TUGAS AKHIR – IF184802

# PENGENALAN NOMOR POLISI KENDARAAN PADA DATA VIDEO MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK*

**PRADIPTA BASKARA**
**NRP 05111540000055**

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

**TUGAS AKHIR – IF184802**

# *PENGENALAN NOMOR POLISI KENDARAAN PADA DATA VIDEO MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK*

**PRADIPTA BASKARA**
**NRP 05111540000055**

**Dosen Pembimbing I**
**Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**

**Dosen Pembimbing II**
**Dini Adni Navastara, S.Kom., M.Sc.**

**Departemen Informatika**
**Fakultas Teknologi Informasi dan Komunikasi**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2019**

*(Halaman ini sengaja dikosongkan)*


**UNDERGRADUATE THESIS – IF184802**

# VIDEO BASED LICENSE PLATE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

**PRADIPTA BASKARA**
**NRP 05111540000055**

**First Advisor**
**Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**

**Second Advisor**
**Dini Adni Navastara, S.Kom., M.Sc.**

**Department of Informatics**
**Faculty of Information and Communication Technology**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2019**

*(Halaman ini sengaja dikosongkan)*

# LEMBAR PENGESAHAN

## PENGENALAN NOMOR POLISI KENDARAAN PADA DATA VIDEO MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK*

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Teknik Informatika
Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
**PRADIPTA BASKARA**
**NRP: 05111540000055**

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. .....
   (NIP. 19751220 200112 2 001) (Pembimbing 1)

2. Dini Adni Navastara, S.Kom., M.Sc. .....
   (NIP. 19851017 201504 2 001) (Pembimbing 2)

**SURABAYA**
**Juli, 2019**

*(Halaman ini sengaja dikosongkan)*

# PENGENALAN NOMOR POLISI KENDARAAN PADA DATA VIDEO MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK*


| | |
|---|---|
| **Nama Mahasiswa** | **: Pradipta Baskara** |
| **NRP** | **: 05111540000055** |
| **Jurusan** | **: Informatika, FTIK-ITS** |
| **Dosen Pembimbing 1** | **: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.** |
| **Dosen Pembimbing 2** | **: Dini Adni Navastara, S.Kom., M.Sc.** |


## ABSTRAK


*Machine learning telah menjadi bagian dari kehidupan sehari-hari bagi banyak orang. Salah satu pengaplikasian machine learning adalah pengenalan nomor polisi kendaraan. Pengenalan nomor polisi kendaraan mengkategorikan citra karakter-karakter yang ada. Banyak perusahaan, badan riset dan universitas yang terus mengembangkan machine learning agar mendapat hasil yang lebih akurat dan cepat. Dari situlah lahir algoritma deep learning, yang merupakan bagian dari machine learning. Convolutional Neural Network (CNN) adalah salah satu deep neural network yang cocok digunakan untuk mengolah data yang berbentuk 2 dimensi, seperti gambar dan video.*

*Pada tugas akhir ini, penulis mengusulkan penggunaan CNN untuk melakukan pengenalan karakter plat nomor pada data video. Tujuannya adalah untuk membangun sistem yang dapat mengenali karakter pada citra plat nomor. Data pelatihan dan uji merupakan data yang diambil secara mandiri oleh penulis. Data latih merupakan karakter-karakter dari citra plat nomor yang telah dilakukan proses cropping. Praproses terhadap data antara lain dilakukan perubahan kanal citra menjadi grayscale, padding, perubahan resolusi gambar menjadi 32x32 piksel, dilakukan proses morfologi, dan dilakukan proses augmentasi data berupa rotasi dan perbesaran ukuran gambar. Hasil uji coba optimal didapatkan dari arsitektur CNN dengan optimizer Adam, ukuran*

*kernel 4x4 pada Convolution Layer ketiga, dan lokalisasi YOLO dengan nilai akurasi 89,53%.*

**Kata kunci: *Convolutional Neural Network*, Data Citra, Pengenalan nomor polisi kendaraan**.

***VIDEO BASED LICENSE PLATE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK***

**Student's Name : Pradipta Baskara**
**Student's ID : 05111540000055**
**Department : Informatics, Faculty of ICT-ITS**
**First Advisor : Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**
**Second Advisor : Dini Adni Navastara, S.Kom., M.Sc.**

## ABSTRACT

*Machine learning has become a part of the daily life of people around the world. One of the application of machine learning is automatic plate number recognition. Automatic plate number recognition categorize characters found on the plate itself. Many companies, researchers and universities keep improving the machine learning to get a better and faster result. And from those improvements, deep learning algorithm is born. Convolutional Neural Network (CNN) is one of the deep neural network that suitable to process 2 dimentional data like image and video.*

*In this undergraduate thesis, author suggest the use of CNN to classify characters from the plate number. The purpose of this undergraduate thesis is to build a system that can regognize characters on plate number. The datasets itself are taken by the author from author's surrounding neighborhood vehicles while the data test also taken by the author surrounding neighborhood vehicles. The data train are characters, cropped from plate number that taken by the author. Some preprocessing include are image grayscaling, image padding, resizing to 32x32 size, morphological operation, and data augmentation to increase the variety of the dataset. Final optimal result got from the testing from CNN architecture with Adam optimizer, kernel size 4x4 on the third Convolution Layer and YOLO localization are 89,53% for testing accuracy.*

**Keywords:** ***Convolutional Neural Network, Automatic License Plate Recognition, Image Data..***

# KATA PENGANTAR

Puji syukur saya sampaikan kepada Tuhan yang Maha Esa karena berkat rahmat-Nya saya dapat melaksanakan Tugas Akhir yang berjudul:

"**PENGENALAN NOMOR POLISI KENDARAAN PADA DATA VIDEO MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK***"

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan banyak pihak, oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan kepada:

1. Kedua orangtua penulis, yang telah memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. dan Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Dr. Eng. Radityo Anggoro, S.Kom., M.Kom. selaku Ketua Program Studi S1 Departemen Informatika ITS dan seluruh dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Informatika ITS.
4. Admin-admin Laboratorium Komputasi Cerdas & Visi (KCV) yang memberikan kesempatan penulis untuk fokus mengerjakan Tugas Akhir ini dan menyediakan tempat di laboratorium tersebut.
5. Achmad Ibnu Malik Al Chasni, Cynthia Dewi Tejakusuma, Hendry Wiranto, serta Aaron Setiawan yang

telah menemani dan membantu penulis selama perkuliahan semester 8 dan pengerjaan Tugas Akhir ini.

6. Vicky Mahfudi, Reza Ar Razi, Achmad Raca yang selalu membantu dan memberi dukungan moral selama pengerjaan Tugas Akhir ini.
7. Nuzul Ristyantika, Rahandi Noor Pasha, dan Dandy Naufaldi yang telah membantu penulis selama proses pengerjaan Tugas Akhir ini.
8. Rezky Alamsyah, Akram Abdullah, Ronald Sumbayak, Rogo Jagad Alit, Tegar Satrio, dan Firman Aqil sebagai teman-teman *shift* malam yang telah membantu, menemani, dan menghibur penulis.
9. Yolanda Samosir, Arij Nafi'atul Mashuda, Ana Alimatus Zaqiyah, serta teman-teman *user* TA KCV 2019 yang juga telah membantu penulis dalam mengerjakan tugas akhir ini.
10. Seluruh mahasiswa Informatika ITS angkatan 2015 yang telah menjadi teman penulis selama menjalani masa kuliah di Informatika ITS.
11. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa laporan Tugas Akhir ini masih memiliki banyak kekurangan. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis kedepannya. Selain itu, penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum.

Surabaya, Januari 2019

# DAFTAR ISI

*(Halaman ini sengaja dikosongkan)*

# DAFTAR TABEL

*(Halaman ini sengaja dikosongkan)*

# DAFTAR KODE SUMBER

*(Halaman ini sengaja dikosongkan)*

# DAFTAR GAMBAR

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

*Machine learning* telah menjadi bagian dari kehidupan sehari-hari bagi banyak orang di seluruh dunia. Penemuan dan implementasi *machine learning* memungkinkan komputer untuk belajar dan melakukan prediksi pola yang mungkin terjadi dan dapat digunakan untuk membantu manusia melakukan kegiatan sehari-hari. Teknologi ini di zaman modern memungkinkan penyelesaian masalah lama dengan cara yang baru dan efisien. Beberapa pengaplikasian *machine learning* meliputi *fraud detection, image classification, information retrieval* dan *medical diagnosis* [1].

Salah satu pengaplikasian *machine learning* yang populer adalah *image classification*. *Image classification* mengkategorikan piksel-piksel di dalam suatu gambar menjadi satu dari banyak kelas gambar berdasarkan fitur yang berhasil diekstrak dari gambar tersebut [1]. Banyak bidang menggunakan *image classification* untuk meningkatkan kualitas produk, seperti bidang bisnis, finansial, kesehatan, riset, teknologi dan lain-lain. Seiring dengan berkembangnya teknologi, banyak perusahaan, badan riset dan universitas yang terus mengembangkan *machine learning* agar mendapat hasil yang lebih akurat, efisien dan cepat. Dari situlah lahir algoritma *deep learning*, yang merupakan bagian dari *machine learning*.

*Convolutional Neural Network* (CNN) adalah salah satu *deep neural network* yang cocok digunakan untuk mengolah data yang berbentuk 2 dimensi, seperti gambar dan video. Normalnya, CNN diterapkan untuk mengklasifikasikan gambar yang belum diolah (*raw image*). Di dalam tugas akhir ini, penulis mengusulkan penggunaan CNN untuk melakukan pengenalan nomor polisi kendaraan. Data pelatihan merupakan kumpulan citra karakter yang telah dikumpulkan, yang berasal dari citra-citra kendaraan yang telah diambil penulis dari beberapa lokasi. Citra-citra ini

kemudian dipotong secara mandiri untuk memperoleh karakter-karakter yang akan menjadi data pelatihan untuk CNN. Sedangkan data uji coba merupakan video kendaraan yang juga diambil secara mandiri.

## 1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana praproses pada data *frame video* kendaraan untuk deteksi area plat*?*
2. Bagaimana cara melakukan *frame tracking* pada data video?
3. Bagaimana cara segmentasi setiap karakter dari area plat yang sudah terdeteksi?
4. Bagaimana mengimplementasikan *Convolutional Neural Network* dalam mengenali karakter-karakter pada plat nomor polisi kendaran?
5. Bagaimana mengevaluasi kinerja *Convolutional Neural Network* yang telah diimplementasikan?

## 1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Data pelatihan merupakan citra yang diambil secara mandiri kemudian diambil karakter-karakternya secara mandiri.
2. Implementasi program menggunakan bahasa pemrograman *Python 3*.
3. Plat nomor merupakan plat nomor Indonesia
4. Karakter yang ada merupakan karakter A – Z serta angka 0 - 9.

## 1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk membangun sebuah sistem klasifikasi gambar dengan menggunakan metode klasifikasi *Convolutional Neural Network*

yang dapat mengenali nomor polisi kendaraan pada data video kendaraan.

## 1.5 Manfaat

Tugas akhir ini diharapkan dapat membantu menambah kemampuan yang ada pada pengenalan nomor polisi kendaraan dan membantu dalam melakukan pengenalan karakter sehingga dapat diimplementasikan pada sistem-sistem yang membutuhkan pengenalakan nomor polisi kendaraan secara otomatis.

## 1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

### 1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir yang berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri dari latar belakang diajukannya Tugas Akhir, rumusan masalah dan batasan masalah yang ditetapkan, serta manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

### 1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian literatur berupa jurnal yang digunakan sebagai referensi untuk pengerjaan tugas akhir ini. Literatur yang dipelajari pada pengerjaan tugas akhir ini berasal dari jurnal ilmiah yang diambil dari berbagai sumber di internet, beserta berbagai literatur online tambahan terkait *Convolutional Neural Network*, *TensorFlow* dan *Keras*.

### 1.6.3 Implementasi Perangkat Lunak

Pada tahap ini akan dilaksanakan implementasi metode dan algoritma yang telah direncanakan. Implementasi sistem

menggunakan *Python 3* sebagai bahasa pemrograman, *TensorFlow* dan *Keras* sebagai *framework*, serta *library* pendukung lainya.

### 1.6.4 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan menggunakan data video yang diambil secara mandiri serta untuk mengetahui hasil dan performa arsitektur yang telah dibangun. Evaluasi dilakukan dengan mengevaluasi model CNN dari segi akurasi, *precision*, *recall*, serta dengan menggunakan data video sebagai data *testing* untuk melihat performa model yang telah dibuat.

### 1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

## 1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

**Bab I Pendahuluan**

Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

**Bab II Tinjauan Pustaka**

Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang *Convolutional Neural Network* dan *library* yang digunakan.

**Bab III Perancangan Sistem**

Bab ini berisi pembahasan mengenai perancangan dari metode *Convolutional Neural Network* yang digunakan untuk pengenalan nomor polisi kendaraan pada data video.

**Bab IV Implementasi**

Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

**Bab V Uji Coba Dan Evaluasi**

Bab ini membahas tahapan uji coba, kemudian hasil uji coba dievaluasi terhadap kinerja dari sistem yang dibangun.

**Bab VI Kesimpulan dan Saran**

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

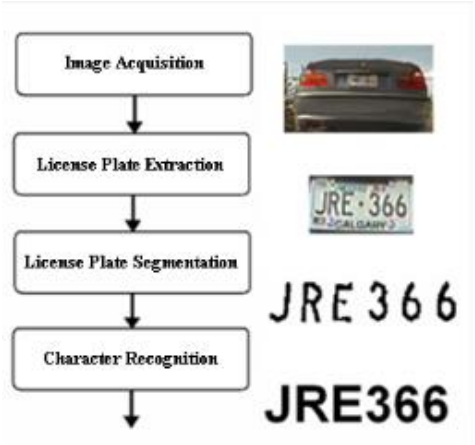
***(Halaman ini sengaja dikosongkan)***

# BAB II
# TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut adalah *Convolutional Neural Network*, dan beberapa teori lain yang mendukung pembuatan Tugas Akhir. Penjelasan ini bertujuan untuk memberikan gambaran umum dan diharapkan dapat mendukung sistem yang dibangun.

## 2.1 Pengenalan Nomor Polisi Kendaraan

Pengenalan nomor polisi kendaraan atau Automatic *License Plate Recognition* (ALPR) adalah kemampuan untuk mepengenalan karakter-karakter pada citra nomor polisi kendaraan , seperti yang terlihat pada Gambar 2.1. Pengenalan nomor polisi kendaraan merupakan bidang penting dalam *Computer Vision* dan penerapannya dapat digunakan pada pembayaran tol otomatis, pengaturan lalu lintas, *monitoring* lalu lintas, serta kontrol akses tempat parkir [2].

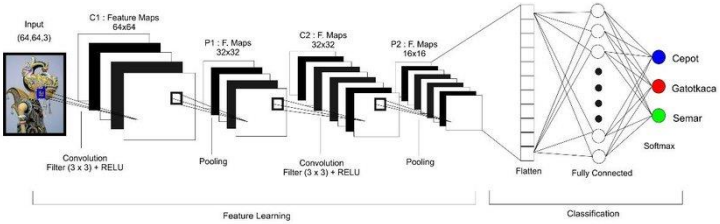



Gambar 2.1 Struktur Umum Pengenalan Nomor Polisi Kendaraan [2]

## 2.2 Convolutional Neural Network

*Convolutional Neural Network* (CNN) adalah salah satu algoritma dari *deep learning* yang merupakan pengembangan dari *Multi Layer Percepton* (MLP) yang dirancang untuk mengolah data dalam bentuk dua dimensi, misalnya gambar atau suara. CNN sering digunakan untuk mengenali citra benda atau pemandangan, melakukan deteksi dan segmentasi objek [3].

Penelitian awal yang mendasari penemuan ini dilakukan oleh Hubel dan Wiesel [4] yang melakukan penelitian visual korteks pada indera penglihatan kucing. Penelitian ini sangat berguna dalam sistem pemrosesan visual yang pernah ada. Hingga banyak penelitian yang terinspirasi dari cara kerjanya dan menghasilkan model-model baru. Arsitektur dari CNN dibagi menjadi 2 bagian besar, *Feature Learning* / *Extraction Layer* dan *Classification Layer* [5], seperti yang dipaparkan pada Gambar 2.2.




