Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IF184802

DETEKSI KETERSEDIAAN LAHAN PARKIR PADA DATA VIDEO MENGGUNAKAN MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORK

RAHANDI NOOR PASHA
NRP 05111640000054

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

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

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IF184802

DETEKSI KETERSEDIAAN LAHAN PARKIR PADA DATA VIDEO MENGGUNAKAN MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORK

RAHANDI NOOR PASHA
NRP 05111640000054

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

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

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF184802

PARKING LOT OCCUPANCY DETECTION IN VIDEO DATA USING MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORK

RAHANDI NOOR PASHA
NRP 05111640000054

First Advisor
Dini Adni Navastara, S.Kom., M.Sc.

Second Advisor
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

DETEKSI KETERSEDIAAN LAHAN PARKIR PADA DATA VIDEO MENGGUNAKAN MASK REGION-BASED 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 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
RAHANDI NOOR PASHA
NRP: 05111640000054

Disetujui oleh Pembimbing Tugas Akhir:

1. Dini Adni Navastara, S.Kom., M.Sc. .....
   (NIP. 19851017 201504 2 001) (Pembimbing 1)

2. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. .....
   (NIP. 19751220 200112 2 002) (Pembimbing 2)

SURABAYA
Januari, 2020

(Halaman ini sengaja dikosongkan)

DETEKSI KETERSEDIAAN LAHAN PARKIR MENGGUNAKAN MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORK

Nama Mahasiswa : Rahandi Noor Pasha
NRP : 05111640000054
Departemen : Teknik Informatika, FTEIC-ITS
Dosen Pembimbing 1 : Dini Adni Navastara, S.Kom., M.Sc.
Dosen Pembimbing 2 : Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRAK

Machine learning telah menjadi bagian dari kehidupan sehari-hari bagi banyak orang. Salah satu pengaplikasian machine learning adalah deteksi ketersediaan lahan parkir. Deteksi ketersediaan lahan parkir mengkategorikan lahan parkir menjadi tersedia atau sedang terpakai berdasarkan fitur dari gambar tersebut. Banyak perusahaan, badan riset dan universitas yang terus mengembangkan machine learning agar mendapat hasil yang lebih akurat dan cepat. Convolutional Neural Network (CNN) adalah salah satu deep neural network yang cocok digunakan untuk mengolah data yang berbentuk 2 dimensi, seperti gambar dan video.

Tugas akhir ini mengusulkan sistem deteksi ketersediaan lahan parkir secara otomatis menggunakan Mask Region-based Convolutional Neural Network. Data pelatihan menggunakan dataset "Common Objects in Context" (COCO) yang berisi gambar bermacam – macam objek, data uji coba diambil dari dataset CNRPark yang berisi foto dari banyak lahan parkir dan data CCTV departemen Informatika ITS yang nantinya menjadi tujuan deteksi ketersediaan lahan parkir yang dibuat. Hasil evaluasi sistem pada dataset CNRPark didapatkan rata – rata akurasi sebesar 81,14% dan pada data CCTV departemen Informatika ITS didapatkan rata – rata akurasi sebesar 86,07%

Kata kunci: *Convolutional Neural Network*, Data CCTV, Dataset Common Objects in Context, Dataset CNRPark, Deteksi Ketersediaan Lahan Parkir, *Mask Region-based Convolutional Neural Network*.

PARKING LOT OCCUPANCY DETECTION USING MASK REGION-BASED CONVOLUTIONAL NEURAL NETWORK

Student's Name : Rahandi Noor Pasha
Student's ID : 05111640000054
Department : Informatics, Faculty of ELECTICS-ITS
First Advisor : Dini Adni Navastara, S.Kom., M.Sc.
Second Advisor : Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRACT

Machine learning has become a part of the daily life of people around the world. One of the application of machine learning is parking lot occupancy detection. Parking lot occupancy detection categorize a parking lot whether its occupied or unoccupied based on the features extracted from the image. Many companies, researchers and universities keep improving the machine learning to get a better and faster result. 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, the writer is proposed an algorithm for parking lot occupancy detection automatically. The goal is simplifying and reducing the cost of parking lot occupancy detection. The train data used in this thesis is taken from "Common Objects in Context" (COCO) dataset which contains images of various kinds of objects, the test data is taken from CNRPark which contains pictures from many parking spots which will be the goal of parking lot occupancy detections. The result of the system evaluation on the CNRPark datasets obtained an average accuracy of 81.14% and the CCTV data of the ITS Informatics Department obtained an average accuracy of 86,07%

Keywords: ***Convolutional Neural Network, Human Facial Expression Recognition, Image Data, Karolinska Directed Emotional Faces Dataset, Wavelet Transform.***

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

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

"**DETEKSI KETERSEDIAAN LAHAN PARKIR MENGGUNAKAN *MASK REGION-BASED 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. Tuhan Yang Maha Esa, karena limpahan rahmat dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Informatika ITS.
2. Kedua orangtua penulis, dan anggota keluarga lainnya yang telah memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Dini Adni Navastara, S.Kom., M.Sc. dan Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku Ketua 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.
5. Admin-admin Laboratorium Komputasi Cerdas & Visi (KCV) yang memberikan kesempatan penulis untuk fokus

mengerjakan Tugas Akhir ini dan menyediakan tempat di laboratorium tersebut.

6. Yoshima Syach Putri yang selalu memberi semangat dan dukungan moral kepada penulis.
7. Seluruh mahasiswa Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama menjalani masa kuliah di Informatika ITS.
8. 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 2020

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 memprediksi 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*.

