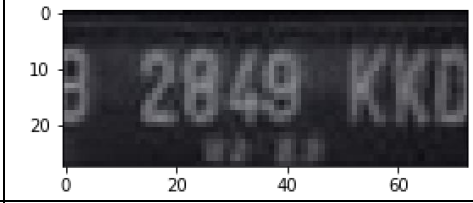
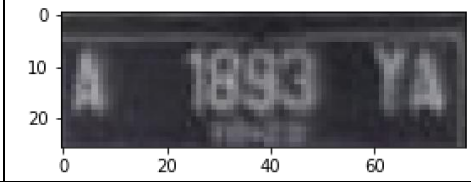
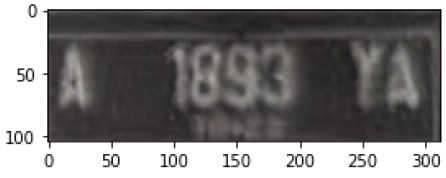
LAMPIRAN

LAMPIRAN A. Hasil citra *cropping* dan *super-resolution* YOLOv4

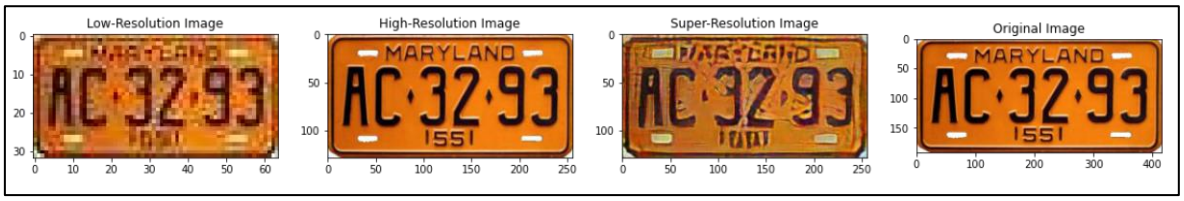
Citra Ke-	Hasil *Cropping* Plat Nomor	Hasil *Super-Resolution*
1		
2		
3		
4		
5		
6		
7		

Citra Ke-	Hasil *Cropping* Plat Nomor	Hasil *Super-Resolution*
8		
9		
10		
11		
12		
13		
14		
15		
16		

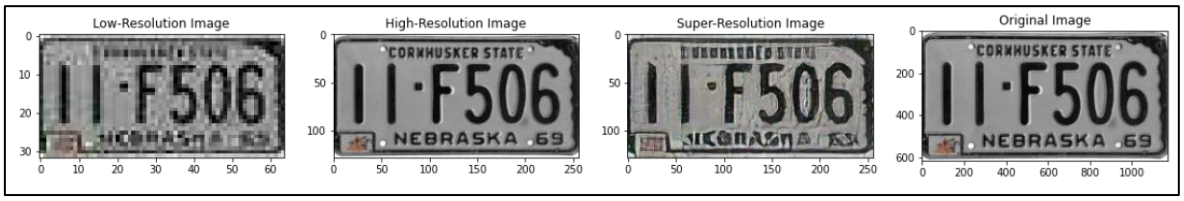
Citra Ke-	Hasil *Cropping* Plat Nomor	Hasil *Super-Resolution*
17		
18		
19		
20		
21		
22		
23		
24		

Citra Ke-	Hasil *Cropping* Plat Nomor	Hasil *Super-Resolution*
25		
26		
27		
28		
29		
30		