Gambar 2.2 Contoh arsitektur *Convolutional Neural Network* [6]

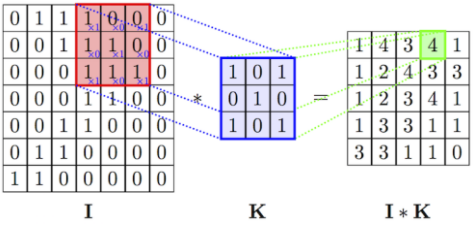
*Feature Learning* / *Extraction Layer* adalah bagian dimana terjadi proses penerjemahan dari sebuah citra menjadi *features. Features* ini berupa angka-angka yang merepresentasikan citra tersebut, yaitu berupa *feature map*. Proses ini terdiri dari *Convolutional Layer* dan *Pooling Layer*.

*Classification Layer* adalah dimana *feature map* yang dihasilkan dari *convolutional layers* masih berbentuk array multidimensi, sehingga harus dilakukan pengubahan *feature map* menjadi sebuah *feature vector* agar bisa digunakan sebagai

masukan dari *fully connected layer*. *Fully connected layer* yang dimaksud disini adalah *Multi Layer Perceptron* (MLP) yang memiliki beberapa *hidden layer*, *activation function*, *output layer* dan *loss function*.

### 2.2.1 Convolution Layer

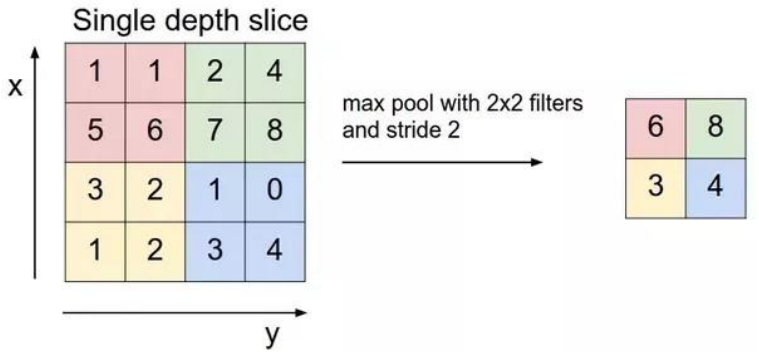
*Convolution Layer* melakukan operasi konvolusi pada output dari lapisan sebelumnya. Konvolusi adalah istilah matematis yang artinya mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstrak fitur dari citra masukan [3]. Ilustrasi cara kerja konvolusi bisa dilihat pada Gambar 2.3, dimana I adalah citra, K adalah *filter* atau *kernel* yang digunakan, I * K adalah hasil operasi konvolusi.




Gambar 2.3 Ilustrasi cara kerja konvolusi [7]

### 2.2.2 Pooling Layer

Fungsi dari *Pooling Layer* adalah mereduksi ukuran dari data. Terdapat beberapa tipe *Pooling Layer* diantaranya yaitu *max*, *average*, *sum* dan lainnya. Metode *Pooling* dalam CNN yang biasa digunakan adalah *Max Pooling* & *Average Pooling*. *Max Pooling* membagi *output* dari *Convolution Layer* menjadi beberapa matriks kecil lalu mengambil nilai maksimal dari tiap matriks untuk menyusun matriks citra yang telah direduksi, sedangkan *Average Pooling* akan memilih nilai rata-ratanya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun obyek citra mengalami translasi. Ilustrasi cara kerja *Max Pooling* bisa dilihat pada Gambar 2.4.



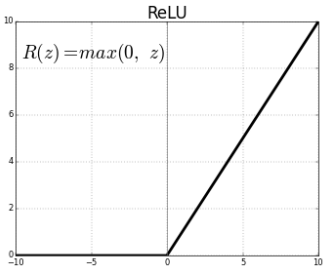

Gambar 2.4 Ilustrasi cara kerja *Max Pooling* [8]

### 2.2.3 Fully Connected Layer

*Fully Connected Layer* dalam penerapannya sama dengan *Multi Layer Perceptron* (MLP) yang bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. *Feature map* dari *Convolution Layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu yang disebut *feature vector* sebelum dapat dimasukkan ke dalam sebuah *Fully Connected Layer*. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversibel, *Fully Connected Layer* diimplementasikan di akhir jaringan [9].

### 2.2.4 ReLU Activation Function

Fungsi aktivasi befungsi untuk menentukan apakah neuron tersebut harus aktif atau tidak berdasarkan nilai masukan. Salah satu contoh fungsi aktivasi adalah ReLU (Rectified Linear Unit) dimana fungsi ini melakukan *thresholding* dengan nilai nol terhadap nilai masukan, dimana seluruh nilai yang kurang dari nol akan dijadikan nol, seperti pada Gambar 2.5.




Gambar 2.5 ReLU *Activation Function* [10]

### 2.2.5 Fungsi Softmax

Fungsi *softmax* biasa digunakan dalam klasifikasi banyak kelas. *Softmax* memberikan nilai probabilitas untuk setiap label kelas, dimana jumlah seluruh probabilitas adalah 1. *Softmax* pada dasarnya adalah probabilitas eksponensial yang dinormalisasi dari nilai masukan sejumlah kelas pada model klasifikasi seperti pada Persamaan (2.1).

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.1}$$

Dimana y adalah nilai masukan. Operasi akan menghasilkan nilai probabilitas. Label dari data masukan akan ditentukan berdasarkan kelas dengan nilai probabilitas tertinggi.

### 2.2.6 Cross Entropy

*Loss function* merupakan fungsi yang menggambarkan kerugian yang dihasilkan oleh model. *Loss function* dikatakan baik, ketika menghasilkan *error* yang diharapkan paling rendah. Pada permasalahan klasifikasi banyak kelas, *cross entropy* adalah *loss function* yang biasa digunakan. *Cross entropy* akan menghitung *error* antara nilai prediksi *S* dengan nilai sebenarnya *T*, seperti pada Persamaan (2.2). Selanjutnya, nilai *error* akhir diambil dari rata-rata hasil *cross entropy*, seperti pada Persamaan (2.3).

$$D(S_i, T_i) = -\sum_j T_{ij} \log S_{ij} \quad (2.2)$$

$$J(W, b) = \frac{1}{n} \sum_i D(S_i, T_i) \quad (2.3)$$

### 2.2.7 Stochastic Gradient Descent

Ketika melatih sebuah model, dibutuhkan sebuah *loss function* yang dapat mengukur kualitas dari setiap bobot atau parameter tertentu. *Stochastic Gradient Descent* (SGD) adalah algoritma pengotimalan. Tujuan pengoptimalan adalah untuk menemukan parameter yang dapat meminimalkan nilai *error* dari *loss function.* SGD adalah algoritma yang digunakan untuk memperbarui nilai bobot dan bias pada neuron di *neural network.* Pada dasarnya operasi yang dilakukan hanya mengurangi bobot awal dengan sebagian nilai dari nilai gradien yang sudah kita dapat. Nilai sebagian disini diwakili oleh parameter bernama *learning rate*, seperti yang terlihat pada Persamaan (2.4) dan (2.5).

$$w_{j+1} = w_j - \alpha \frac{\partial}{\partial w_j} J(W, b) \quad (2.4)$$

$$b_{j+1} = w_j - \alpha \frac{\partial}{\partial b_j} J(W, b) \quad (2.5)$$

### 2.2.8 Adagrad

Adagrad adalah algoritma pengoptimalan berbasis gradien yang memperbarui *learning rate* setiap parameternya. Adagrad adalah algoritma yang digunakan untuk memperbarui nilai bobot dan bias pada neuron di *neural network*. Adagrad menggunakan *adaptive learning rate* untuk setiap parameter. Berbeda dengan *Stochastic Gradient Descent* (SGD) yang selalu menggunakan *learning rate* yang sama, Adagrad memiliki sebuah *learning rate* untuk setiap parameter dan secara terpisah beradaptasi saat proses pelatihan. Pembaruan dilakukan untuk tiap parameter **θ(j)** dengan gradien *loss function* **g(j)**, seperti yang dapat dilihat pada Persamaan (2.6) dan (2.7).

$$g_j = \frac{\partial}{\partial \theta_j} J(\theta_j) \tag{2.6}$$

$$\theta_{j+1,} = \theta_j - \frac{\alpha}{\sqrt{\sum_{i=0}^{j} (g_i)^2}} g_j \tag{2.7}$$

### 2.2.9 RMSProp

RMSProp (*Root Mean Square*) adalah metode pengoptimalan berbasis *adaptive learning rate* yang diusulkan oleh Geoffrey Hinton [11]. RMSProp memodifikasi Adagrad dengan mengganti akumulasi gradien menjadi rata-rata bergerak gradien yang diberi bobot secara kuadratik, seperti yang dapat dilihat pada Persamaan (2.8) dan (2.9).

$$s_j = \beta . s_{j-1} + (1 - \beta)(g_j)^2 \tag{2.8}$$

$$\theta_{j+1} = \theta_j - \frac{\alpha}{\sqrt{s_j + \varepsilon}} g_j \tag{2.9}$$

### 2.2.10 Adam

Adam (*Adaptive Moment Estimation)* juga adalah algoritma pengoptimalan yang dapat digunakan. Hampir sama dengan Adagrad, Adam memiliki sebuah *learning rate* untuk setiap parameter dan secara terpisah beradaptasi saat proses pelatihan. Adam memperbarui nilai setiap parameter seperti RMSProp [12]. Perbedaannya Adam menggunakan gradien yang telah diperhalus dan semakin mengecil seperti yang dapat dilihat pada Persamaan (2.10). Lalu gradien tersebut akan digunakan untuk memperbarui parameter, seperti yang dapat dilihat pada Persamaan (2.11) dan (2.12).

$$m_j = \beta_1 . m_{j-1} + (1 - \beta_1) . g_j \tag{2.10}$$

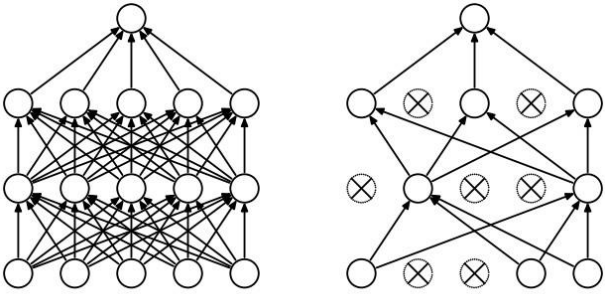
$$s_j = \beta_2 . s_{j-1} + (1 - \beta_2)(g_j)^2 \tag{2.11}$$

$$\theta_{j+1} = \theta_j - \frac{\alpha}{\sqrt{s_j + \varepsilon}} m_j \tag{2.12}$$

### 2.2.11 Dropout

*Dropout* merupakan proses mencegah terjadinya *overfitting* dan juga mempercepat proses learning. *Dropout* mengacu kepada menghilangkan neuron yang berupa *hidden layer* mapun *visible layer* di dalam jaringan [13]. Dengan menghilangkan suatu neuron, berarti menghilangkannya sementara dari jaringan yang ada. Neuron yang akan dihilangkan akan dipilih secara acak.

Pada Gambar 2.6, (a) neuron tetap utuh pada *neural network* yang belum memakai *Dropout*, dan (b) *neural network* yang sebagian dari neuronnya tidak digunakan setelah diaplikasikan Dropout.




Gambar 2.6 Ilustrasi *neural network* dalam mengaplikasikan *Dropout* [14]

### 2.2.12 Batch Normalization

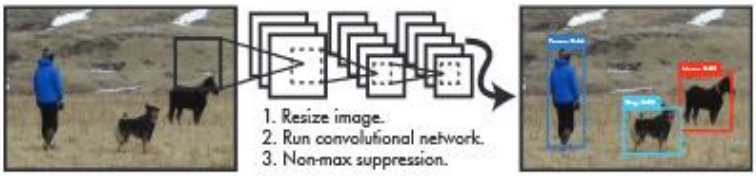
*Batch Normalization* adalah teknik melakukan normalisasi terhadap *batch* atau kumpulan data masukan, seperti yang terlihat pada Persamaan (2.13). Dimana *x* adalah nilai masukan, $\mu_b$ adalah *mean* dari *batch*, $\sigma_b$ adalah standar deviasi dari *batch*. Normalisasi dilakukan agar data memiliki *mean* mendekati 0 dan standar deviasi mendekati 1.

$$x_{baru} = \frac{x-\mu_b}{\sigma_b} \tag{2.13}$$

## 2.3 You Only Look Once (YOLO)

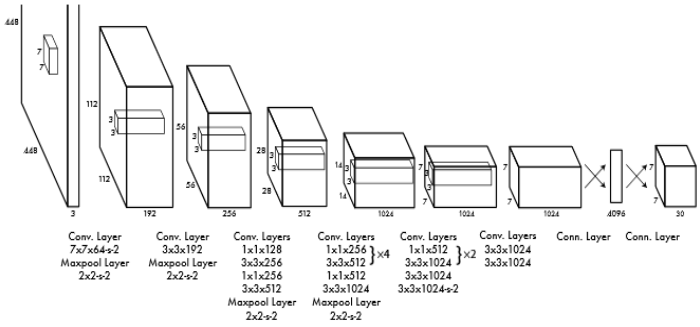
YOLO adalah sebuah metode deteksi objek. YOLO melihat deteksi objek sebagai sebuah permasalahan regresi, mulai dari piksel, koordinat *bounding box*, dan probabilitas kelas [15].

YOLO menerapkan sebuah *neural network* secara bersamaan untuk melakukan prediksi *bounding box* dan probabilitas kelas untuk *bounding box* bersangkutan seperti pada Gambar 2.7.




Gambar 2.7 Ilustrasi Cara Kerja YOLO [15]

YOLO sendiri dikembangkan dengan berpatokan pada GoogLeNet model untuk klasifikasi citra [16]. *Network* YOLO sendiri terdiri dari 24 *Convolutional Layers* kemudian diikuti dengan 2 *fully connected layers*. Secara utuh, network dapat dilihat pada Gambar 2.8




Gambar 2.8 Arsitektur YOLO [15]

## 2.4 Otsu Thresholding

Merupakan sebuah metode *thresholding* citra di mana diasumsikan pada sebuah citra bimodal yang mengandung dua kelas piksel yang tinggi, akan dikalkulasi nilai *threshold* optimum yang memisahkan dua kelas agar *intra-class variance* minimal atau setara, agar *inter-class variance* maksimal [17].

## 2.5 SIFT

SIFT (*Scale-invariant feature transform*) merupakan algoritma deteksi fitur untuk mendeteksi dan mendeskripsikan fitur lokal dalam suatu citra.

SIFT *keypoints* dari suatu objek awalnya diekstrak dari citra referensi dan disimpan. Sebuah objek dikenali dari citra lain dengan cara membandingkan secara individual setiap fitur dari citra baru kepada *keypoints* yang telah disimpan dan mencari kandidat fitur yang cocok dengan menggunakan *Euclidean Distance* pada *vector* fitur [18].

## 2.6 Augmentasi Data

Dalam mendapatkan performa yang optimal, umumnya *deep learning* seperti *Convolutional Neural Network* membutuhkan data yang lebih banyak dibandingkan dengan algoritma yang lain, untuk itu kita perlu melakukan augmentasi data. Augmentasi data adalah sebuah teknik menambah data dengan cara memanipulasi data yang telah ada dengan pengaturan keragaman tertentu. Untuk data berupa citra, kita bisa lakukan operasi seperti translasi, refleksi, rotasi, perbesaran ukuran, dan lain-lain.

### 2.6.1 Rotasi Citra

Rotasi citra adalah salah satu teknik yang digunakan dalam augmentasi data yang memodifikasi citra yang berakibat pada citra bersangkutan mengalami rotasi berdasarkan sudut yang telah ditentukan. Seperti rotasi 10 derajat, 30 derajat, dan lain-lain.

### 2.6.2 Perbesaran Ukuran Gambar

Perbesaran ukuran gambar adalah proses memperbesar ukuran gambar secara acak untuk memperoleh gambar yang lebih variatif. Gambar diperbesar pada salah satu titiknya untuk dicari keragaman lain dari gambar.