Salah satu pengaplikasian *machine learning* yang populer adalah *image classification. Image classification* mengategorikan 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*.

Sistem *Parking Guidance and Information* (PGI) merupakan sistem yang bertujuan untuk menyediakan indikasi secara *real-time* tentang keberadaan tempat parkir. Hingga saat ini sistem tersebut diimplementasikan menggunakan sensor dengan harga tinggi dan pada tempat tertutup.

Convolutional Neural Network (CNN) adalah salah satu *deep neural network* yang cocok digunakan untuk mengolah data yang berbentuk dua dimensi, seperti gambar dan video. Normalnya, CNN diterapkan untuk mengklasifikasikan gambar yang belum diolah (citra asli). Di dalam Tugas Akhir ini, penulis

mengusulkan penggunaan *Mask Region-based Covolutional Neural Network*. Data latih diambil dari dataset "*Common Object in Context*" (COCO) yang terdiri dari 123.287 gambar yang memiliki 80 jenis objek, data test akan diambil dari CNRPark dataset.

1.2 Rumusan Masalah

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

1. Bagaimana cara mendeteksi lahan parkir?
2. Bagaimana cara implementasi *Convolutional Neural Network* untuk ekstraksi fitur pada data lahan parkir?
3. Bagaimana cara implementasi *Region Proposal Network* untuk klasifikasi?
4. Bagaimana cara menandai ketersediaan lahan parkir secara otomatis pada data uji?
5. Bagaimana cara evaluasi sistem yang dibangun?

1.3 Batasan Permasalahan

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

1. Data latih diambil dari dataset "Common Object in Context" (COCO).
2. Data uji diambil dari CNRPark dataset.
3. Implementasi program menggunakan bahasa pemrograman *Python*.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah membangun sebuah sistem deteksi ketersediaan lahan parkir dengan menggunakan *Mask Region-based Convolutional Neural Network.*

1.5 Manfaat

Tugas akhir ini diharapkan dapat membantu menambah kemampuan yang ada pada sistem *Parking Guidance and Information* (PGI).

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 dataset CNRPark untuk mengetahui hasil dan performa arsitektur yang telah dibangun. Evaluasi dilakukan dengan metode pengukuran akurasi, presisi, dan *recall*.

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 arsitektur *Mask Region-based Convolutional Neural Network* yang digunakan untuk pengenalan ketersediaan lahan parkir 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)

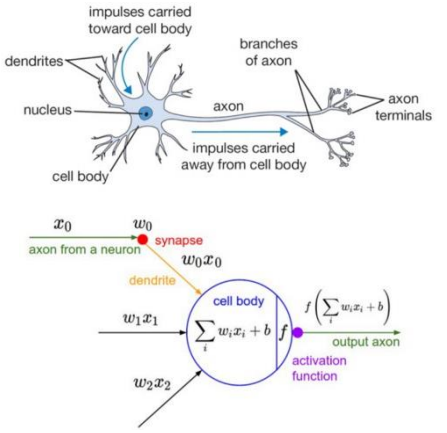
BAB II
TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut diantaranya 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.

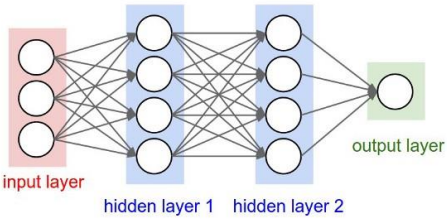
2.1 Multi Layer Perceptron

Multi Layer Perceptron (MLP) atau dikenal dengan *Fully Connected Layer* adalah sebuah algoritma yang terinspirasi dari bagaimana neuron dalam otak manusia bekerja. Tiap neuron pada otak manusia saling berhubungan dan informasi mengalir dari setiap neuron tersebut. Tiap neuron menerima *input* dan melakukan operasi dot dengan sebuah bobot atau nilai bobot dan menambahkan nilai bias. Hasil dari operasi ini akan dijadikan parameter dari fungsi aktivasi yang akan dijadikan *output* dari neuron tersebut [1]. Ilustrasi dapat dilihat pada Gambar 2.1. MLP terdiri dari *Input Layer*, *Hidden Layer*, *Output Layer*. *Input Layer* menerima masukan (tanpa melalui fungsi aktivasi), kemudian nilai diberikan ke *Hidden Layer*, dimana akan dilakukan perhitungan hasil fungsi aktivasi untuk tiap-tiap neuron, lalu hasilnya diberikan ke *Output Layer*. Ilustrasi arsitektur *Multi Layer Perceptron* terlihat pada Gambar 2.2.

Pada tahap pelatihan nilai bobot dan bias pada tiap neuron akan diperbarui terus menerus hingga *output* yang dihasilkan sesuai yang diharapkan. Pada tiap iterasi akan dilakukan proses evaluasi yang biasanya digunakan untuk menentukan kapan harus menghentikan proses pelatihan. Proses pelatihan MLP terdiri dari *Forward* dan *Backward Pass.*



Gambar 2.1 Ilustrasi *neuron* manusia dalam model matematika [1]



Gambar 2.2 Ilustrasi arsitektur *Multi Layer Perceptron* [1]

Tahap *Forward Pass* adalah proses dimana data masukan dibawa melewati tiap neuron sampai kepada *Output Layer* yang nantinya akan dihitung nilai *error*. Persamaan (2.1) adalah operasi dot *x* (*input*) dengan *w* (bobot) dan ditambah dengan *b* (bias) yang kemudian menggunakan fungsi aktivasi, misalnya ReLU pada Persamaan (2.2).