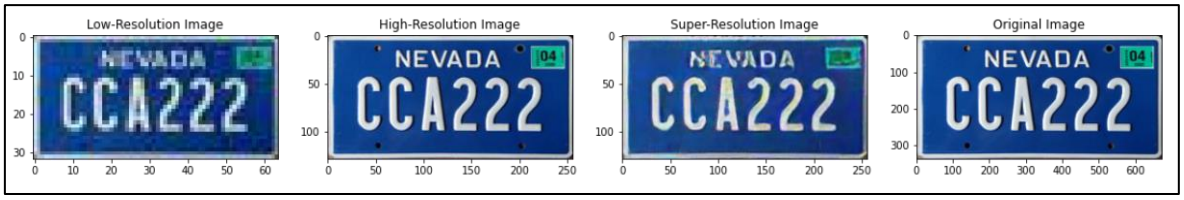
LAMPIRAN B. Evaluasi citra sintetis



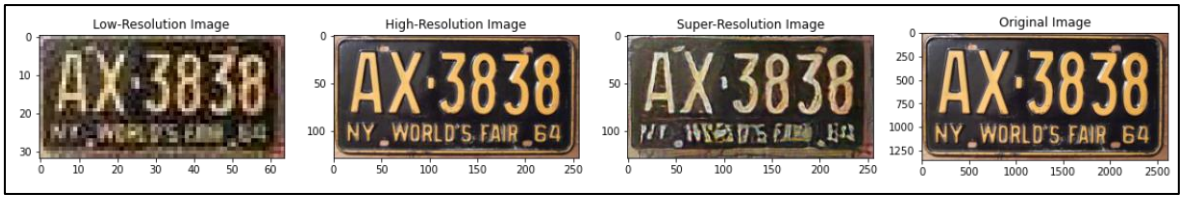
Lampiran B.1. Evaluasi citra sintetis ke-1



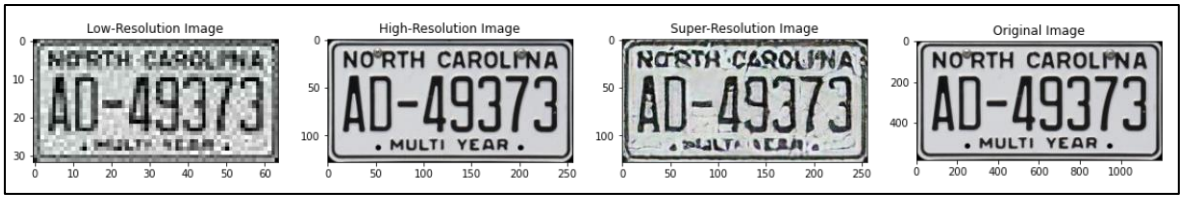
Lampiran B.2. Evaluasi citra sintetis ke-2