## 2.7 Akurasi, Precision & Recall

Ketika membangun sebuah model klasifikasi, pertanyaan yang muncul adalah bagaimana mengetahui seberapa baik model tersebut. Mengevaluasi model klasifikasi dilakukan dengan mencari tahu seberapa baik hasil prediksi dari model tersebut. *Recall* di Persamaan (2.14), *precision* di Persamaan (2.15), dan akurasi di Persamaan (2.16) adalah metode pengukuran yang biasa digunakan dalam mengevaluasi model, penjelasan variabel ada pada *confusion matrix* pada Tabel 2.1.

Tabel 2.1 *Confusion matrix*

| | | **Kelas Prediksi** | |
|---|---|---|---|
| | | Benar | Salah |
| **Kelas sebenarnya** | Benar | *True Positive* (TP) | *False Negative* (FN) |
| | Salah | *False Positive* (FP) | *True Negative* (TN) |

Keterangan :

1. *True Positive* (TP) pada gambar dikenali karakter secara benar, pada keluaran program mengenali ekspresi tersebut.
2. *True Negative* (TN) pada gambar tidak dikenali karakter, pada keluaran program tidak mengenali karakter tersebut.
3. *False Positive* (FP) pada gambar dikenali karakter, pada keluaran program tidak mengenali karakter tersebut.
4. *False Negative* (FN) pada gambar tidak dikenali karakter, pada keluaran program mengenali karakter tersebut.

$$Recall = TP / (TP + FN) \quad (2.14)$$

$$Precision = TP / (TP + FP) \quad (2.15)$$

$$Akurasi = (TP + TN) / (TP + FP + TN + FN) \quad (2.16)$$

## 2.8 Python

Python adalah bahasa pemograman yang populer. *Python* sering dimanfaatkan dalam pengembangan web, perangkat lunak, penelitian, dan *system scripting*. Python dapat digunakan untuk menangani data besar dan melakukan operasi matematika yang kompleks. Python bekerja di berbagai *platform* seperti Windows, Mac, Linux, Raspberry Pi, dan lain-lain. Python dirancang untuk mudah dibaca, yaitu memiliki sintaks yang sederhana dan menggunakan bahasa inggris [19].

## 2.9 Library

*Library* merupakan sekumpulan program yang dapat digunakan pada program lain tanpa terikat satu dengan yang lainnya. Terdapat beberapa *library* yang digunakan dalam melakukan implementasi tugas akhir ini. *Library* yang digunakan antara lain, Keras, TensorFlow, OpenCV, Numpy, Scikit-learn, dan Matplotlib

### 2.9.1 Keras

Keras adalah *high-level neural networks* API, yang ditulis dalam bahasa pemograman Python dan mampu berjalan di atas TensorFlow dan Theano. Keras dikembangkan dalam rangka memungkinkan eksperiman dilakukan dengan cepat. Keras dapat berjalan baik di CPU dan GPU. Keras berisi banyak implementasi *neural network* yang umum digunakan, fungsi aktivasi, *optimizer*, dan *tool* lain yang memudahkan dalam pengolahan citra dan data teks [20].

### 2.9.2 TensorFlow

TensorFlow adalah *library open source* untuk pembuatan program yang membutuhkan komputasi numerik berkinerja tinggi. TensorFlow dikembangakan oleh tim Google Brain. TensorFlow

menyediakan fungsi-fungsi *machine learning* dan *deep learning*, dan dapat dijalankan dalam CPU atau GPU [21].

### 2.9.3 OpenCV

OpenCV (*Open Source Computer Vision*) adalah *library* yang dimanfaatkan dalam pengolahan citra dinamis secara *real-time*. OpenCV dapat digunakan dalam berbagai bahasa pemograman seperti Python, C++, Java, atau MATLAB. OpenCV memiliki fitur seperti *Feature & Object Detection, Motion Analysis and Object Tracking, Image Filtering, Image Processing*, dan lain-lain [22].

### 2.9.4 Numpy

Numpy adalah *library Python* yang mendukung pengolahan data pada *array* dan matriks multidimensi yang besar. Numpy menyediakan kumpulan fungsi matematika, seperti aljabar linear, transformasi Fourier, pembuatan angka acak, dan lain-lain. *Numpy* bersifat *open source* sehingga banyak dimanfaatkan dalam pengolahan data peneilitian [23].

### 2.9.5 Scikit-learn

Scikit-learn adalah *open source machine learning library* untuk bahasa pemograman Python. Scikit-learn menyediakan fitur seperti *classification*, *regression*, *clustering*, termasuk juga didalamnya algoritma *support vector machines*, *random forest*, *gradient boosting*, dan lain-lain [24].

### 2.9.6 Matplotlib

Matplotlib adalah *library Python* yang mendukung pembuatan grafik dua dimensi dalam berbagai format dan dari berbagai jenis data. Matplotlib bersifat *open source* dan banyak digunakan untuk pengolahan data dalam penelitian. Matplotlib dapat membuat plot, histogram, spektrum daya, diagram batang, diagram kesalahan, plot pencar, dan lain-lain [25].

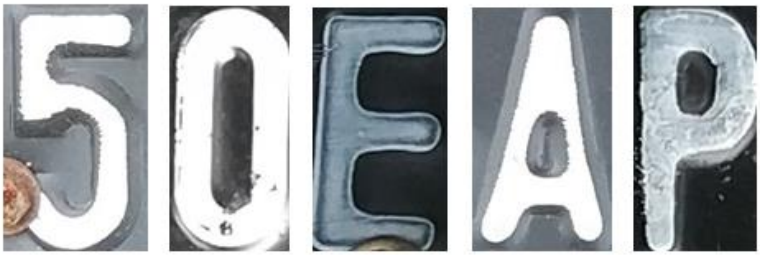
***(Halaman ini sengaja dikosongkan)***

# BAB III
# PERANCANGAN SISTEM

Bab ini menjelaskan mengenai perancangan data dan sistem pengenalan nomor polisi kendaraan menggunakan *Convolutional Neural Network*. Bab ini juga akan menjelaskan gambaran umum sistem dalam bentuk diagram alir.

## 3.1 Perancangan Data

Data yang digunakan sebagai masukan awal dari sistem pengenalan nomor polisi kendaraan menggunakan *Convolutional Neural Network* adalah data yang diambil secara mandiri yang terdiri dari 36 kelas di mana di masing-masing kelas terdapat 10 sampai dengan 11 citra. Data ini merupakan sekumpulan citra karakter A sampai dengan Z serta angka 0 sampai dengan 9. Citra-citra ini merupakan hasil pemotongan mandiri yang sebelumnya berasal dari citra nomor polisi kendaraan. Contoh karakter bisa dilihat pada Gambar 3.1.



Gambar 3.1 Contoh karakter pada *dataset*

Pada setiap karakter terdapat beberapa variasi sudut pengambilan, seperti tampak depan yang tegak lurus dengan kamera, rotasi ke kanan, rotasi ke kiri, miring dari atas, miring dari kanan, miring dari kiri. Citra-citra ini dapat dilihat pada Gambar 3.2

Gambar 3.2 Contoh pengambilan citra dari sudut pandang yang bervariasi

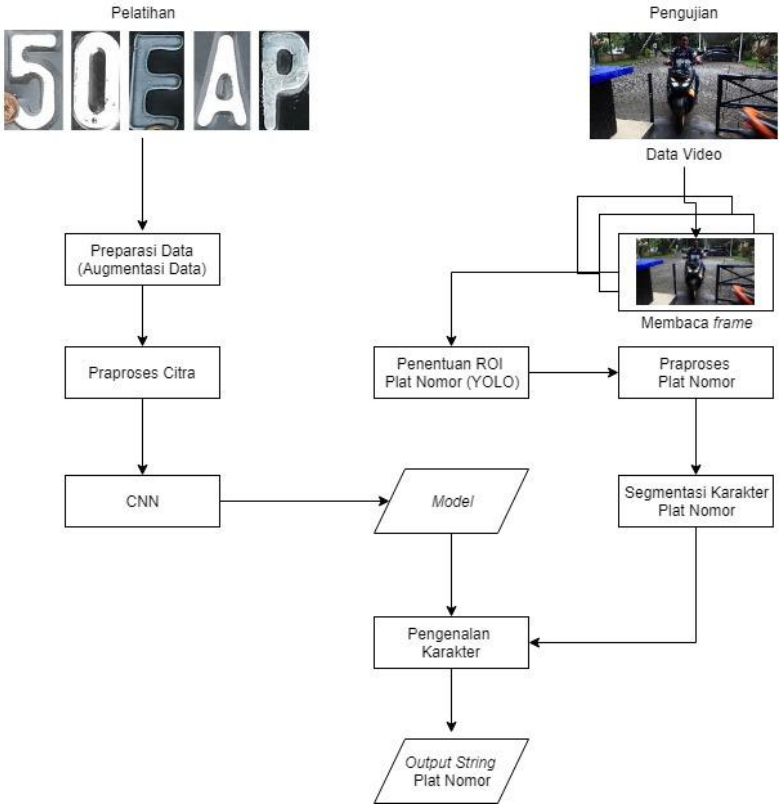
Dataset yang digunakan awalnya terdiri dari 6 - 11 citra tiap kelas. Data tiap kelas ini kemudian di augmentasi hingga nantinya terdapat 75932 data secara total. Pembagian data latih dan data uji menjadi 53169 data uji dan 22763 data latih. Spesifikasi lengkap dataset dapat dilihat pada Tabel 3.1. Sedangkan pada data uji, data uji merupakan kumpulan video dengan dengan ukuran 1920x1080 di mana setiap video terdiri berdurasi 1 – 5 menit.

Tabel 3.1 Spesifikasi awal dataset

| **Keterangan** | **Spesifikasi** |
|---|---|
| Ekstensi | .jpg |
| Jumlah gambar | 300 - 360 |
| Jumlah kelas | 36 kelas |
| Jumlah gambar per kelas | 6 - 11 |
| Ukuran file | 1 - 30 kB |
| Kanal warna | 3 (RGB) |

## 3.2 Desain Umum Sistem

Sistem pengenalan nomor polisi kendaraan yang dibangun memiliki proses utama diantaranya praproses data, pelatihan dan pengujian *Convolutional Neural Network*. Diagram alir dari sistem ditunjukan pada Gambar 3.3.




Gambar 3.3 Diagram alir sistem yang dibangun

Akan dilakukan praproses data terlebih dahulu sebelum data digunakan sebagai data untuk melatih model. Setiap data akan dijadikan format *grayscale*, lalu dilakukan *thresholding* untuk membuat citra menjadi citra biner. Citra biner ini kemudian akan di lakukan padding, dengan menambahkan pixel bernilai 0 (warna hitam) hingg a membuat rasio citra menjadi 1:1. Citra biner tersebut kemudian akan dilakukan *resize* untuk mengubah ukuran citra menjadi ukuran 32x32.

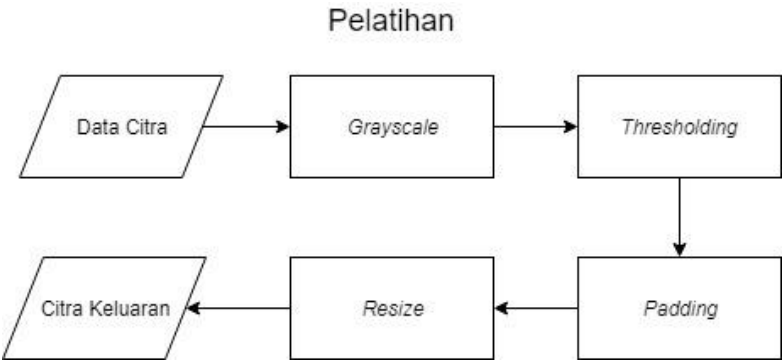
Proses pelatihan adalah proses pembuatan model klasifikasi karakter dari nomor polisi kendaraan. Data latih yang telah dilakukan praproses data akan diekstraksi fiturnya yakni melalui proses konvolusi, *pooling*, dan fungsi-fungsi aktivasi. Selanjutnya, hasil dari proses pelatihan tersebut akan menjadi model untuk proses klasifikasi. Data uji juga akan dilakukan praproses data dahulu sebelum diekstraksi fiturnya untuk menjadi *input* dalam proses pengujian.

Pada data uji, proproses data dilakukan dengan cara yang sama seperti data latih dengan beberapa perbedaan. Citra dijadikan citra *grayscale*, kemudian citra akan di *thresholding* sehingga citra menjadi citra biner. Kemudian, akan diterapkan operasi morfologi pada citra, yaitu dengan melakukan operasi erosi, lalu dilakukan dilasi sebanyak dua kali, lalu citra akan kembali di erosi. *Kernel* yang digunakan sama pada setiap operasi morfologi, yakni 3x3. Setelah di morfologi, Citra biner ini akan di-*padding* hingga rasio citra menjadi 1:1. Kemudian diterapkan *resize* pada citra, yang kemdian citra baru dimasukkan ke dalam *network* CNN.

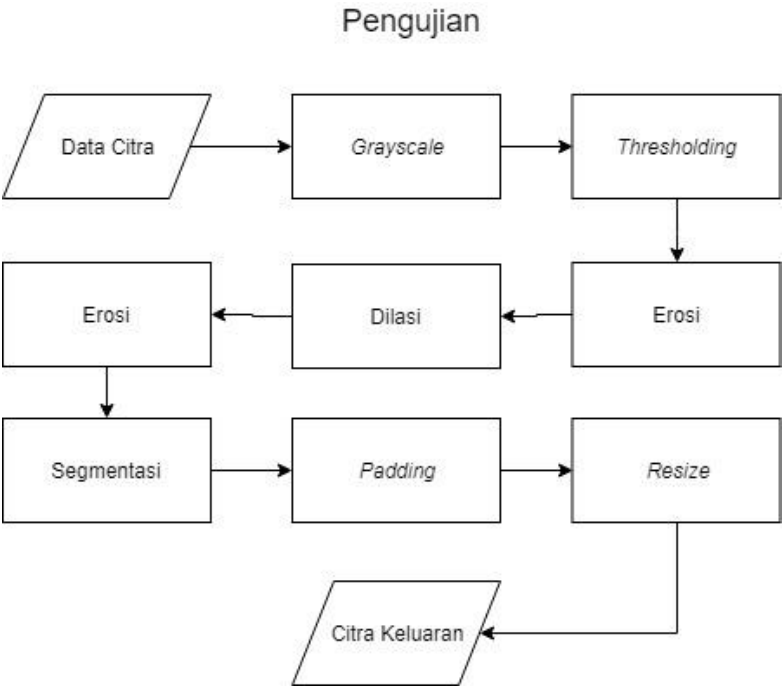
Proses pengujian memanfaatkan fungsi *softmax* untuk mengetahui label kelas dengan menghitung nilai probabilitas tertinggi. Selanjutnya prediksi tersebut akan dibandingkan dengan label kelas sebenarnya, maka dapat dievaluasi nilai akurasi, *precision*, dan *recall* dari model tersebut.

### 3.2.1 Tahap Praproses Data

Pada tugas akhir ini, akan dilakukan praproses data sebelum citra nantinya akan dijadikan *input network*. Diagram alir praproses data dapat dilihat pada Gambar 3.4.




Gambar 3.4 Diagram alir praproses data pelatihan




Gambar 3.5 Diagram alir praproses data pengujian

Setiap data akan dijadikan format *grayscale*, lalu dilakukan *thresholding* untuk membuat citra menjadi citra biner. Citra biner ini kemudian akan di lakukan padding, dengan menambahkan pixel bernilai 0 (warna hitam) hingga membuat rasio citra menjadi 1:1. Citra biner tersebut kemudian akan dilakukan *resize* untuk mengubah ukuran citra menjadi ukuran 32x32.