$$dot_j = \sum_i^n w_{ij} x_i + b_j \tag{2.1}$$

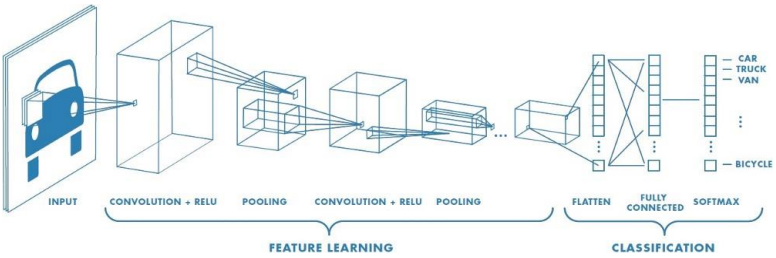
$$h_j = f(dot_j) = \max(0, dot_j) \tag{2.2}$$

Tahap *Backward Pass* bertujuan untuk menyesuaikan kembali tiap bobot dan bias berdasarkan *error* yang didapat pada saat *forward pass*. Proses ini disebut sebagai propagasi balik (*backpropagation*) yaitu tahap pelatihan yang mengubah *weight* neuron-neuron di dalam MLP. Propagasi balik memanfaatkan sebuah *loss function* untuk menghitung *error* dari nilai prediksi dengan nilai sebenarnya. Selanjutnya, digunakan algoritma pengoptimalan untuk memperbarui bobot dan bias dengan tujuan menurunkan nilai *error*, hal ini dilakukan secara iteratif sampai *epoch* tertentu.

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 [2].

Penelitian awal yang mendasari penemuan ini dilakukan oleh Hubel dan Wiesel [3] 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* [4], seperti yang dipaparkan pada Gambar 2.3.



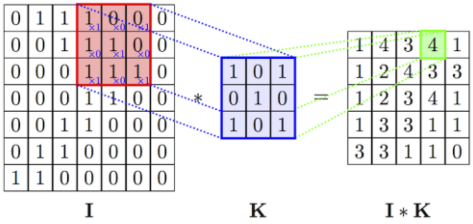
Gambar 2.3 Contoh arsitektur *Convolutional Neural Network* [5]

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

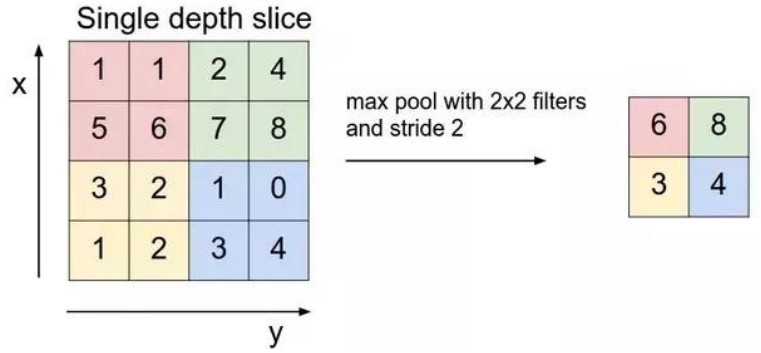
Convolution Layer melakukan operasi konvolusi pada output dari lapisan sebelumnya. Konvolusi adalah istilah matematis yang artinya mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstrak fitur dari citra masukan [2]. Ilustrasi cara kerja konvolusi bisa dilihat pada Gambar 2.4, dimana I adalah citra, K adalah *filter* atau *kernel* yang digunakan, I * K adalah hasil operasi konvolusi.



Gambar 2.4 Ilustrasi cara kerja konvolusi [6]

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 2.5.



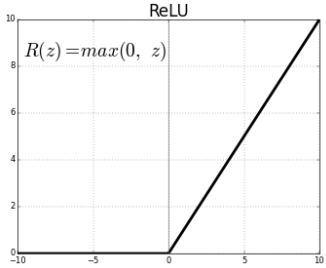
Gambar 2.5 Ilustrasi cara kerja *Max Pooling* [7]

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 [8].

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 2.6.



Gambar 2.6 ReLU *Activation Function* [9]

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 (2.3).

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.3}$$

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 (2.4). Selanjutnya, nilai *error* akhir diambil dari rata-rata hasil *cross entropy*, seperti pada Persamaan (2.4) dan (2.5).

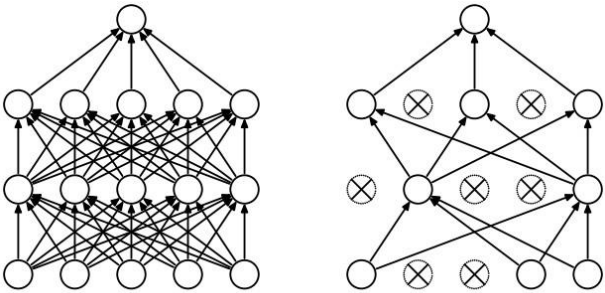
$$D(S_i, T_i) = -\sum_j T_{ij} \log S_{ij} \tag{2.4}$$

$$J(W, b) = \frac{1}{n} \sum_i D(S_i, T_i) \tag{2.5}$$

2.2.7 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 [10]. Dengan menghilangkan suatu neuron, berarti menghilangkannya sementara dari jaringan yang ada. Neuron yang akan dihilangkan akan dipilih secara acak.