Lampiran B.3. Evaluasi citra sintetis ke-3

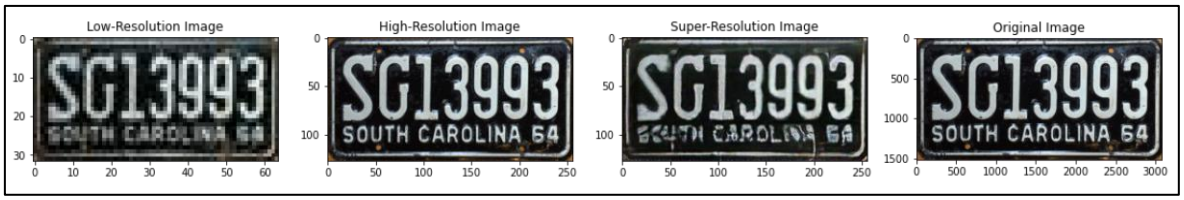


Lampiran B.4. Evaluasi citra sintetis ke-4

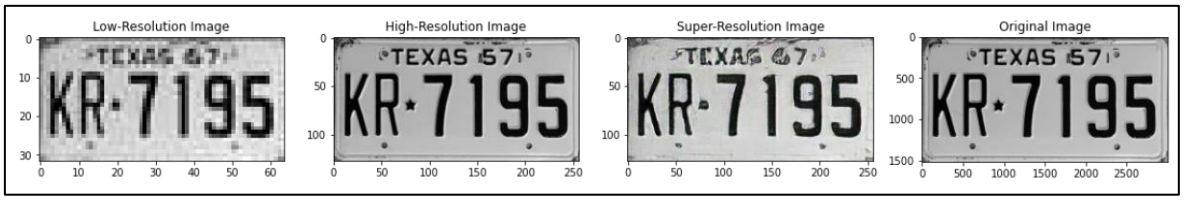


Lampiran B.5. Evaluasi citra sintetis ke-5

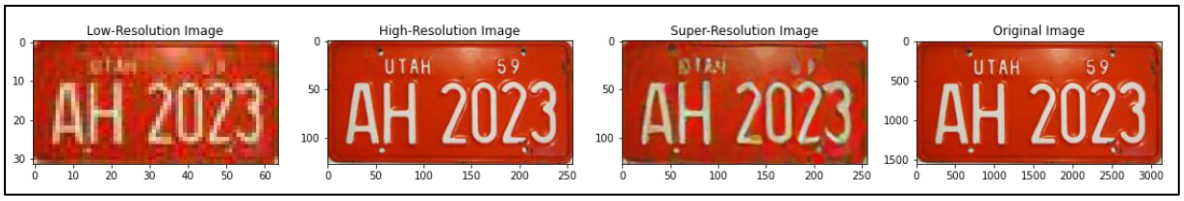
Lampiran B.6. Evaluasi citra sintetis ke-6



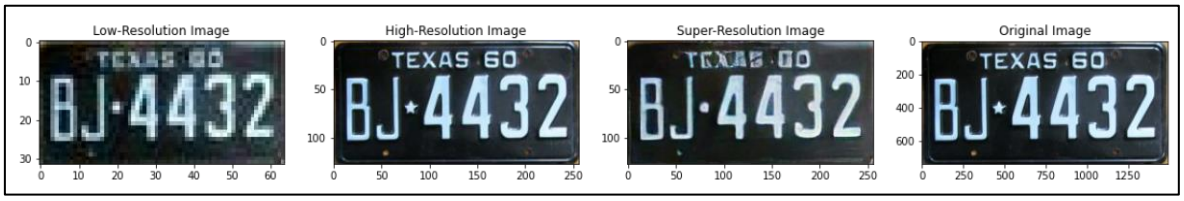
Lampiran B.7. Evaluasi citra sintetis ke-7



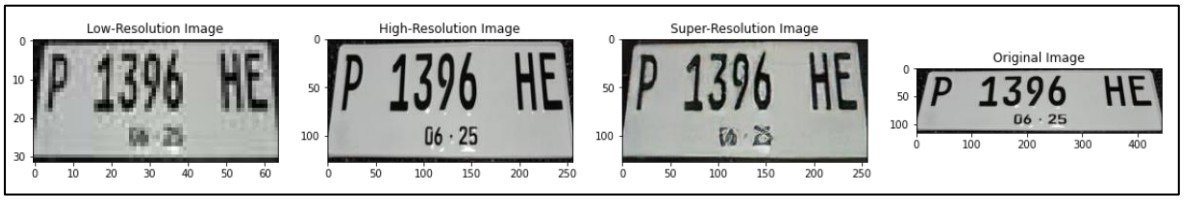
Lampiran B.8. Evaluasi citra sintetis ke-8



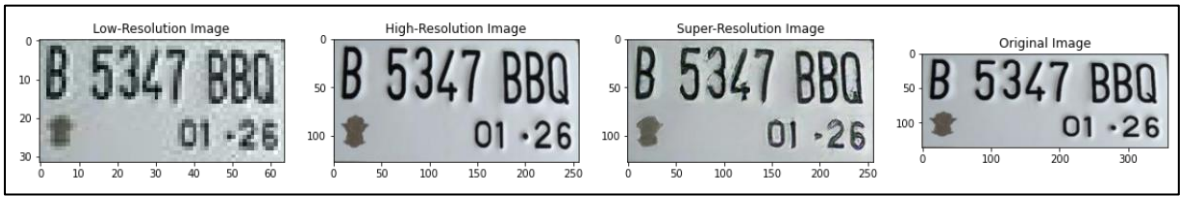
Lampiran B.9. Evaluasi citra sintetis ke-9