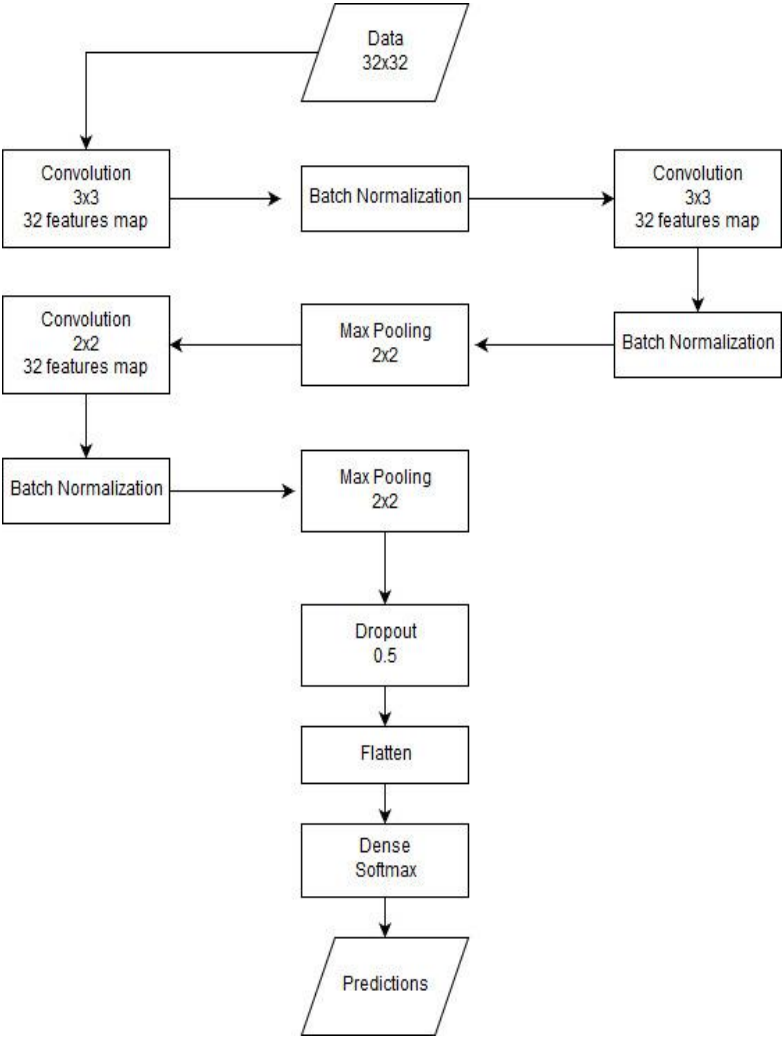
Pada data uji, proproses data dilakukan dengan cara yang sama seperti data latih dengan beberapa perbedaan. Citra dijadikan citra *grayscale*, kemudian citra akan di *thresholding* sehingga citra menjadi citra biner. Kemudian, akan diterapkan operasi morfologi pada citra, yaitu dengan melakukan operasi erosi, lalu dilakukan dilasi sebanyak dua kali, lalu citra akan kembali di erosi. *Kernel* yang digunakan sama pada setiap operasi morfologi, yakni 3x3. Setelah di morfologi, karakter akan segmentasi. Segmentasi dilakukan dengan mencari *contour* dari karakter pada plat. Contour dicari dengan mencari *tracing* pada *edge* area berwarna putih. Citra karakter biner ini akan di-*padding* hingga rasio citra menjadi 1:1. Kemudian diterapkan *resize* pada citra, yang kemdian citra baru dimasukkan ke dalam *network* CNN.

Selain praproses yang telah dijelaskan di atas, akan dilakukan juga proses augmentasi data untuk memperbanyak data latih yang dimasukkan ke arsitektur CNN. Proses augmentasi data dilakukan untuk memperbanyak variasi data. Banyaknya variasi data diharapkan dapat meningkatkan kualitas data dan model yang dibangun dapat memiliki hasil yang lebih baik. Augmentasi yang dilakukan adalah dengan melakukan rotasi dan perbesaran ukuran citra.

### 3.2.2 Tahap Pembangunan Arsitektur

Pembangunan model bertujuan untuk menyiapkan *layer*, fungsi aktivasi, *loss function*, dan parameter apa saja yang dibutuhkan. CNN (*Convolutional Neural Network*) yang digunakan terdiri dari *convolutional layer*, *max pooling layer* dan

*fully connected layer*. *Dropout layer* juga digunakan untuk menghindari *overfitting*. Arsitektur CNN dapat dilihat pada Gambar 3.6.




Gambar 3.6 Arsitektur CNN yang digunakan

Berikut detail arsitektur CNN:

1. Input untuk arsitektur CNN adalah data yang sudah di praproses sebelumnya. Di mana data merupakan data karakter yang akan diprediksi kelasnya.
2. Input kemudian akan melewati *Convolution Layer* dengan ukuran ukuran *kernel* 3x3 dengan *stride* 1. Sebelum masuk ke *Convolution Layer* kedua, hasil *input* akan dilakukan Batch Normalization, kemudian melewati *Convolution Layer* kedua, dan kembali dengan ukuran *kernel* 3x3 yang sama dengan ukuran *stride* 1, lalu kembali dilakukan *Batch Normalization*. Setelah itu akan dilakukan *Max Pooling* dengan *pool size* 2x2. Setelah itu, data akan melewati *Convolution Layer* ketiga dengan ukuran *kernel* 2x2, lalu akan dilakukan *Batch Normalization*, dan akhirnya melewati *Max Pooling* dengan *pool size* 2x2.
3. Semua *Convolution Layer* memiliki 32 feature map.
4. Setelah melewati *Max Pooling Layer*, data akan melewati *Dropout Layer*. Setelah melewati *Dropout Layer, input* kemudian dilakukan *Flatten* untuk membuat input matriks yang sebelumnya matriks menjadi *vector* fitur, dan setelah itu, *vector* fitur ini akan masuk ke dalam *Fully Connected Layer* dengan fungsi Softmax.
5. *Dropout layer* memiliki nilai probabilitas 0.5.
6. *Semua Max Pooling Layer* memiliki *pool size* berukuran 2x2 dan *stride* berukuran 2x2.
7. CNN dilatih menggunakan *Adam Optimizer* dengan *learning rate* 0,0001.
8. Data yang digunakan memiliki 1 kanal warna (*grayscale*) di mana data ini merupakan data citra biner, yaitu citra hitam putih.

Pembangunan arsitektur CNN mengikuti arsitektur yang dibangun oleh Peiqi Wu dkk dengan detail seperti pada Tabel 3.2.

Tabel 3.2 Arsitektur CNN

| ***Layer*** | ***Input*** | ***Output*** | **Spesifikasi** |
|---|---|---|---|
| *Input Layer* | (1, *32*, *32*, 1) | | - |
| *Convolution Layer* 1 | (1, 32, 32, 1) | (1, 30, 30, 32) | *Filter*: 3 x 3 – 32<br>*Stride*: 1 x 1<br>Fungsi: Linier |
| *Batch Normalization* | (1, 30, 30, 32) | (1, 30, 30, 32) | |
| *Convolution Layer* 2 | (1, 30, 30, 32) | (1, 28, 28, 32) | *Filter*: 3 x 3 – 32<br>*Stride*: 1 x 1<br>Fungsi: Leaky ReLU |
| *Batch Normalization* | (1, 28, 28, 32) | (1,28, 28, 32) | |
| *Max Pooling Layer* 1 | (1, 28, 28, 32) | (1, 14, 14, 32) | *Kernel*: 2 x 2 |
| *Convolution Layer* 3 | (1, 14, 14, 32) | (1, 13, 13, 32) | *Filter*: 2 x 2 – 32<br>*Stride*: 1 x 1<br>Fungsi:Leaky ReLU |
| *Batch Normalization* | (1, 13, 13, 32) | (1, 13, 13, 32) | |
| *Max Pooling Layer 2* | (1, 13, 13, 32) | (1, 6, 6, 32) | *Filter*: 2 x 2 |
| *Dropout Layer* 1 | (1, 6, 6, 32) | (1, 6, 6, 32) | Fungsi: *Dropout 0.5* |
| *Fully Connected Layer* | (1, 1, 1, ) | 36 | Fungsi : Softmax |

### 3.2.3 Tahap Pelatihan dan Pengujian CNN

Setelah melakukan augmentasi, serta praproses terhadap data dan menyiapkan arsitektur. Data akan dibagi menjadi 2, yaitu data latih dan data uji dengan jumlah data latih sebanyak 53169 citra dan jumlah data uji sebanyak 22763 citra. Pada awalnya data hanya berjuma 280 sampai dengan 360 citra di mana setiap kelas karakter, terdapat 8 sampai dengan 11 citra. Citra total menjadi 75932 citra dikarenakan telah dilakukan augmentasi yang bertujuan untuk memperbanyak variasi citra untuk dimasukkan ke dalam *network* CNN yang telah dibuat.

Proses pelatihan memanfaatkan data latih untuk membangun model CNN. Pelatihan menggunakan Adam *Optimizer* dengan *learning rate* yang sudah ditentukan sebelumnya. Proses pelatihan akan dijalankan pada *batch size* 50 dan jumlah *epoch*

menyesuaikan skenario uji coba. Dalam setiap akhir *epoch* terdapat proses pengujian model terhadap data uji untuk mengetahui seberapa baik model dilatih. Lalu pada akhir pelatihan akan dilakukan pengujian untuk mendapatkan nilai akurasi, *precision* dan *recall*.

# BAB IV
# IMPLEMENTASI

Bab ini menjelaskan mengenai implementasi perangkat lunak dari rancangan sistem yang telah dibahas pada Bab 3 meliputi kode program dalam perangkat lunak. Selain itu, implementasi dari tiap proses, parameter masukan, keluaran, dan beberapa keterangan yang berhubungan dengan program juga dijelaskan.

## 4.1 Lingkungan Implementasi

Dalam mengimplementasikan aplikasi pengenalan ekspresi manusia diperlukan beberapa perangkat pendukung sebagai berikut.

### 4.1.1 Perangkat Keras

Implementasi tugas akhir ini menggunakan desktop *personal computer* (PC) MS-7886. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi AMD Ryzen 7 2700 dengan kecepatan 3,2 GHz, *Random Access Memory* (RAM) sebesar 16 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA GeForce RTX 2080 sebesar 8 GB.

### 4.1.2 Perangkat Lunak

PC dari sisi perangkat lunak memiliki spesifikasi antara lain menggunakan bahasa pemograman Python 3.6, dilengkapi dengan *library* antara lain OpenCV, Tensorflow-GPU, Keras-GPU, Numpy, Matplotlib dan Scikit-learn.

## 4.2 Implementasi Augmentasi Data

Pada subbab ini akan dijelaskan proses augmentasi data yaitu refleksi dan memperbesar ukuran gambar.

### 4.2.1 Rotasi

Refleksi diimplementasikan pada Kode Sumber 4.7. Proses rotasi dilakukan dengan menggunakan fungsi *ImageDataGenerator* dari library Keras. Parameter *rotation_range=10* menandakan derajat rotasi yang akan diimplementasikan pada citra. Lalu setelah objek *ImageDataGenerator* dibuat, akan dipanggil fungsi *flow* untuk menerapkan pengaturan augmentasi yang telah dibuat.

### 4.2.2 Perbesar Ukuran Objek

Perbesaran ukuran gambar dilakukan untuk mencari keragaman yang lebih variatif dari data. Perbesaran ukuran gambar diimplementasikan pada Kode Sumber 4.1. Proses perbesaran ukuran gambar dilakukan dengan menggunakan fungsi *ImageDataGenerator* dari library Keras. Parameter *zoom_range*=0.1 menandakan perbesaran ukuran gambar sebesar 10% akan diimplementasikan pada data gambar secara acak. Lalu setelah objek *ImageDataGenerator* dibuat, akan dipanggil fungsi *flow* untuk menerapkan pengaturan perbesaran ukuran gambar yang telah dibuat.

```
1. datagen = ImageDataGenerator(rotation_range=10,
   width_shift_range=0.1, height_shift_range=0.1,
   shear_range=0.1,zoom_range=0.1,
   horizontal_flip=False, fill_mode='nearest')
2. self.datagen.flow(self.data, batch_size=1,
   save_to_dir=path, save_prefix=prefix,
   save_format='jpg')
```

Kode Sumber 4.1 Implementasi augmentasi data

## 4.3 Implementasi Praproses Data

Pada subbab ini akan dijabarkan implementasi pada tahap praproses data, yaitu *Grayscale*, *Thresholding*, Erosi, Dilasi, *Padding*, serta *Resize*.

### 4.3.1 Implementasi Perubahan Kanal Citra Menjadi Grayscale

Proses perubahan kanal citra diimplementasikan pada Kode Sumber 4.2. Proses perubahan kalan citra yang awalnya 3 kanal menjadi 1 kanal dilakukan pada citra dengan memanfaatkan fungsi *cvtColor* dari OpenCV. Dimana parameter yang digunakan adalah *frame* sebagai input citra yang akan diubah kanalnya dan *cv2.COLOR_BGR2GRAY* sebagai mode perubahan kanal, di mana BGR merupakan kanal awal saat citra akan diproses.

```
1.  grayy = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Kode Sumber 4.2 Fungsi perubahan kanal citra

### 4.3.2 Implementasi Thresholding pada Citra

Proses *Thresholding* merupakan proses yang membuat sebuah citra 1 kanal menjadi sebuah citra biner. Citra biner merupakan citra yang setiap pikselnya hanya bernilai 1 atau 0. Dalam hal ini, fungsi *threshold* OpenCV akan digunakan untuk melakukan proses ini seperti pada Kode Sumber 4.3. Fungsi threshold pada OpenCV sendiri akan memberikan *return* citra dengan nilai piksel 0 atau 255.

```
1.  ret,binImg=cv2.threshold(img,0,255,cv2.THRESH_BINAR
    Y+cv2.THRESH_OTSU)
```

Kode Sumber 4.3 Fungsi *Otsu Thresholding*

Metode *thresholding* yang digunakan adalah Otsu. Parameter yang digunakan pada fungsi *threshold* ini adalah, *img*, yang merupakan matriks citra masukan. Lalu, parameter kedua, dan ketiga secara berturut-turut adalah nilai batas dan nilai yang akan digunakan jika piksel melewati atau sama dengan nilai batas. Parameter ke-empat merupakan metode yang akan digunakan untuk melakukan *thresholding*.

### 4.3.3 Implementasi Operasi Morfologi Erosi

Proses morfolgi yang akan diterapkan pada citra adalah erosi. Erosi diterapkan dengan menggunakan fungsi OpenCV, yakni fungsi *erode* dan diimplementasikan pada Kode Sumber 4.4. Fungsi ini sendiri memiliki 3 parameter. Parameter pertama *img*, merupakan matriks citra masukan. Parameter kedua merupakan *kernel*. Kernel yang diterima hanya bisa berukuran ganjil, seperti 3x3, 5x5, 7x7 dan seterusnya. Selanjutnya, parameter ketiga merupakan *iterations*. *Iterations* merupakan banyaknya perulangan erosi yang akan diterapkan pada citra.

```
1.  erosion = cv2.erode(img,kernel,iterations = 1)
```

Kode Sumber 4.4 Fungsi morfologi erosi

### 4.3.4 Implementasi Operasi Morfologi Dilasi

Proses morfolgi yang akan diterapkan pada citra adalah dilasi. Dilasi diterapkan dengan menggunakan fungsi OpenCV, yakni fungsi *dilate* dan diimplementasikan pada Kode Sumber 4.5. Fungsi ini sendiri memiliki 3 parameter. Parameter pertama *img*, merupakan matriks citra masukan. Parameter kedua merupakan *kernel*. Kernel yang diterima hanya bisa berukuran ganjil, seperti 3x3, 5x5, 7x7 dan seterusnya. Selanjutnya, parameter ketiga merupakan *iterations*. *Iterations* merupakan banyaknya perulangan dilasi yang akan diterapkan pada citra.

```
1.  dilation = cv2.dilate(img,kernel,iterations = 1)
```

Kode Sumber 4.5 Fungsi morfologi dilasi

### 4.3.5 Implementasi Padding Citra

Proses *padding* citra diimplementasikan pada Kode Sumber 4.5. Proses dilakukan pada citra dengan memanfaatkan fungsi *copyMakeBorder* dari OpenCV. Dimana parameter yang digunakan adalah *img* sebagai matriks input citra yang akan

dilakukan *padding*, *top*, *bottom*, *left*, *right* yang merupakan panjang piksel untuk citra yang akan dilakukan *padding*. Lalu *borderType*, tipe *border* yang akan digunakan, serta *value*, nilai yang digunakan untuk mengisi border.

Dikarenakan tujuan penggunaan *padding* citra untuk membuat citra yang rasio ukurannya beragam menjadi rasio 1:1, maka sebelum *padding*, dilakukan beberapa hal terlebih dahulu seperti pada Kode Sumber 4.6.