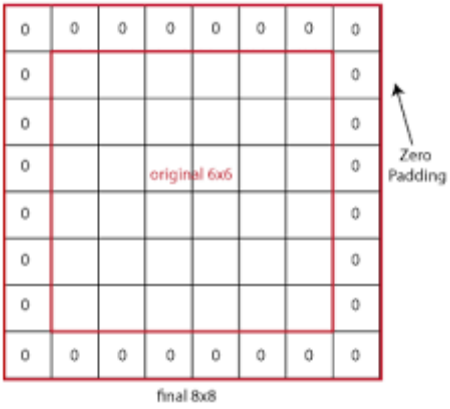
Pada Gambar 2.7, (a) neuron tetap utuh pada *neural network* yang belum memakai *Dropout*, dan (b) *neural network* yang sebagian dari neuronnya tidak digunakan setelah diaplikasikan Dropout.



Gambar 2.7 Ilustrasi *neural network* dalam mengaplikasikan *Dropout* [1]

2.2.8 ZeroPadding2D

ZeroPadding2D bertujuan untuk menambahkan nilai 0 pada sisi tepi atas, bawah, kanan, dan kiri pada sebuah gambar agar gambar tersebut memiliki ukuran yang dibutuhkan [11]



Gambar 2.8 Ilustrasi pengaplikasian ZeroPadding2D

2.2.9 UpSampling2D

UpSampling2D adalah sebuah *layer* yang berguna untuk menggandakan semua baris dan kolom dari matrix yang dimasukkan [11]

```
            1, 2
Input = (3, 4)

             1, 1, 2, 2
Output = (1, 1, 2, 2)
             3, 3, 4, 4
             3, 3, 4, 4
```

Gambar 2.9 Ilustrasi pengaplikasian UpSampling2D

2.2.10 Batch Normalization

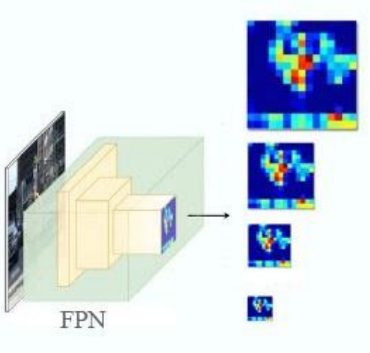
Batch Normalization adalah teknik melakukan normalisasi terhadap *batch* atau kumpulan data masukan, seperti yang terlihat

pada Persamaan (2.6). Dimana *x* adalah nilai masukan, μ_b adalah *mean* dari *batch*, σ_b adalah standar deviasi dari *batch*. Normalisasi dilakukan agar data memiliki *mean* mendekati 0 dan standar deviasi mendekati 1. [11]

$$x_{baru} = \frac{x-\mu_b}{\sigma_b} \tag{2.6}$$

2.3 Feature Pyramid Network

Feature Pyramid Network (FPN) adalah sebuah algoritma yang berfungsi untuk mengekstrak fitur dari gambar dengan ukuran yang berbeda sehingga dapat lebih baik saat menentukan fitur dari gambar tersebut. [12]



Gambar 2.10 Ilustrasi pengaplikasian FPN Top-Down

2.4 Non-maximum Suppression

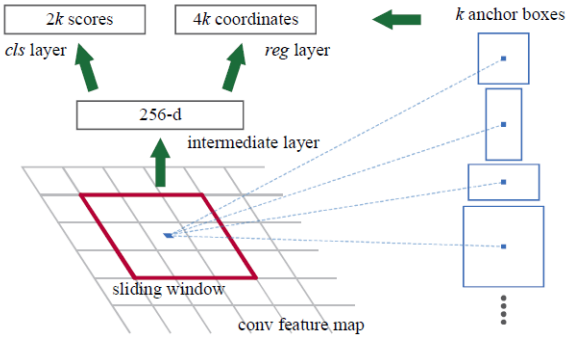
Non-maximum Suppresion (NMS) adalah sebuah algoritma yang berguna untuk mengurangi jumlah proposal yang hasil dari RPN berdasarkan skornya, hal ini bertujuan untuk menambah performa dari arsitektur karena tidak perlu memproses semua proposal yang ada. [13]

Gambar 2.11 Ilustrasi pengaplikasian NMS

2.5 Region Proposal Network

Region Proposal Network adalah sebuah algoritma yang bertujuan untuk mendapatkan bagian – bagian gambar yang memiliki kemungkinan sebagai sebuah objek hal ini dilakukan agar proses klasifikasi menjadi lebih ringan karena hanya memproses sebagian kecil dari gambar.

Region Proposal Network mengambil masukan berupa *feature maps* dan memberikan keluaran berupa kumpulan kotak dengan koordinat yang menunjukkan posisi kemungkinan objek di dalam gambar. Ilustrasi dari *Region Proposal Network* dapat dilihat pada Gambar 2.12.



Gambar 2.12 Ilustrasi *Region Proposal Network* [14]

2.6 Region of Interest

Region of Interest (ROI) merupakan bagian dari sebuah gambar yang dibutuhkan untuk identifikasi dengan tujuan tertentu. Konsep dari ROI telah banyak digunakan di berbagai bidang. Berikut adalah contoh ROI berdasarkan dimensinya:

- Dataset 1D: berupa waktu atau interval frekuensi dalam bentuk gelombang.
- Dataset 2D: berupa batas – batas suatu objek pada gambar.
- Dataset 3D: kontur atau permukaan dai sebuah objek volume (atau biasanya dikenal sebagai *Volume of Interest* (VOI)).
- Dataset 4D: *outline* dari sebuah objek atau interval waktu pada volume waktu.