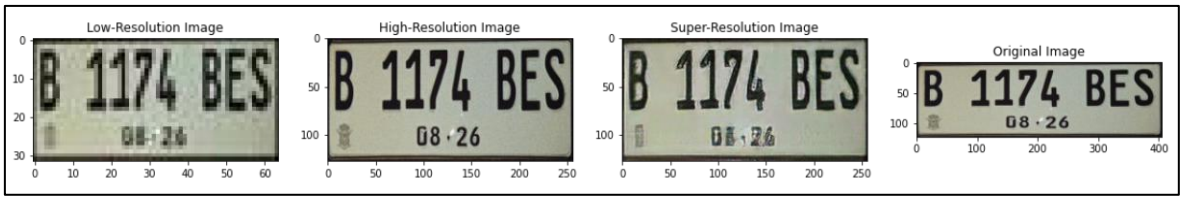
Lampiran B.10. Evaluasi citra sintetis ke-10



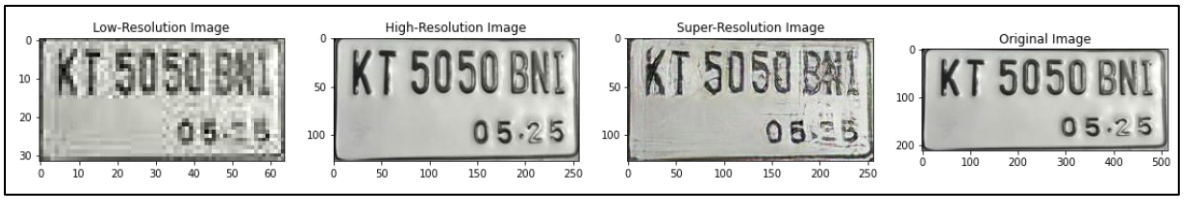
Lampiran B.11. Evaluasi citra sintetis ke-11



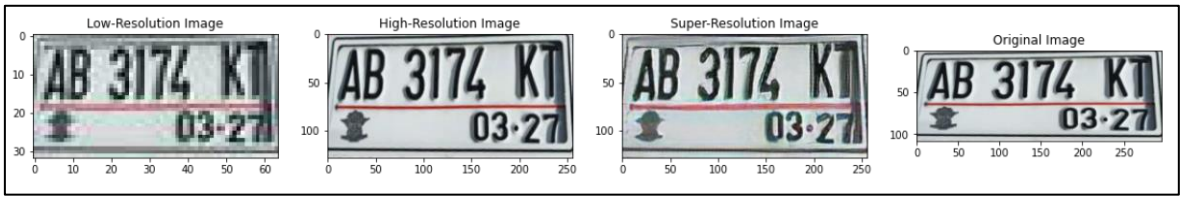
Lampiran B.12. Evaluasi citra sintetis ke-12



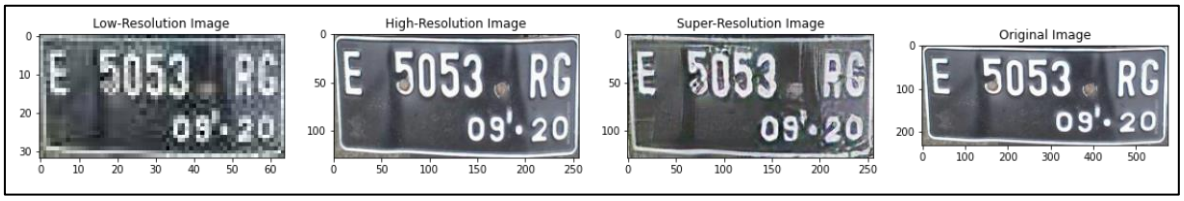
Lampiran B.13. Evaluasi citra sintetis ke-13