```
1.  if hei>wid:
2.          widPadSize = hei-wid
3.          leftWidPadSize = int((hei-wid)/2)
4.          rightWidPadSize = widPadSize-leftWidPadSize
5.          newimg=cv2.copyMakeBorder(img,        top=0,
    bottom=0,                        left=leftWidPadSize,
    right=rightWidPadSize,                    borderType=
    cv2.BORDER_CONSTANT, value=[0,0,0] )
6.  elif wid>hei:
7.          widPadSize = wid-hei
8.          topWidPadSize = int((wid-hei)/2)
9.          bottomWidPadSize = widPadSize-topWidPadSize
10.         newimg=cv2.copyMakeBorder(img,
    top=topWidPadSize, bottom=bottomWidPadSize, left=0,
    right=0,      borderType=      cv2.BORDER_CONSTANT,
    value=[0,0,0] )
```

Kode Sumber 4.6 Fungsi *padding*

Dibandingkan tinggi dan lebar citra. Baris 1 merupakan kondisi jika tinggi (hei) citra nilainya lebih tinggi dibandingkan dengan lebar (wid) citra. Jika demikian, maka akan dicari panjang sisa yang dibutuhkan untuk membuat rasio citra menjadi 1:1, caranya dengan mengurangi nilai tinggi (hei) dengan nilai lebar (wid) dan disimpan pada variabel *widPadSize*. Ukuran *padding* sebelah kiri citra dihitung pada baris 4, yakni dengan mengurangi nilai tinggi (hei) dengan lebar (wid), lalu hasilnya dibagi 2 dan disimpan pada variabel *leftPadSize*. Baris 4 mencari ukuran *pad*

sebelah kanan citra dengan mengurangi *widPadSize* dengan *leftPadSize* dan disimpan pada variabel *rightPadSize*.

Baris 6 merupakan kondisi jika lebar (wid) citra nilainya lebih tinggi dibandingkan dengan tinggi (hei) citra. Jika demikian, maka akan dicari panjang sisa yang dibutuhkan untuk membuat rasio citra menjadi 1:1, caranya dengan mengurangi nilai lebar (wid) dengan nilai tinggi (hei) dan disimpan pada variabel *widPadSize*. Ukuran *padding* sebelah atas citra dihitung pada baris 8, yakni dengan mengurangi nilai lebar (wid) dengan tinggi (hei), lalu hasilnya dibagi 2 dan disimpan pada variabel *topPadSize*. Baris 9 mencari ukuran *pad* sebelah bawah citra dengan mengurangi *widPadSize* dengan *topPadSize* dan disimpan pada variabel *bottomPadSize*.

### 4.3.6 Implementasi Perubahan Ukuran Citra

Proses perubahan ukuran citra diimplementasikan pada Kode Sumber 4.7. Proses mengubah ukuran citra diimplementasikan dengan menggunakan fungsi *resize* dari OpenCV. Fungsi *resize* memiliki 2 parameter. Pertama *newImg* merupakan matriks citra masukan. Dan parameter kedua merupakan sebuah *tuple* ukuran hasil *resize* yang diinginkan.

```
1. newly = cv2.resize(newimg,(size,size))
```

Kode Sumber 4.7 Fungsi *resize*

## 4.4 Implementasi Lokalisasi Area Plat Nomor (YOLO)

Lokalisasi area plat nomor dilakukan dengan menggunakan YOLO. YOLO diimplementasikan dengan menggunakan darknet [26]. Implementasi YOLO dapat dilihat pada

```
1.  detections = darknet.detect_image(netMain,
    metaMain, darknet_image, thresh=0.25)
```

Kode Sumber 4.8 Fungsi *YOLO Darknet*

Fungsi detect_image dari darknet sendiri menerima 4 parameter. Parameter pertama merupakan *network* dari YOLO sendiri, di mana *weights* yang telah dibuat akan di-*load* untuk melakukan prediksi dari *frame* video. Parameter kedua merupakan *file* yang perlu untuk di-*load* untuk darknet, di mana *file* yang digunakan merupakan file .obj, yang berisi data teks informasi data latih *frame* dan data uji *frame*. Parameter ketiga merupakan citra dan parameter keempat merupakan *threshold*.

## 4.5 Implementasi Segmentasi Citra Plat

Proses segmentasi citra plat untuk mendapatkan calon karakter diimplementasikan pada Kode Sumber 4.9. Fungsi yang digunakan adalah fungsi findContours dari *library* OpenCV

```
1.  img1, contours, hierarchy =
    cv2.findContours(erode1
    ,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
2.  cou = 0
3.
4.  for element in contours:
5.      x,y,w,h = cv2.boundingRect(element)
6.      if h>w and w/w_imggray > 0.04 and
    w/w_imggray <=0.15 and h/h_imggray >= 0.29
    and h/h_imggray < 0.55:
7.          the_charas_candidate[x]=[y,[w,h]]
8.
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),
    2)
```

Kode Sumber 4.9 Fungsi *segmentasi citra*

Baris 1 merupakan pemanggilan fungsi findContours dari OpenCV, di mana, *parameter* dari fungsi ini ada 3. Yang pertama merupakan matriks citra masukan. Parameter kedua merupakan mode *contour retrieval*, di mana yang digunakan saat ini adalah cv2.RETR_TREE. Parameter ketiga merupakan metode aproksimasi kontur. Yang digunakan pada parameter ketiga merupakan cv2.CHAIN_APPROX_SIMPLE. Baris 4 – 8

merupakan iterasi dari variabel *contours*, di mana *contours* merupakan titik-titik koordinat yang didapatkan dari fungsi *findContours*. Lalu untuk memilih kandidat karakter, dipilih berdasarkan *threshold* tertentu. Di mana *threshold* ini dapat dilihat pada baris 6 pada statement *if*. Rasio-rasio ini didapatkan berdasarkan percobaan pada area plat yang telah didapatkan dan dilakukan *cropping* secara manual serta perhitungan langsung dari program dengan melihat semua kandidiat area tanpa syarat seperti pada Baris 6.

## 4.6 Implementasi Pembangunan Arsitektur

Pada subbab ini akan dijabarkan implementasi fungsi-fungsi pada tahap pembangunan model. Arsitektur CNN dimulai dengan mengimplementasi desain *convolution layer*. Implementasi tercantum pada Kode Sumber 4.10 dengan fungsi *convolution_layer* dengan layer-layer sesuai dengan spesifikasi yang dijelaskan pada Bab 3. Lalu pada tahap selanjutnya, hasil konvolusi akan ke *dropout layer* hingga akhirnya akan dimasukkan ke fungsi aktivasi *Softmax*. Pembangunan arsitektur memanfaatkan Keras dan Tensorflow.

```
1.  def convolution_layer(inputs, size):
2.      conv_layer = Conv2D(32, (3,3), strides=(1,1),
    kernel_initializer='he_normal')(inputs)
3.      conv_layer = BatchNormalization()(conv_layer)
4.
5.      conv_layer = Conv2D(32, (3,3), strides=(1,1),
    kernel_initializer='he_normal')(conv_layer)
6.      conv_layer = LeakyReLU()(conv_layer)
7.      conv_layer = BatchNormalization()(conv_layer)
8.      conv_layer = MaxPooling2D((2, 2))(conv_layer)
9.
10.     conv_layer = Conv2D(32, (2,2), strides=(1,1),
    kernel_initializer='he_normal')(conv_layer)
11.     conv_layer = LeakyReLU()(conv_layer)
12.     conv_layer = BatchNormalization()(conv_layer)
13.     conv_layer = MaxPooling2D((2, 2))(conv_layer)
14.     return conv_layer
```

Kode Sumber 4.10 Fungsi pembangunan *layer* konvolusi

Pada Baris 1, parameter *inputs* adalah *input layer* dan parameter *size* adalah dimensi data masukan yakni 32x32 (*image length, image width)*. Baris 2-14 masing-masing adalah penambahan *layer* yang dibutuhkan, seperti *Max Pooling* serta fungsi aktivasi *LeakyReLU*.

```
1.  def modelBuild():
2.      inputs = Input(shape=(32, 32, 1))
3.      conv_layer = convolution_layer(inputs, 32)
4.
5.      conv_layer = Dropout(0.5)(conv_layer)
6.
7.      flatten = Flatten()(conv_layer)
8.      outputs = Dense(36,
    activation='softmax')(flatten)
9.
10.     model = Model(inputs=inputs,
    outputs=outputs)
11.     optimizer = Adam(lr=0.0001,beta_1=0.9,
    beta_2=0.999, epsilon=None, decay=0.0,
    amsgrad=False)#beta_1=0.9, beta_2=0.999,
    epsilon=None, decay=0.0, amsgrad=False
12.     return model,optimizer
```

Kode Sumber 4.11 Fungsi pembangunan arsitektur CNN

Pada Kode Sumber 4.11, Baris 3 merupakan pemanggilan fungsi convolution_layer dengan parameter pertama *input* dari baris 2, dan parameter kedua merupakan size yang digunakan. Baris 5-11 secara berturut-turut adalah layer Dropout, lalu *Flatten*, yang digunakan untuk menjadikan *feature map* yang sebelumnya berbentuk matriks, menjadi vector. Selanjutnya, dilanjutkan dengan *Fully Connected Layer*, dan *optimizer* yang digunakan.

Arsitektur yang dibangun terdiri dari 6 macam *layer* antara lain Convolution2D, MaxPooling2D, BatchNormalization, Flatten, Dropout, dan Dense.

Convolution2D adalah implementasi *Convolution Layer* dalam bentuk 2 dimensi. Convolution2D menggunakan 6 parameter. Berikut penjelasan parameter-parameter tersebut:

1. Parameter *filters* adalah jumlah *filter* atau *kernel*.
2. Parameter *kernel_size* adalah ukuran *filter* atau *kernel*.
3. Parameter *strides* adalah ukuran *stride* yang digunakan.
4. Parameter *acitivation* adalah fungsi aktivasi apa yang digunakan, terdapat banyak pilihan fungsi aktivasi yang disediakan oleh Keras, salah satunya adalah 'relu' yakni fungsi ReLU.
5. Parameter *kernel_initializer* untuk menentukan weights awal matrix.
6. Parameter *padding* menentukan apakah *output* diaplikasikan *Zero Padding*. Jika *padding* bernilai '*valid*' maka *output* tidak diaplikaskan *Zero Padding* dan jika bernilai '*same'* maka bentuk keluaran akan diaplikasikan *Zero Padding* untuk menjaga keluaran sama dengan *input*. Nilai default *padding* jika tidak dicantumkan adalah *'valid'*.

MaxPooling2D adalah implementasi *Max Pooling Layer* dalam bentuk 2 dimensi. MaxPooling2D menggunakan 2 parameter. Berikut penjelasan parameter-parameter tersebut:

1. Parameter *pool_size* adalah ukuran *kernel Pooling Layer*.
2. Parameter *strides* adalah ukuran *stride* yang digunakan. Nilai default *strides* jika tidak dicantumkan adalah *'None'* yang berarti ukuran *stride* akan disamakan dengan *pool_size*.

Flatten biasa digunakan oleh arsitektur CNN pada Keras ketika *Convolution* dan *Max Pooling Layer* akan memasuki *Fully Connected Layer*. Hal ini karena Flatten memiliki fungsi untuk merubah keluaran sebelumnya yang berupa *feature map* menjadi *feature vector*, sehingga dapat dijadikan *input* pada *Fully Connected Layer*.

Dropout adalah implementasi teknik *dropout* yang berfungsi membuat beberapa *neuron* yang dipilih secara acak untuk tidak

dipakai selama proses pelatihan. Hal ini dilakukan untuk mengurangi *overfitting*. Dropout memiliki parameter *rate* yakni nilai probabilitas yang dipakai dalam menentukan secara acak neuron.

Dense adalah implementasi *Fully Connected Layer*. Dense menggunakan 2 parameter. Berikut penjelasan parameter-parameter tersebut:

1. Parameter *units* adalah jumlah neuron.
2. Parameter *activation* adalah fungsi aktivasi apa yang digunakan. Terdapat banyak pilihan fungsi aktivasi yang disediakan oleh Keras, ada 'relu' atau fungsi ReLU dan 'softmax' atau fungsi *softmax* yang biasa digunakan pada akhir arsitektur untuk mengklasifikasikan label kelas.

Pemanggilan fungsi pembangunan CNN diimplementasikan pada Kode Sumber 4.12.

```
1.  model = modelBuild()
```

Kode Sumber 4.12 Pemanggilan fungsi pembangunan arsitektur CNN

## 4.7 Implementasi Pelatihan dan Evaluasi CNN

Setelah pembangunan arsitektur *Convolutional Neural Network*, dilakukan proses pelatihan dan pengujian. Data akan dibagi menjadi 2, 70% data latih, dan 30% data uji, yaitu data latih dan data uji dengan jumlah data latih sebanyak 53169 citra dan jumlah data uji sebanyak 22763 citra. Implementasi pemisahan data dapat dilihat pada Kode Sumber 4.13.

```
1.  def       train_test_dataSplitting(current_dir_path,
    current_dir_path):
2.      random_index = []
3.      n = len(current_dir_files)
4.      n_limit = int(30/100*n)
```

```
5.      i=0
6.      while i<n_limit:
7.          x = randint(0,n-1)
8.          if x not in random_index and x != 0 and x !=
    264 and x != 527:
9.              random_index.append(x)
10.             i+=1
11.     for i in range(n):
12.         temp                                        =
    cv2.imread(current_dir_path+current_dir_files[i],0)
13.         if i in random_index:
14.             test_x.append(temp)
15.             test_y.append(directory_check)
16.         else:
17.             train_x.append(temp)
18.             train_y.append(directory_check)
```

Kode Sumber 4.13 Pembuatan index acak data uji

Baris 2 merupakan inisialisasi *list* untuk menampung *random index* yang akan digunakan untuk menjadi data latih. Baris 3 dan 4 secara berturut-turut merupakan banyaknya data yang disimpan pada variabel n dan *n_limit* merupakan Batasan banyaknya data latih, yakni sebesar 30% dari total data. Baris 7 sampai 10 merupakan proses mendapatkan *index* yang akan digunakan sebagai data latih, dan disimpan pada variabel *random_index*. Baris 11 sampai dengan 18. Merupakan pemisahan data latih dan data uji. Data latih disimpan pada variabel *train_x* dan data uji disimpan pada variabel *test_x.* Sedangkan *train_y* dan *test_y* merupakan label dari data secara berurutan. Kemudian, pemisahan dilakukan dengan melakukan iterasi dari i sampai dengan n, di mana, jika nilai i ada di dalam *list random_index*, maka data yang telah dibaca pada Baris 12, nantinya akan dimasukkan pada variabel *list test_x* dan labelnya pada *test_y*. Sedangkan jika nilai i tidak ada dalam *list random_index*, maka data yang telah dibaca pada Baris 12 akan masuk ke dalam *list train_x* dan labelnya *train_y*.

Proses pelatihan memanfaatkan data latih untuk membangun model CNN. Pelatihan menggunakan *loss function* berupa *Cross*

*Entropy* dan *optimizer* berupa Adam *Optimizer* dengan *learning rate* yang telah ditentukan sebelumnya. Hal ini diimplementasikan pada Kode Sumber 4.14 di baris 1 dan 2. Proses pelatihan akan dijalankan pada *batch size* 50 dan jumlah *epoch* 200. Dalam setiap akhir *epoch* terdapat proses pengujian model terhadap data uji dan didapatkan nilai akurasi, *precision* dan *recall*. Pada baris 3 dilakukan proses pelatihan dengan memanggil fungsi *fit* dari Keras. Dimana *batch_size* adalah ukuran *batch*, *epochs* adalah jumlah *epoch* yang akan dijalankan, *validation_data* adalah data uji dan label kelasnya.

```
1.  optimizer = Adam(lr=0.0001,beta_1=0.9,
    beta_2=0.999, epsilon=None, decay=0.0,
    amsgrad=False)
2.  model.compile(optimizer=optimizer,
    loss='categorical_crossentropy',
    metrics=['accuracy'])
3.  model.fit(train_x, train_y, batch_size=50,
    epochs=1000, verbose=1, callbacks=[time_callback],
    validation_data=(test_x, test_y))
```

Kode Sumber 4.14 Pelatihan CNN

Setelah proses pelatihan selesai, model akan diuji dengan menggunakan data uji. Hal ini diimplementasikan pada Kode Sumber 4.15. Fungsi *evaluate* akan mengevaluasi model dengan data uji dan labelnya.