Macam – macam bentuk ROI 2D yaitu *circle, ellipse, rectangle,* dan *polygonal*. [15]

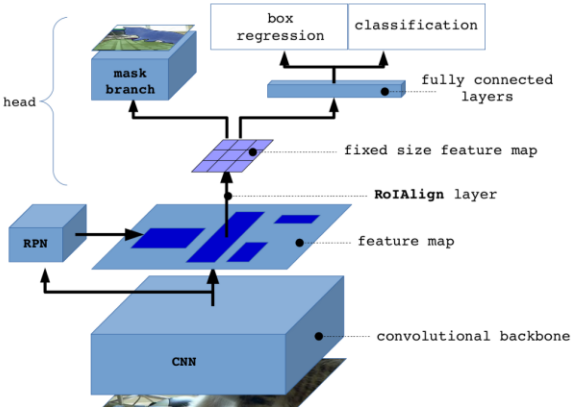
2.7 Mask Region-based Convolutional Neural Network

Mask Region-based Convolutional Neural Network (Mask RCNN) adalah algoritma *machine learning* yang menggunakan *Region Proposal Network* untuk mempercepat keseluruhan proses klasifikasi [16].

Cara kerja dari *Mask Region-based Convolutional Neural Network*, adalah:

1. Gambar masukan akan dilewatkan sebuah arsitektur *Convolutional Neural Network* untuk mendapatkan *feature maps* dari gambar tersebut.
2. *Feature maps* akan dimasukkan ke arsitektur *Region Proposal Network* hingga mendapatkan kumpulan kotak ROI dari kemungkinan – kemungkinan objek yang ada dalam gambar
3. *Feature maps* akan dipasangkan dengan ROI untuk dimasukkan ke sebuah MLP untuk mendapatkan hasil dari klasifikasi dan mask dari setiap ROI.

Ilustrasi cara kerja *Faster Region-based Convolutional Neural Network* bisa dilihat pada Gambar 2.13.



Gambar 2.13 Cara kerja Mask RCNN [17]

2.8 Confusion Matrix

Evaluasi terhadap kinerja suatu algoritma klasifikasi dilakukan untuk mengetahui seberapa baik algoritma dalam mengklasifikasikan data. *Confusion Matrix* merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu algoritma klasifikasi. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh algoritma dengan hasil klasifikasi yang seharusnya.

Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat empat istilah sebagai representasi hasil proses klasifikasi yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Nilai TN merupakan jumlah data negatif yang terdeteksi dengan benar, sedangkan FP merupakan data negatif yang terdeteksi dengan benar, sedangkan FP merupakan data negatif namun terdeteksi sebagai data positif. Sementara TP merupakan data positif

yang terdeteksi benar dan FN merupakan kebalikan dari TP, yaitu data positif yang terdeteksi sebagai data negatif.

Dari *confusion matrix* bisa didapatkan hasil pengukuran berupa presisi, *recall*, dan akurasi.

2.8.1 Presisi

Presisi adalah perbandingan dari data positif yang terdeteksi positif dibanding dengan total data yang terdeteksi positif

$$Presisi = \frac{TP}{TP + FP} \tag{2.7}$$

2.8.2 Recall

Recall adalah perbandingan dari data positif yang terdeteksi positif disbanding dengan total data positif

$$Recall = \frac{TP}{TP + FN} \tag{2.8}$$

2.8.3 Akurasi

Akurasi adalah perbandingan dari data positif yang terdeteksi positif dibandingkan dengan semua data.

$$Akurasi = \frac{TP}{TP + FP + FN + TN} \tag{2.9}$$

2.9 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 [18].

2.10 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 [19].

2.11 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 [11].

2.12 OpenCV

OpenCV (*Open Source Computer Vision*) adalah *library* yang dimanfaatkan dalam pengolahan citra dinamis secara *real-time*. OpenCV dapat digunakan dalam berbagai bahasa pemrograman 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 [20].

2.13 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 [21].

2.14 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 [22].

2.15 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 [23].

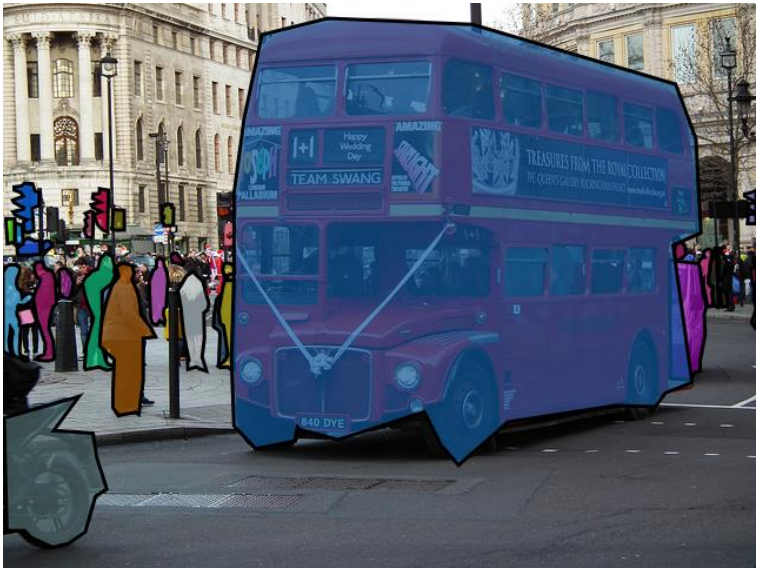
(Halaman ini sengaja dikosongkan)

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan mengenai perancangan data dan sistem deteksi ketersediaan lahan parkir menggunakan *Mask Region-based Convolutional Neural Network*. Bab ini juga akan menjelaskan gambaran umum sistem dalam bentuk diagram alir.