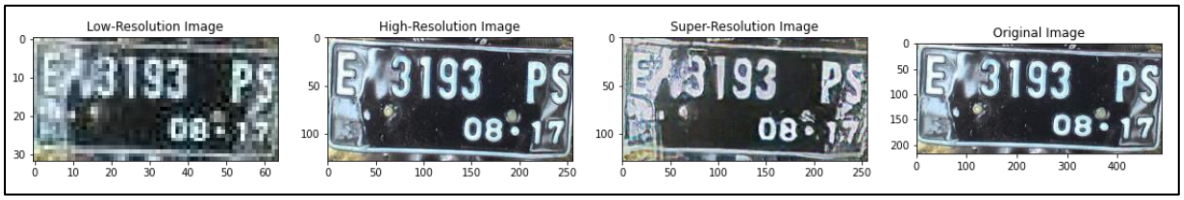
Lampiran B.14. Evaluasi citra sintetis ke-14



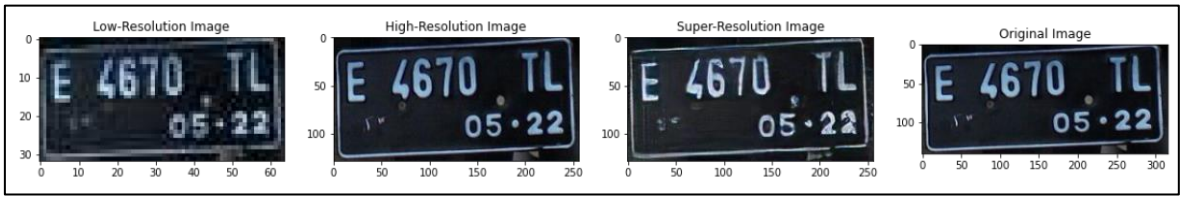
Lampiran B.15. Evaluasi citra sintetis ke-15



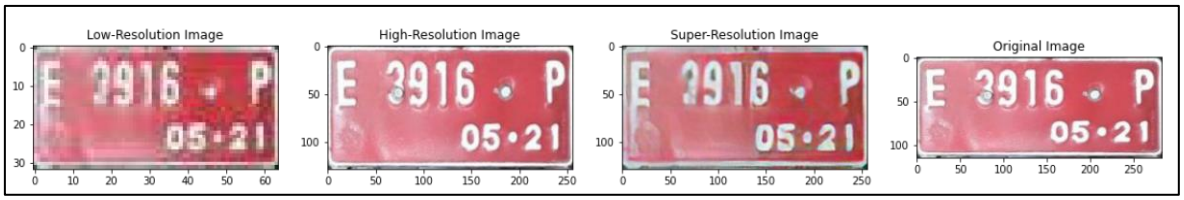
Lampiran B.16. Evaluasi citra sintetis ke-16



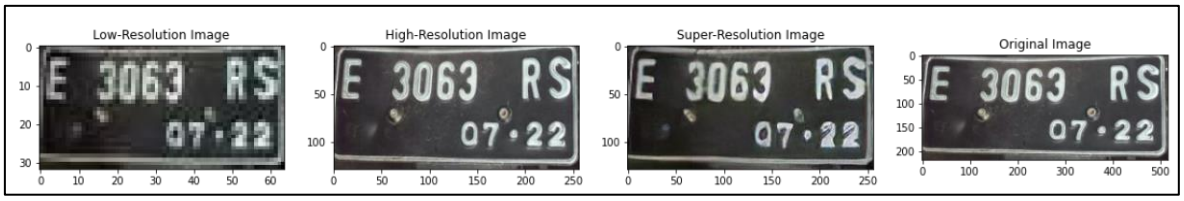
Lampiran B.17. Evaluasi citra sintetis ke-17



Lampiran B.18. Evaluasi citra sintetis ke-18



Lampiran B.19. Evaluasi citra sintetis ke-19



Lampiran B.20. Evaluasi citra sintetis ke-20