```
1.  print(model.evaluate(test_x, test_y))
```

Kode Sumber 4.15 Pengujian CNN

Pada fungsi *evaluate*, didapatkan nilai akurasi model terhadap data uji. Namun library Keras ini tidak memiliki fitur untuk menghitung *precision* dan *recall*. Sehingga perlu dilakukan cara alternatif untuk melakukan evaluasi terhadap model agar mendapatkan nilai *precision* dan *recall*. Dengan memanfaatkan fungsi *predict_classes* dari *library* Keras terhadap *test_x* (data uji).

Hal ini diimplementasikan pada Kode Sumber 4.16. Pada baris 7, fungsi *predict_classes* digunakan untuk mendapatkan nilai prediksi untuk masing-masing kelas dan disimpan pada variabel *pred_y*. Pada baris 1-5, dibuat sebuah fungsi *return_to_label* untuk mengembalikan bentuk data prediksi dari Keras yang masih berupa tipe kelas ketegorikal menjadi tipe kelas yang numerik. Lalu pada baris 9-10, fungsi *return_to_label* digunakan untuk mengembalikan *pred_y* dan *test_y* ke tipe kelas numerik. Baris 12 memanfaatkan fungsi *classification_report* dari *library* Scikit-learn untuk mengevaluasi nilai prediksi (*pred*) terhadap nilai sesungguhnya (*test*) untuk masing-masing kelas sehingga didapatkan nilai *precision* dan *recall*.