3.1 Perancangan Data

Data yang digunakan sebagai data latih dari sistem deteksi ketersediaan lahan parkir menggunakan *Mask Region-based Convolutional Neural Network* adalah data "Common Object in Context" (COCO). COCO adalah dataset yang terdiri dari 123.287 gambar yang memiliki 80 jenis objek. Setiap gambar memiliki masking dan anotasi yang menandakan objek yang ada dalam gambar tersebut. Contoh data gambar bisa dilihat pada Gambar 3.1.



Gambar 3.1 Contoh gambar pada dataset COCO

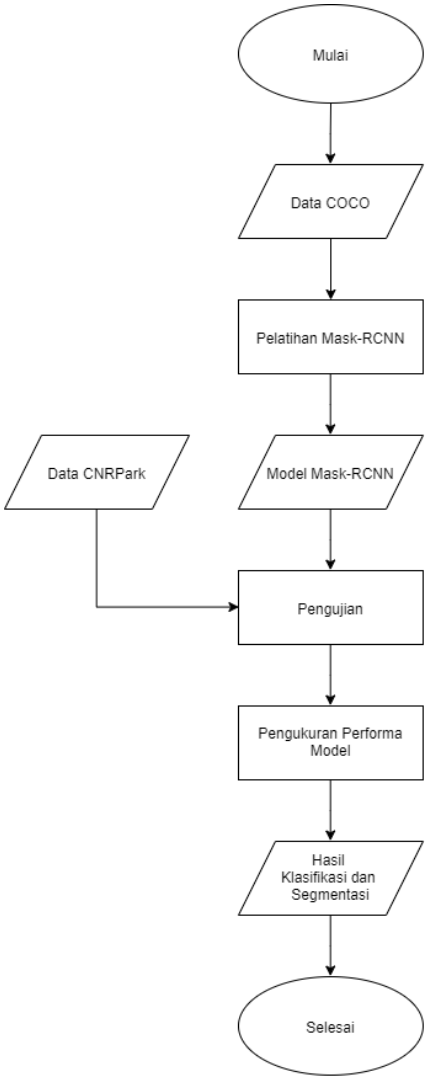
Data yang digunakan sebagai data uji dari sistem deteksi ketersediaan lahan parkir menggunakan *Mask Region-based Convolutional Neural Network* adalah data dari CNRPark Dataset. CNRPark Dataset terdiri dari 4.081 gambar yang diambil dari 9 area parkir yang berbeda pada cuaca cerah, berawan, dan hujan. Setiap gambar memiliki koordinat lahan parkir dan ada tidaknya mobil di lahan parkir tersebut. Contoh data gambar dari CNRPark Dataset bias dilihat pada Gambar 3.2

Gambar 3.2 Contoh gambar pada dataset CNRPark

Data latih diproses menggunakan *Mask Region-based Convolutional Neural Network* untuk membangun model, kemudian model digunakan dalam pengenalan terhadap data uji dan kinerja model diukur dengan akurasi, presisi, dan *recall* untuk mengukur kinerja deteksi ketersediaan lahan parkir.

3.2 Desain Umum Model Mask-RCNN

Model Mask-RCNN yang dibangun memiliki dua proses utama yaitu pelatihan dan pengujian *Mask Region-based Convolutional Neural Network*. Diagram alir dari sistem ditunjukan pada Gambar 3.3.



Gambar 3.3 Diagram alir pelatihan dan pengujian Mask-RCNN

Proses pelatihan adalah proses pembuatan model deteksi ketersediaan lahan parkir. Data latih akan diekstraksi fiturnya

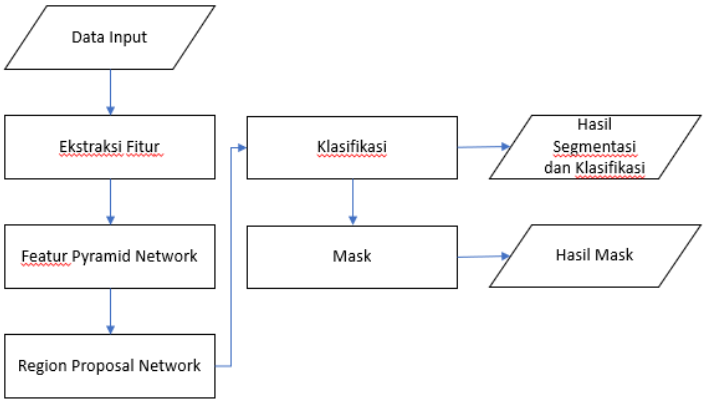
yakni melalui proses konvolusi, *pooling*, dan fungsi-fungsi aktivasi. Selanjutnya, hasil dari proses pelatihan tersebut akan menjadi model untuk proses klasifikasi. Data uji juga akan diekstraksi fiturnya untuk menjadi *input* dalam proses pengujian dengan cara di masukkan ke dalam model yang sudah dilatih untuk mendapatkan prediksi label kelasnya.