```
1.  def return_to_label(y):
2.      label = []
3.      for i in range(len(y)):
4.          label.append(np.argmax(y[i]))
5.      return label
6.
7.  pred_y = model.predict(test_x)
8.
9.  test = return_to_label(test_y)
10. pred = return_to_label(pred_y)
11.
12. print(classification_report(test, pred))
```

Kode Sumber 4.16 Evaluasi *precision* dan *recall*

# BAB V
# UJI COBA DAN EVALUASI

Bab ini akan membahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

## 5.1 Lingkungan Uji Coba

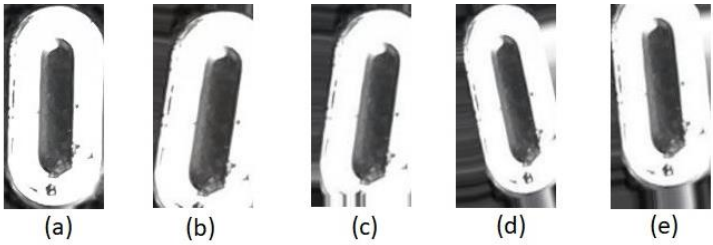
Lingkungan uji coba pada tugas akhir ini adalah sebuah desktop *personal computer* (PC) MS-7886. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi perangkat keras AMD Ryzen 7 2700 dengan kecepatan 3,2 GHz, *Random Access Memory* (RAM) sebesar 16 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA GeForce RTX 2080 sebesar 8 GB. Pada sisi perangkat lunak, uji coba pada tugas akhir ini dilakukan dengan menggunakan bahasa pemograman Python 3.6 dilengkapi dengan *library* antara lain Keras, Tensorflow, OpenCV, Numpy, Matplotlib dan Scikit-learn.

## 5.2 Dataset

Pada tugas akhir ini, data yang digunakan adalah data plat nomor polisi kendaraan yang diambil secara mandiri. Kumpulan citra ini diambil dengan bermacam variasi sudut pandang. *Dataset* yang dibutuhkan sendiri merupakan dataset karakter, maka citra-citra plat nomor polisi ini kemudian dilakukan *cropping* untuk mendapatkan karakter-karakternya, yakni karakter A sampai dengan Z kemudian angka 0 sampai dengan 9. Citra-citra karakter ini berjumlah 280 citra sampai dengan 360 citra di mana terdapat sekitar 8 sampai dengan 11 citra untuk setiap karakternya. Setelah itu, dilakukan augmentasi citra untuk mendapatkan variasi-variasi karakternya, mulai dari rotasi, hingga perbesaran, di mana semau variasi ini yang nanti akan dimasukkan ke dalam *Convolutional Neural Network.*

## 5.3 Augmentasi Citra

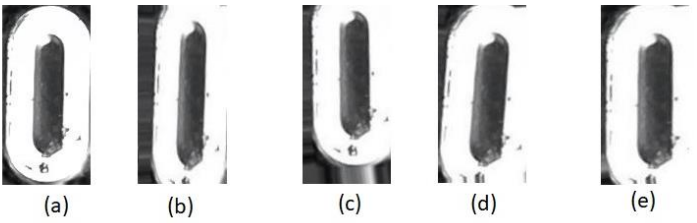
Augmentasi citra dilakukan untuk mendapatkan lebih banyak variasi citra untuk dimasukkan ke dalam *network* CNN yang telah dibuat. Augmentasi yang dilakukan terhadap *dataset* awal adalah rotasi dan perbesaran. Rotasi dilakukan terhadap citra dengan sudut tertentu. Sudut yang dipakai adalah sebesar 10 derajat. Sudut ini digunakan karena rotasi yang dihasilkan nantinya akan ada sedikit *noise* seperti pada Gambar 5.1. Sudut 10 derajat digunakan untuk menghindari hasil augmentasi yang terlalu merusak kumpulan data pada suatu kelas tertentu dan membuat satu data pada kelas tersebut mirip (menimpa) kelas lain sehingga dapat mengganggu hasil klasifikasi citra.




Gambar 5.1 Hasil Augmentasi Rotasi Citra. (a) Citra Asli, (b) Citra Rotasi 20 derajat kanan, (c) Citra Rotasi 30 derajat ke kanan, (d) Citra Rotasi 20 derajat ke kiri, (e) Citra Rotasi 10 derajat ke kiri

Selanjutnya, juga dilakukan proses perbesaran objek. Pembesaran objek juga dilakukan untuk memberikan variasi data pada *network* CNN yang dibangun. Pembesaran objek ini diterapkan dengan menganggap adanya kemungkinan jika citra karakter yang nantinya didapatkan pada uji coba tidak selalu dalam kondisi yang terbaik. Ada kemungkinan bentuk karakter pada data itu ada bagian yang tidak terlihat. Selanjutnya, untuk perbesaran objek, digunakan 0.1 karena jika terlalu besar, perbesaran bisa membuat citra sangat rusak dan membuat *network* CNN lebih

susah mengenali pola. Citra asli dan citra setelah dilakukan perbesaran dapat dilihat pada Gambar 5.2




Gambar 5.2 Hasil Augmentasi Pembesaran Citra, (a) Citra Asli, (b) Citra Pembesaran 0.1, (c) Citra Pembesaran 0.1, (d) Citra Pembesaran 0.2, € Citra Pembesaran 0.2

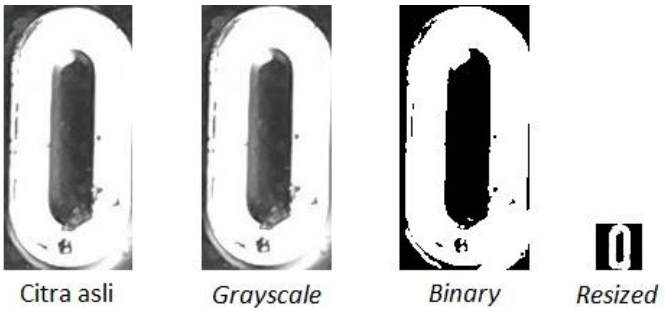
## 5.4 Hasil Praproses

Sebelum memasuki proses pelatihan dengan arsitektur CNN yang telah dirancang sebelumnya, akan dilakukan praproses citra. Terdapat perbedaan antara praproses citra yang dilakukan saat pelatihan dan pengujian. Perbedaan ini terletak pada tujuan praproses citra itu sendiri. Pada proses pelatihan, data citra merupakan data-data citra tunggal yang akan menjadi data latih untuk *network* CNN, sedangkan pada proses pengujian, data citra merupakan data citra plat nomor polisi kendaraan yang utuh. Sehingga pada tahap pengujian, tujuan praproses citra adalah untuk memodifikasi citra sehingga mempermudah proses segmentasi karakter yang akan diprediksi kelasnya.

### 5.4.1 Praproses Pelatihan

Pada tahap praproses untuk pelatihan, citra awalnya dirubah kanalnya, yang awalnya citra BGR (citra 3 kanal) dirubah menjadi citra *grayscale* (citra 1 kanal). Setelah dilakukan perubahan kanal citra, citra kemudian dilakukan Otsu *Thresholding.* Kemudian citra hasil *thresholding* ini nantinya akan diterapkan *padding* untuk membuat rasio citra yang sebelumnya variatif menjadi 1:1. Setelah dilakukan *padding* maka citra akan di-*resize* menjadi ukuran

32x32, menyesuaikan dengan *input* CNN. Citra hasil dapat dilihat pada Gambar 5.3.




Gambar 5.3 Citra asli, citra setelah dirubah kanalnya, citra biner, serta citra yang telah dilakukan *padding* dan *resize*

### 5.4.2 Praproses Pengujian

Pada tahap praproses untuk pengujian, citra merupakan citra plat nomor yang didapatkan setelah melakukan *object detection* menggunakan YOLO (*You Only Look Once*). Citra plat nomor yang telah didapatkan dirubah kanalnya, yang awalnya citra BGR (citra 3 kanal) dirubah menjadi citra *grayscale* (citra 1 kanal). Setelah dilakukan perubahan kanal citra, citra kemudian dilakukan Otsu *Thresholding.* Hasil *thresholding* dapat dilihat pada Gambar 5.4.

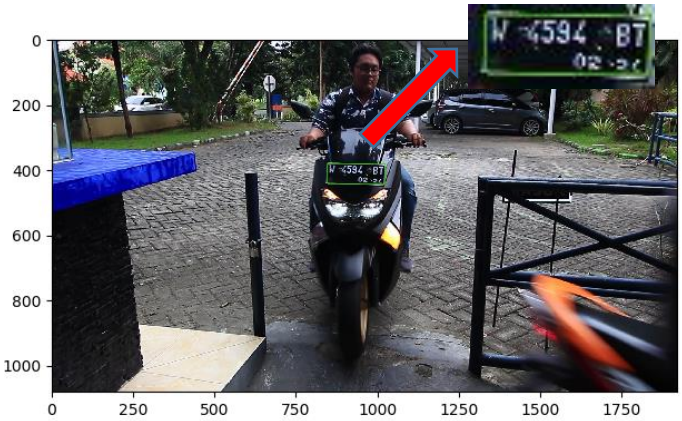

Gambar 5.4 Citra asli, citra grayscale dan citra biner


Gambar 5.5 Citra biner setelah dierosi, setelah dilasi dua kali dan setelah dierosi kembali

Kemudian, dilakukan operasi morfologi, di mana operasi morfologi ini meliputi erosi, dua kali dilasi, kemudian kembali di erosi. Hal ini dilakukan karena ada kemungkinan karakter-karakter dalam citra plat nomor menyatu sehingga menjadi lebih susah untuk dilakukan segmentasi karakter yang nantinya akan dimasukkan ke dalam *network* CNN. Hasil morfologi dapat dilihat pada Gambar 5.5. Barulah citra hasil morfologi dilakukan segmentasi untuk mendapatkan karakter-karakter dari citra plat. Karakter-karakter itu kemudian dilakukan *padding* dan kemudian di-*resize* menjadi ukuran 32x32 seperti pada Gambar 5.3.

## 5.5 Hasil Lokalisasi Area Plat Nomor

Lokalisasi area plat nomor dilakukan dengan menggunakan YOLO. *Frame* yang telah diambil dari data video nantinya akan di-*resize* menjadi ukuran 416x416 sesuai dengan *input* yang dibutuhkan oleh arsitektur YOLO. *Frame* kemudian dimasukkan ke dalam *network* dan keluarannya merupakan sebuah *list* koordinat hasil deteksi plat, probabilitas kelas, serta nama kelas, di mana pada Tugas Akhir ini, jumlah kelas yang digunakan pada YOLO hanya sejumlah satu kelas, yakni kelas Plate. Koordinat hasil deteksi ini merupakan koordinat dari citra masukan setelah dilakukan *resize*, yakni *frame* yang telah menjadi ukuran 416x416, di mana koordinat ini perlu dirubah menjadi koordinat asli dari citra sebelum *resize*, citra ukuran asli, yakni citra ukuran 1920x1080 sebelum koordinat-koordinat ini nantinya digunakan untuk menggambarkan *bounding box* area plat pada *frame* awal sebelum dilakukan *resize* masuk ke dalam *network* YOLO. Hasil lokalisasi dapat dilihat pada Gambar 5.6.




Gambar 5.6 Hasil lokalisasi YOLO

## 5.6 Skenario Uji Coba

Proses uji coba berguna untuk menemukan parameter-parameter yang menghasilkan performa model yang paling optimal. Parameter yang tepat akan memberikan hasil yang lebih baik pada saat proses uji coba. Hasil terbaik dari suatu skenario uji coba akan digunakan untuk skenario uji coba berikutnya. Ada 4 macam skenario uji coba dan semuanya akan dicoba pada arsitektur CNN yang telah dirancang. Skenario uji coba yang akan dilakukan yaitu:

1. Uji Coba Parameter CNN
2. Uji Coba K-Folds *Cross Validation*
3. Uji Coba Perbandingan Hasil Lokalisasi SIFT dan YOLO
4. Uji Coba pada Data Video

Tabel 5.1 berisi parameter-parameter awal arsitektur CNN yang digunakan dan dapat berubah di setiap uji coba yang dilakukan. Pada setiap skenario uji coba akan ditetapkan nilai parameter yang dapat meningkatkan kinerja arsitektur.

Tabel 5.1 Parameter awal yang digunakan dalam arsitektur

| **Keterangan** | **Parameter** |
|---|---|
| Jumlah *epoch* | 200 |
| Ukuran *batch* | 50 |
| *Optimizer* | Adam *Optimizer* |
| *Learning Rate* | 0,0001 |
| *Loss function* | *Categorical Cross Entropy* |
| *Kernel Convolution 1* | 3x3 |
| *Kernel Convolution 2* | 3x3 |
| *Kernel Convolution 3* | 2x2 |

### 5.6.1 Uji Coba Parameter CNN

Uji coba penggantian parameter CNN digunakan untuk mengetahui parameter mana saja yang menghasilkan performa model terbaik. Parameter CNN yang akan dicoba untuk divariasikan antara lain:

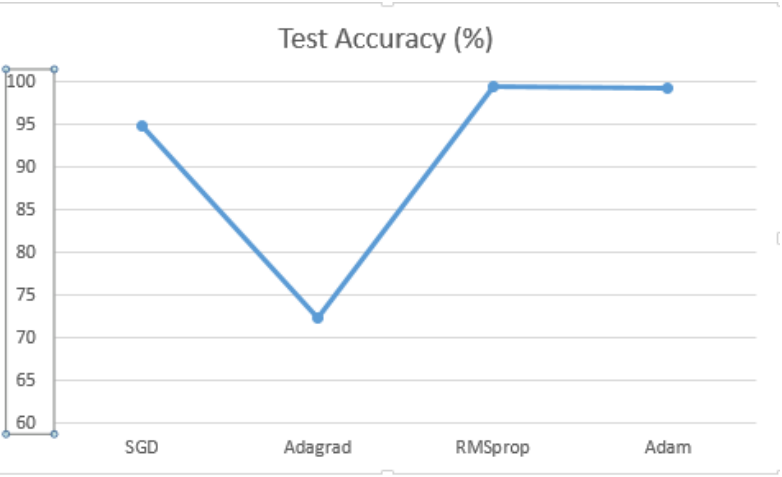
1. *Optimizer*

Uji coba pertama adalah uji coba *Optimizer* pada model yang akan dibuat. *Optimizer* yang akan digunakan pada uji coba ini antara lain SGD, Adagrad, RMSprop, dan Adam. Percobaan dilakukan pada *learning rate* 0.0001.

2. *Kernel* pada *Convolution Layer*

Uji coba *kernel* pada *Convolution Layer* dicobakan pada setiap *kernel Convolution Layer* diubah seperti pada Tabel 5.3 di mana setiap ukuran pada salah satu *kernel Convolution Layer* diubah, *kernel size* pada *Convolution Layer* yang lain dibiarkan sesuai dengan parameter awal seperti pada Tabel 5.1

Uji coba penggantian *optimizer* pada arsitektur CNN menghasilkan grafik akurasi pengujian yang dapat dilihat pada Gambar 5.7 dan perbandingan nilai akurasi, *weighted average precision*, *weighted average recall* pengujian serta lama waktu pelatihan dapat dilihat pada Tabel 5.2.




Gambar 5.7 Perbedaan akurasi pada arsitektur CNN dalam uji coba penggantian *optimizer*

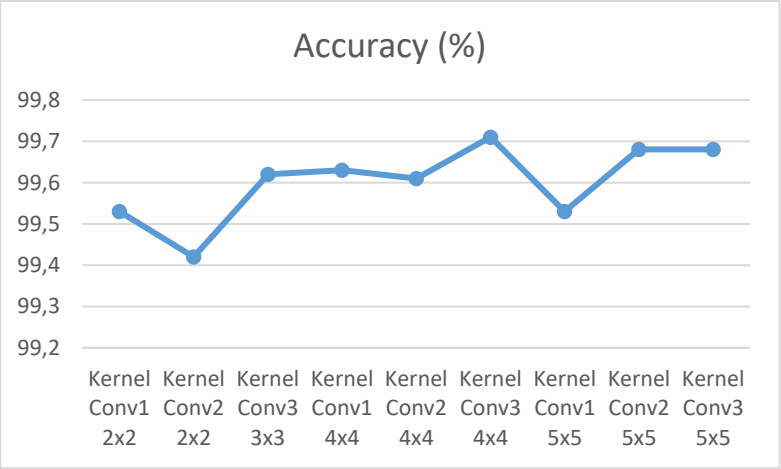
Tabel 5.2 Perbandingan lama waktu pelatihan dan akurasi terbaik, *weighted average precision*, *weighted average recall* arsitektur CNN pada uji coba penggantian parameter CNN

| **Optimizer** | **Lama waktu pelatihan** | **Akurasi** | ***Weighted Average Precision*** | ***Weighted Average Recall*** |
|---|---|---|---|---|
| SGD | 1595 detik | 94.90% | 95% | 95% |
| Adagrad | 1653 detik | 72.32% | 73% | 72% |
| **Adam** | **1821 detik** | **99.52%** | **100%** | **100%** |
| RMSprop | 1711 detik | 99.35% | 99% | 99% |

Pada pengujian optimizer, didapatkan *optimizer* Adam merupakan *optimizer* optimal. Maka, pengujian ukuran *kernel* dilakukan dengan menggunakan *optimizer* Adam.

Pada uji coba *Kernel* pada *Convolution Layer*, dilakukan 9 kali percobaan pada ketiga *layer* konvolusi. Percobaan pertama adalah dengan mengganti ukuran *kernel* pada *layer* konvolusi pertama menjadi ukuran 2x2. Begitu pula untuk percobaan kedua, yang diaplikasikan pada *layer* konvolusi kedua dengan ukuran 2x2. Sedangkan pada percobaan ketiga, *kernel* pada *layer* konvolusi diubah menjadi ukuran 3x3. Sedangkan pada percobaan keempat, kelima, dan keenam, secara bertutut *kernel* pada *layer* konvolusi pertama, kedua, dan ketiga diubah ukurannya menjadi 4x4. Sedangkan pada percobaan ketujuh, kedelapan, dan kesembilan secara berutut-berturut ukuran ukuran *kernel* diubah menjadi 5x5.

Uji coba penggantian ukuran *kernel* pada *layer* konvolusi pada arsitektur CNN menghasilkan grafik akurasi pengujian pada semua model yang dapat dilihat pada Gambar 5.8. Sedangkan perbandingan akurasi, *loss*, dan waktu pelatihan masing-masing model dapat dilihat pada Tabel 5.3.




Gambar 5.8 Grafik perbandingan akurasi model berdasarkan perubahan ukuran *kernel* pada *layer* konvolusi

Tabel 5.3 Tabel perbandingan akurasi, *loss*, serta waktu pelatihan model berdasarkan perubahan ukuran *kernel layer* konvolusi.

| ***Layer* konvolusi dan ukuran *kernel*** | **Waktu** | **Akurasi** | ***Loss*** |
|---|---|---|---|
| Conv1 2x2 | 2150 detik | 99,53% | 0,0175 |
| Conv2 2x2 | 2039 detik | 99,42% | 0,0207 |
| Conv3 3x3 | 2065 detik | 99,62% | 0,0128 |
| Conv1 4x4 | 2012 detik | 99,63% | 0,0117 |
| Conv2 4x4 | 2219 detik | 99,61% | 0,0133 |
| **Conv3 4x4** | **2062 detik** | **99,71%** | **0,0091** |
| Conv1 5x5 | 1945 detik | 99,53% | 0,0145 |
| Conv2 5x5 | 2430 detik | 99,68% | 0,0112 |
| Conv3 5x5 | 2136 detik | 99,68% | 0,0119 |

Pada hasil uji coba, didapatkan bahwa perubahan ukuran *kernel layer* konvolusi ketiga menjadi 4x4 menghasilkan akurasi yang tertinggi dibandingkan percobaan yang lainnya.

### 5.6.2 Uji Coba K-Folds *Cross Validation*

Pada uji coba sebelumnya, didapatkan model terbaik berasal dari model yang menggunakan *optimizer* Adam dengan *kernel layer* konvolusi ketiga 4x4. Maka, pada percobaan *K-Folds Cross Validation*, akan dilakukan dengan menggunakan arsitektur yang optimal yang menggunakan *optimizer* Adam dan ukuran *kernel layer* konvolusi ketiga 4x4.

Uji coba dilakukan dengan menggunakan *k* bernilai 4 sampai dengan *k* bernilai 10. Nilai *k* akan membuat data dibagi sebanyak *k batch* secara rata. Hasil ujicoba dapat dilihat pada Tabel 5.4.

Tabel 5.4 Tabel perbandingan rata-rata akurasi model berdasarkan nilai *k* pada *K-Folds Cross Validation*

| **Nilai *k*** | **Akurasi Model** |
|---|---|
| 4 | 99,64% |
| 5 | 99,68% |
| 6 | 99,70% |
| 7 | 99,69% |
| 8 | 99,71% |
| 9 | 99,69% |
| **10** | **99,73%** |

Didapatkan akurasi terbaik dengan nilai *k* berjumlah 10 dengan akurasi sebesar 99,73%.

### 5.6.3 Uji Coba Perbandingan Hasil Lokalisasi SIFT+Tracking dan YOLO

Uji coba metode lokalisasi dilakukan untuk menentukan seberapa baik performa lokalisasi area plat nomor kendaraan. Uji coba metode lokalisasi dilakukan dengan membandingkan 2 metode. Yang pertama adalah SIFT dengan *tracking*, di mana pada SIFT, dicari *keypoints* dari citra refensi dan citra pembanding. Sedangkan metode yang kedua adalah dengan menggunakan YOLO, di mana YOLO (*You Only Look Once*) merupakan sebuah arsitektur *deep learning*, di mana citra masukkan merupakan citra utuh dari *frame* video.

Berdasarkan data uji coba pada Tabel 5.5, didapatkan bahwa akurasi lokalisasi dari YOLO lebih baik dibandingkan dengan SIFT + Tracking. Di mana akurasi lokalisasi yang didapatkan dari YOLO adalah sebesar 100%. Sedangkan akurasi lokalisasi dari SIFT + Tracking sendiri adalah sebesar 35,17%. Kedua metode dilakukan pada total 163 *frame*.

Tabel 5.5 Perbandingan Performa Lokalisasi antara SIFT dan Tracking dengan YOLO

| Video | Banyak frame | SIFT + Tracking | YOLO |
|---|---|---|---|
| 1 | 52 | 44,24% | 100,00% |
| **2** | **54** | **61,11%** | **100,00%** |
| 3 | 57 | 0% | 100,00% |
| Rata-rata | | 35,17% | 100% |

### 5.6.4 Hasil Uji Coba pada Data Video

Uji coba pada data video dilakukan pada 3 data video untuk mengukur performa model yang telah dibuat. Data video yang digunakan merupakan data video kendaraan motor yang akan dikenali karakter-karakter dari plat nomornya.

Untuk melakukan evaluasi performa, akan dilakukan pada area plat pada karakter yang terdeteksi. Kemudian akan dipilih dua

frame terbaik dari ketiga video dan diukur akurasi pengenalan karakter dari keduanya. Hasil uji coba dapat dilihat pada Tabel 5.6

Tabel 5.6 Perbandingan performa model yang telah dicoba

| No | Video | Akurasi Segmentasi | Akurasi Karakter |
|---|---|---|---|
| 1 | Video 1 | 56,92% | 84,18% |
| **2** | **Video 2** | **80,13%** | **98,98%** |
| 3 | Video 3 | 75,82% | 85,44% |
| Rata-rata | | 70,95% | 89,53% |

Akurasi segmentasi dan akurasi pengenalan karakter didapatkan dengan cara membandingkan seacar visual, label prediksi dengan citra itu sendiri, benar atau salah. Metode ini dilakukan karena setiap kandidat karakter yang didapatkan tidak mempunya *ground truth* untuk membandingkan apakah segmentasi yang dilakukan sudah benar atau tidak ataupun prediksi karakter yang didapatkan sudah benar atau tidak.

Pengambilan dilakukan dengan memberikan area pencarian yang telah dedefinisikan, di mana area ini merupakan area tengah ke bawah *frame*. Area ini dipilih karena area ini lebih dekat dengan kamera pengambilan yang nantinya akan dapat memengaruhi hasil segmentasi karakter.

Didapatkan hasil evaluasi akurasi terbaik dan segmentasi terbaik terdapat pada video ke 2. Di mana didapatkan akurasi segmentasi video 2 sebesar 80,13%, dan pada akurasi pengenalan karakter pada video 3, sebesar 98,98%.

## 5.7 Hasil dan Evaluasi

Pada uji coba penggantian parameter CNN, diperoleh hasil akurasi yang paling baik pada penggunaan Adam *optimizer* dengan akurasi sebesar 99.52%. Adam *optimizer* tepat digunakan untuk arsitektur CNN yang telah dibangun karena menggunakan *adaptive learning rate* yang cenderung lebih cepat konvergen serta