3.2.1 Tahap Pembangunan Arsitektur

Pembangunan model bertujuan untuk menyiapkan layer, fungsi aktivasi, *loss function* dan paremeter apa saja yang dibutuhkan. Arsitektur Mask RCNN memiliki lima bagian utama, yaitu:

1. Ekstraksi Fitur
2. *Feature Pyramid Network*
3. *Region Proposal Network*
4. Klasifikasi
5. Mask

Untuk desain umum dari arsitektur Mask RCNN dapat dilihat pada Gambar 3.4

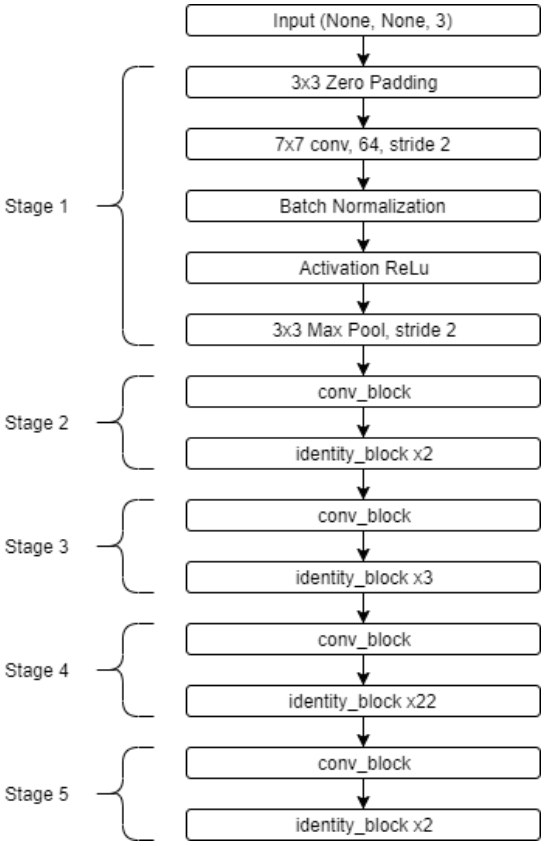


Gambar 3.4 Desain umum arsitektur Mask-RCNN

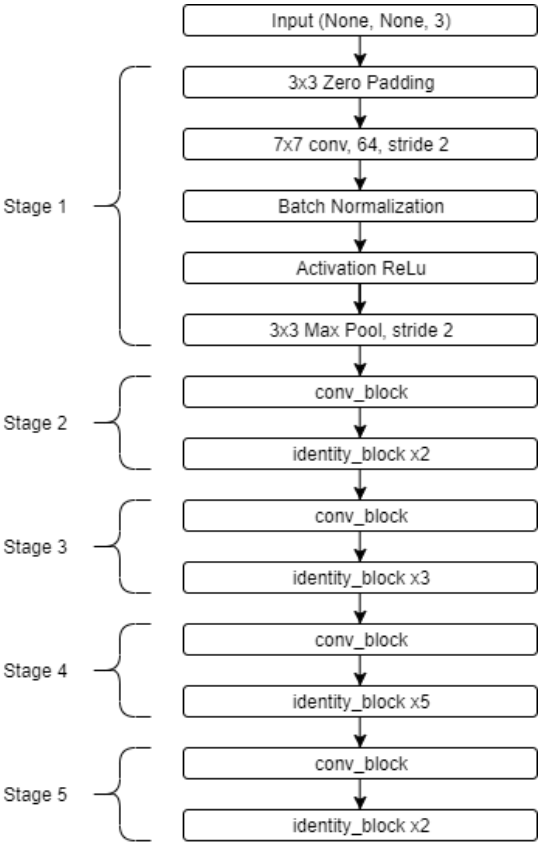
3.2.1.1 Ekstraksi Fitur

Pada Tugas Akhir kali ini penulis menggunakan arsitektur Resnet101 sebagai arsitektur untuk ekstraksi fitur dan menggunakan Resnet50 sebagai scenario ujicoba.

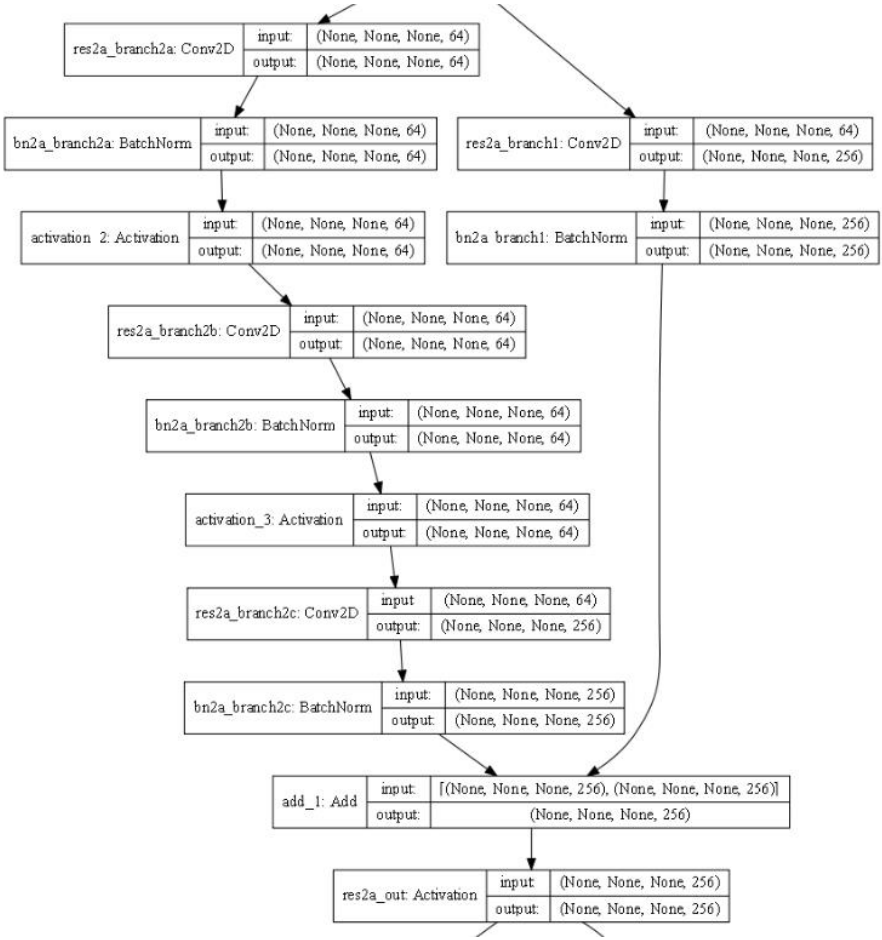
Pada arsitektur Resnet101 dan Resnet50 memiliki 2 kumpulan *layer* berulang dengan nama conv_block dan identity_block, untuk arsitektur Resnet101 secara keseluruhan dapat dilihat pada Gambar 3.5, untuk arsitektur Resnet50 secara keseluruhan dapat dilihat pada Gambar 3.6, untuk arsitektur conv_block dapat dilihat pada Gambar 3.7 dan untuk arsitektur identity_block dapat dilihat pada Gambar 3.8