menghasilkan performa yang lebih baik dibandingkan dengan *optimizer* yang menggunakan *learning rate* statis.

Selanjutnya, dilakukan uji coba pada ukuran *kernel* pada *layer* konvolusi. Didapatkan hasil akurasi terbesar dengan merubah ukuran *kernel layer konvolusi* ketiga menjadi 4x4. Akurasi model yang didapatkan sebesar 99,71%.

Pada uji coba K-Folds Cross Validation, akurasi yang hampir sama didapatkan untuk nilai *k* yang digunakan, yaitu 99.64% untuk nilai *k* 4, 99,68% untuk nilai *k* 5, 99,70% untuk nilai *k* 6, 99,69% untuk nilai *k* 7, 99,71% untuk nilai *k* 8, 99,69% untuk nilai *k* 9, dan 99.73% untuk nilai *k 10*. Variasi nilai *k* yang digunakan tidak memperlihatkan akurasi yang jauh berbeda. Akan tetapi, dipilih *k* dengan nilai 10 karena, dari segi akurasi yang lebih besar, nilai *k* lebih besar juga dapat berarti bias yang lebih kecil terhadap kelebihan estimasi terhadap *true expected error*. Meskipun demikian, dengan nilai *k* yang lebih besar, maka waktu komputasi dapat dikatakan lebih lama pula, karena jumlah data latih akan lebih banyak pada tiap *batch*-nya dibandingkan dengan jumlah data latih pada *batch* data yang menggunakan nilai *k* yang lebih kecil. Dalam kasus ini, *k* bernilai 10 memiliki data latih kurang lebih 68356 data latih, sedangkan pada *k* bernilai 4 yang merupakan uji coba dengan nilai *k* terkecil, data latih pada *batch* ini sebesar kurang lebih 56940 data.

Pada uji coba pada data video, didapatkan akurasi segementasi dan akurasi pengenalan karakter terbaik berasal dari video 2. Pertama, video 2 mendapat akurasi segmentasi terbaik disebabkan oleh lokalisasi plat yang terambil pada jarak terdekat lebih banyak dibandingkan pada lokalisasi plat yang jaraknya lebih jauh dari kamera, sehingga mempengaruhi performa segmentasi pada area plat yang telah didapatkan. Hal lainnya juga disebabkan oleh data citra plat nomor sudah baik yang secara visual memang

bisa dikenali, sehingga, hasil segmentasi juga lebih baik, seperti pada Gambar 5.9.


Gambar 5.9 Hasil pembagian berdasarkan area plat nomor dan hasil segmentasi karakter pada Data Video 3

Sedangkan pada Gambar 5.10, secara visual, karakter pada area plat tidak terlalu terlihat. Ada juga bagian-bagian yang menyatu, meski pada implementasi telah dilakukan beberapa proses morfologi, akan tetapi, detail citra tetap tidak terlihat. Sehingga hasil prediksi bisa salah. Selain itu, ini terjadi juga karena ketika lokalisasi menggunakan YOLO, plat sudah didapatkan ketika kendaraan dalam kondisi jauh di depan kamera, sehingga citra plat yang didapatkan masih berukuran kecil dan secara visual masih belum terlihat. Hal lain yang menyebabkan hal ini juga karena hasil segmentasi citra asli ukurannya lebih kecil dibandingkan dengan citra masukan ke dalam model CNN, di mana ukuran masukan dalam model CNN yang telah dibuat adalah 32x32. Karena ukuran citra karakter hasil segmentasi lebih kecil, maka saat melakukan *resize*, citra akan dipaksa diperbesar, di mana hal ini tidak baik dilakukan karena dapat merusak citra itu sendiri seperti pada Gambar 5.10 pada karakter A di maana hasil karakter yang diperbesar membuat citra karakter itu kehilangan bentuk aslinya.


Gambar 5.10 Hasil pembagian area plat nomor dan hasil segmentasi karakter pada Data Video 2

Pada beberapa karakter, seperti angka 8 ataupun pada huruf B, terjadi beberapa kali misklasifikasi. Misklasifikasi ini terjadi karena pada dataset, variasi karakter yang dilakukan dari augmentasi hanya melakukan variasi rotasi dan zoom, sedangkan misklasifikasi yang terjadi pada huruf B maupun pada angka 8 menjadi huruf O terjadi karena secara visual, *font* angka 8 ini mirip dengan *font* huruf O pada dataset, seperti terlihat pada Gambar 5.11. Dapat dilihat, secara visual, karakter B pada dataset secara *font* lebih tipis dibandingkan karakter B pada data uji dan membuat karkater B pada data uji secara visual lebih mirip dengan huruf O.


Gambar 5.11 (a) Karakter B pada data uji, (b) karakter B pada dataset, dan (c) karakter O pada dataset

Dari keempat hasil uji coba yang telah dilakukan, pada Tabel 5.4 ditetapkan parameter optimal dari seluruh uji coba tersebut.

Tabel 5.7 Parameter optimal yang ditetapkan

| **Keterangan** | **Parameter optimal** |
|---|---|
| Arsitektur CNN *Optimizer* | Adam *optimizer* |
| Ukuran *Kernel Layer* Konvolusi | 4x4 pada *Layer* Konvolusi ketiga |
| K-Folds Cross Validation | $K = 10$ |
| Metode lokalisasi | YOLO |

# BAB VI
# KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa depan.

## 6.1 Kesimpulan

Dalam pengerjaan Tugas Akhir ini setelah melalui tahap perancangan aplikasi, implementasi metode, serta uji coba, diperoleh kesimpulan sebagai berikut:

1. Berdasarkan uji coba parameter pada arsitektur CNN yang digunakan. Model yang dibangun menghasilkan akurasi yang baik yaitu sebesar 97% untuk Optimizer Adam, ukuran kernel pada *Convolution Layer* ketiga 4x4.
2. Lokalisasi terbaik didapatkan dengan menggunakan YOLO dibandingkan SIFT sebagai metode lokalisasi. Terbukti dengan akurasi rata-rata lokalisasi YOLO sebesar 100% dibandingkan dengan akurasi lokalisasi SIFT + Tracking yaitu sebesar 35,17%.
3. Metode segmentasi karakter sudah cukup berhasil melakukan segmentasi pada area plat nomor yang terdeteksi, dengan akurasi rata-rata segmentasi mencapai 70,95%.
4. Sistem pengenalan nomor polisi kendaraan telah berhasil diimplementasikan dengan akurasi model tertinggi 99,71%, serta akurasi pengujian rata-rata 89,53% yang didapatkan dari uji coba Adam *optimizer* dengan *learning rate* 0,0001.

## 6.2 Saran

Saran yang diberikan untuk pengembangan sistem pengenalan nomor polisi kendaraan menggunakan *Convolutional Neural Network* pada data video, yaitu:

1. Pengembangan sistem yang dapat melakukan segmentasi karakter lebih akurat.
2. Menambah variasi *font* pada karakter untuk memperbanyak variasi karakter yang ada.
3. Melakukan eksplorasi parameter selain *optimizer* dan *learning rate* yang dapat menambah performa arsitektur seperti *activation function*, jumlah dan *stride filter* konvolusi, ukuran *max pooling*.

# DAFTAR PUSTAKA


[1] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos dan E. Kayafas, "A license plate-recognition algorithm for intelligent transportation system applications," *IEEE Trans. Intell. Transp. Syst., vol. 7, no. 3, pp.*, p. 377–391, 2006..

[2] S. Du, M. Ibrahim, M. Shehata dan W. badawy, "Automatic License Plate Recognition (ALPR): A State of the Art Review," *IEEE Transactions on Circuits and Systems for Video Technology* , no. 23(2), p. 311–325, 2013.

[3] S. Fadillah, "Penerapan Pengolahan Citra menggunakan Metode Deep Learning untuk Mendeteksi Kecacatan Permukaan Buah Manggis," Yogyakarta, 2017.

[4] D. Hubel dan T. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *Journal of Physiology*, vol. 195, p. 215–243, 1968.

[5] M. Zufar dan B. Setiyono, "Convolutional Neural Networks untuk Pengenalan Wajah Secara Real-Time," *Jurnal Sains dan Seni ITS,* vol. 5, pp. 2337-3520, 2016.

[6] "Convolutional Neural Network," MathWorks, [Online]. Available: https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html. [Diakses 29 November 2018].

[7] "Deep learning for complete beginners: convolutional neural networks with keras," Cambridgespark, 20 March 2017. [Online]. Available: https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html. [Diakses 29 November 2018].

[8] "An Intuitive Explanation of Convolutional Neural Networks," Ujjwalkarn, 11 August 2016. [Online].

Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets. [Diakses 29 November 2018].

[9] I. W. Suartika, A. Y. Wijaya dan R. Soelaiman, "Klasifikasi Citra Menggunakan Convolutional pada Caltech 101," *JURNAL TEKNIK ITS*, vol. 5, 2016.

[10 S. Sena, "Pengenalan Deep Learning Neural Network," 28 October 207. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac. [Diakses 30 November 2018].

[11 G. Hinton, *Neural Networks for Machine Learning.*

[12 S. Ruder, "Ruder.io," 19 January 2016. [Online]. Available: http://ruder.io/optimizing-gradient-descent/index.html#rmsprop. [Diakses 23 December 2018].

[13 A. Budhiraja, "Dropout in (Deep) Machine Learning," [Online]. Available: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5. [Diakses 11 12 2018].

[14 A. Karpathy, "Convolutional Neural Networks for Visual Recognition," Stanford University, [Online]. Available: http://cs231n.github.io/. [Diakses 30 November 2018].

[15 J. Redmon, S. Divvala, R. Girshick dan A. Farhadi, "You Only Look Once:Unified, Real-Time Object Detection".

[16 C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke dan A. Rabinovich, "Going Deeper with Convolutions," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17 N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” *IEEE Transactions on Systems, Man, and Cybernetics,* no. 9(1), pp. 62-66, 1979.

[18 D. G. Lowe, “Object Recognition from Local Scale-Invariant Features,” *Proceedings of the Seventh IEEE International Conference on Computer Vision,* 1999.

[19 “About Python,” Python, [Online]. Available: https://www.python.org/about/. [Diakses 30 November 2018].

[20 “Keras: The Python Deep Learning library,” Keras, [Online]. Available: https://keras.io/. [Diakses 30 November 2018].

[21 “TensorFlow,” TensorFlow, [Online]. Available: https://www.tensorflow.org/. [Diakses 30 November 2018].

[22 “OpenCV,” [Online]. Available: https://opencv.org/. [Diakses 30 November 2018].

[23 “NumPy,” NumPy, [Online]. Available: http://www.numpy.org/. [Diakses 30 November 2018].

[24 “Scikit-learn,” Scikit-learn, [Online]. Available: http://scikit-learn.org/stable/index.html. [Diakses 30 November 2018].

[25 “Matplotlib,” Matplotlib, [Online]. Available: https://matplotlib.org/index.html. [Diakses 30 November 2018].

[26 A. Farhadi, “YOLO: Real-Time Object Detection,” [Online]. Available: https://pjreddie.com/darknet/yolo/. [Diakses 22 June 2019].

[27 R. Munir, Pengolahan Citra Digital dengan Pendekatan Algoritmik, Bandung: Informatika ITB.

[28 “Introduction to Convolution Neural Networks,” 4 April 2016. [Online]. Available:

https://www.analyticsvidhya.com/blog/2016/04/deep-learning-computer-vision-introduction-convolution-neural-networks/. [Diakses 29 November 2018].

[29 M. S. Sarfraz, A. Shahzad, M. A. Elahi, M. Fraz, I. Zafar dan E. A. Edirisinghe, "Real-time automatic license plate recognition," 2011.

[30 P. Wu, Z. Huang dan D. Li, "Research on the character recognition for Chinese license plate based on CNN," 2017.

[31 J. Redmon dan A. Farhadi, "YOLO9000:Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition,* 2017.

# LAMPIRAN

## L.1 Hasil Uji Coba *Optimizer* SGD

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 | 0,89 | 0,9 | 0,89 | 633 |
| 1 | 0,94 | 0,99 | 0,97 | 474 |
| 2 | 0,96 | 0,91 | 0,93 | 633 |
| 3 | 0,96 | 0,96 | 0,96 | 712 |
| 4 | 0,94 | 0,93 | 0,94 | 553 |
| 5 | 0,95 | 0,96 | 0,96 | 554 |
| 6 | 0,95 | 0,95 | 0,95 | 550 |
| 7 | 0,92 | 0,93 | 0,93 | 788 |
| 8 | 0,95 | 0,93 | 0,94 | 554 |
| 9 | 0,95 | 0,98 | 0,97 | 553 |
| A | 0,94 | 0,97 | 0,96 | 554 |
| B | 0,94 | 0,94 | 0,94 | 554 |
| C | 0,97 | 0,97 | 0,97 | 554 |
| D | 0,95 | 0,91 | 0,93 | 633 |
| E | 0,95 | 0,96 | 0,96 | 553 |
| F | 0,96 | 0,98 | 0,97 | 870 |
| G | 0,95 | 0,94 | 0,94 | 633 |
| H | 0,99 | 0,95 | 0,97 | 553 |
| I | 0,95 | 0,98 | 0,97 | 712 |
| J | 0,97 | 0,97 | 0,97 | 870 |
| K | 0,95 | 0,97 | 0,96 | 554 |
| L | 0,93 | 0,95 | 0,94 | 870 |
| M | 0,93 | 0,96 | 0,95 | 550 |
| N | 0,91 | 0,92 | 0,91 | 633 |
| O | 0,96 | 0,97 | 0,97 | 554 |

| | | | | |
|---|---|---|---|---|
| P | 0,97 | 0,96 | 0,96 | 632 |
| Q | 0,9 | 0,88 | 0,89 | 553 |
| R | 0,96 | 0,96 | 0,96 | 867 |
| S | 0,98 | 0,98 | 0,98 | 870 |
| T | 0,96 | 0,95 | 0,96 | 554 |
| U | 0,95 | 0,96 | 0,95 | 633 |
| V | 0,94 | 0,95 | 0,94 | 791 |
| W | 0,98 | 0,93 | 0,95 | 554 |
| X | 0,98 | 0,93 | 0,95 | 554 |
| Y | 0,96 | 0,95 | 0,96 | 554 |
| Z | 0,9 | 0,9 | 0,9 | 550 |
| Weighted Average | 0,95 | 0,95 | 0,95 | 22763 |

## L.2 Hasil Uji Coba *Optimizer* Adagrad

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 | 0,54 | 0,6 | 0,57 | 633 |
| 1 | 0,78 | 0,96 | 0,86 | 474 |
| 2 | 0,75 | 0,61 | 0,68 | 633 |
| 3 | 0,72 | 0,74 | 0,73 | 712 |
| 4 | 0,77 | 0,77 | 0,77 | 553 |
| 5 | 0,71 | 0,72 | 0,72 | 554 |
| 6 | 0,68 | 0,61 | 0,64 | 550 |
| 7 | 0,66 | 0,65 | 0,66 | 788 |
| 8 | 0,62 | 0,58 | 0,6 | 554 |
| 9 | 0,8 | 0,66 | 0,72 | 553 |
| A | 0,72 | 0,75 | 0,73 | 554 |
| B | 0,63 | 0,61 | 0,62 | 554 |
| C | 0,92 | 0,86 | 0,89 | 554 |

| | | | | |
|---|---|---|---|---|
| D | 0,69 | 0,52 | 0,59 | 633 |
| E | 0,81 | 0,64 | 0,72 | 553 |
| F | 0,69 | 0,84 | 0,76 | 870 |
| G | 0,79 | 0,7 | 0,74 | 633 |
| H | 0,75 | 0,63 | 0,68 | 553 |
| I | 0,7 | 0,87 | 0,78 | 712 |
| J | 0,88 | 0,89 | 0,89 | 870 |
| K | 0,62 | 0,73 | 0,67 | 554 |
| L | 0,67 | 0,8 | 0,73 | 870 |
| M | 0,75 | 0,84 | 0,79 | 550 |
| N | 0,59 | 0,66 | 0,62 | 633 |
| O | 0,78 | 0,66 | 0,72 | 554 |
| P | 0,81 | 0,77 | 0,79 | 632 |
| Q | 0,64 | 0,59 | 0,61 | 553 |
| R | 0,71 | 0,75 | 0,73 | 867 |
| S | 0,78 | 0,83 | 0,8 | 870 |
| T | 0,75 | 0,72 | 0,74 | 554 |
| U | 0,76 | 0,76 | 0,76 | 633 |
| V | 0,68 | 0,68 | 0,68 | 791 |
| W | 0,76 | 0,78 | 0,77 | 554 |
| X | 0,79 | 0,63 | 0,7 | 554 |
| Y | 0,77 | 0,75 | 0,76 | 554 |
| Z | 0,7 | 0,72 | 0,71 | 550 |
| Weighted Average | 0,73 | 0,72 | 0,72 | 22763 |

## L.3 Hasil Uji Coba *Optimizer* RMSprop

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 | 0,99 | 0,98 | 0,98 | 633 |

| | | | | |
|---|---|---|---|---|
| 1 | 0,98 | 0,99 | 0,99 | 474 |
| 2 | 0,99 | 0,99 | 0,99 | 633 |
| 3 | 1 | 0,99 | 0,99 | 712 |
| 4 | 0,99 | 1 | 0,99 | 553 |
| 5 | 0,99 | 0,99 | 0,99 | 554 |
| 6 | 1 | 0,99 | 1 | 550 |
| 7 | 0,99 | 0,99 | 0,99 | 788 |
| 8 | 1 | 0,99 | 1 | 554 |
| 9 | 0,99 | 1 | 0,99 | 553 |
| A | 1 | 0,99 | 0,99 | 554 |
| B | 1 | 1 | 1 | 554 |
| C | 1 | 0,99 | 1 | 554 |
| D | 1 | 0,99 | 0,99 | 633 |
| E | 0,99 | 0,99 | 0,99 | 553 |
| F | 0,99 | 1 | 0,99 | 870 |
| G | 0,99 | 1 | 1 | 633 |
| H | 1 | 1 | 1 | 553 |
| I | 0,98 | 0,99 | 0,99 | 712 |
| J | 1 | 1 | 1 | 870 |
| K | 0,99 | 1 | 1 | 554 |
| L | 0,99 | 0,99 | 0,99 | 870 |
| M | 1 | 1 | 1 | 550 |
| N | 0,99 | 0,99 | 0,99 | 633 |
| O | 1 | 1 | 1 | 554 |
| P | 1 | 1 | 1 | 632 |
| Q | 0,98 | 0,98 | 0,98 | 553 |
| R | 1 | 1 | 1 | 867 |
| S | 0,99 | 0,99 | 0,99 | 870 |
| T | 0,99 | 0,98 | 0,98 | 554 |

| | | | | |
|---|---|---|---|---|
| U | 0,99 | 1 | 1 | 633 |
| V | 1 | 1 | 1 | 791 |
| W | 0,99 | 1 | 1 | 554 |
| X | 1 | 0,99 | 0,99 | 554 |
| Y | 0,99 | 0,99 | 0,99 | 554 |
| Z | 0,99 | 0,99 | 0,99 | 550 |
| Weighted Average | 0,99 | 0,99 | 0,99 | 22763 |

## L.4 Hasil Uji Coba *Optimizer* Adam

| Kelas | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 | 0,99 | 0,98 | 0,99 | 633 |
| 1 | 1 | 1 | 1 | 474 |
| 2 | 0,99 | 0,99 | 0,99 | 633 |
| 3 | 1 | 1 | 1 | 712 |
| 4 | 1 | 1 | 1 | 553 |
| 5 | 0,99 | 1 | 1 | 554 |
| 6 | 1 | 0,99 | 1 | 550 |
| 7 | 0,99 | 0,99 | 0,99 | 788 |
| 8 | 1 | 1 | 1 | 554 |
| 9 | 1 | 1 | 1 | 553 |
| A | 1 | 1 | 1 | 554 |
| B | 1 | 0,99 | 1 | 554 |
| C | 1 | 0,99 | 1 | 554 |
| D | 0,99 | 0,99 | 0,99 | 633 |
| E | 0,99 | 0,99 | 0,99 | 553 |
| F | 1 | 0,99 | 0,99 | 870 |
| G | 0,99 | 0,99 | 0,99 | 633 |
| H | 1 | 0,99 | 1 | 553 |

| | | | | |
|---|---|---|---|---|
| I | 0,99 | 1 | 0,99 | 712 |
| J | 1 | 1 | 1 | 870 |
| K | 1 | 1 | 1 | 554 |
| L | 0,99 | 1 | 0,99 | 870 |
| M | 1 | 1 | 1 | 550 |
| N | 0,99 | 1 | 0,99 | 633 |
| O | 1 | 1 | 1 | 554 |
| P | 0,99 | 1 | 1 | 632 |
| Q | 0,98 | 0,99 | 0,99 | 553 |
| R | 1 | 1 | 1 | 867 |
| S | 1 | 1 | 1 | 870 |
| T | 0,99 | 0,99 | 0,99 | 554 |
| U | 0,99 | 1 | 1 | 633 |
| V | 0,99 | 0,99 | 0,99 | 791 |
| W | 1 | 1 | 1 | 554 |
| X | 1 | 1 | 1 | 554 |
| Y | 1 | 1 | 1 | 554 |
| Z | 1 | 0,99 | 1 | 550 |
| Weighted Average | 1 | 1 | 1 | 22763 |

# BIODATA PENULIS

Pradipta Baskara, lahir di Denpasar pada tanggal 14 Juni 1997. Penulis menempuh pendidikan mulai dari TK Cipta Dharma (2002- 2003), SD Cipta Dharma (2003-2009), SMP Negeri 3 Denpasar (2009-2012), SMA Negeri 4 Denpasar (2012-2015), dan sekarang sedang menjalani pendidikan S1 Informatika di ITS. Penulis aktif dalam organisasi dan kepanitiaan Himpunan Mahasiswa Teknik Computer (HMTC) dan Schematics. Diantaranya adalah menjadi staff Departemen Pengembangan Profesi HMTC ITS 2016-2017, staff ahli Departemen Pengembangan Profesi HMTC ITS 2017-2018, staff Departemen National Seminar of Technology Schematics ITS 2016 dan staff ahli Departemen National Seminar of Technology Schematics ITS 2017. Komunikasi dengan penulis dapat melalui telepon: +6281236220910 dan *email*: **baaskaraaa@gmail.com**.