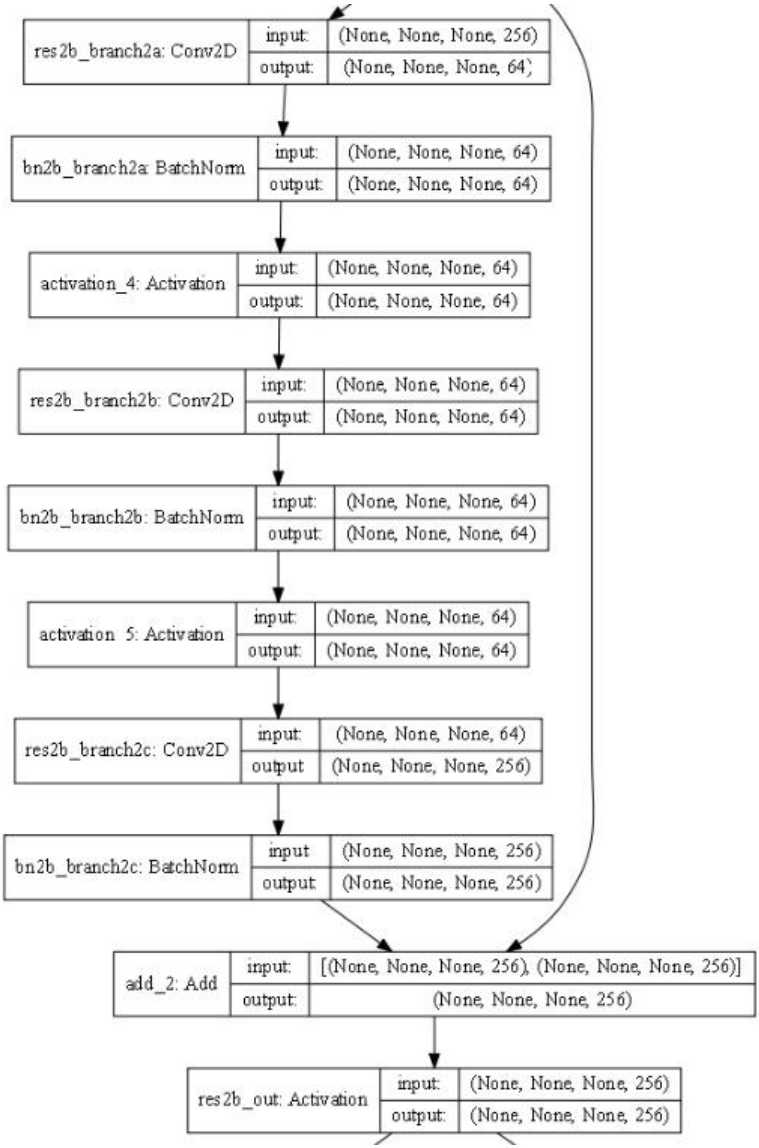
Gambar 3.5 Ilustrasi arsitektur Resnet101



Gambar 3.6 Ilustrasi arsitektur Resnet50



Gambar 3.7 Ilustrasi arsitektur conv_block



Gambar 3.8 Ilustrasi arsitektur identity_block

Berikut detail arsitektur Resnet101:

1. Input untuk arsitektur Resnet101 adalah sebuah gambar yang memiliki 3 channel
2. Untuk tahap pertama setiap input akan melalui sebuah *layer Zero Padding 2D*, lalu melewati *layer Convolutional 2D*, *Batch Normalization*, *layer* aktivasi relu, lalu diakhiri dengan *layer Max Pooling 2D*
3. Pada tahap kedua terdapat 1 conv_block dan 2 identity_block
4. Pada tahap ketiga terdapat 1 conv_block dan 3 identity_block
5. Pada tahap empat terdapat 1 conv_block dan 22 identity_block
6. Pada tahap lima terdapat 1 conv_block dan 2 identity_block

Arsitektur Resnet50 mirip dengan Resnet101 namun poin 5 memiliki 1 conv_block dan 5 identity_block.

3.2.1.2 *Feature Pyramid Network*

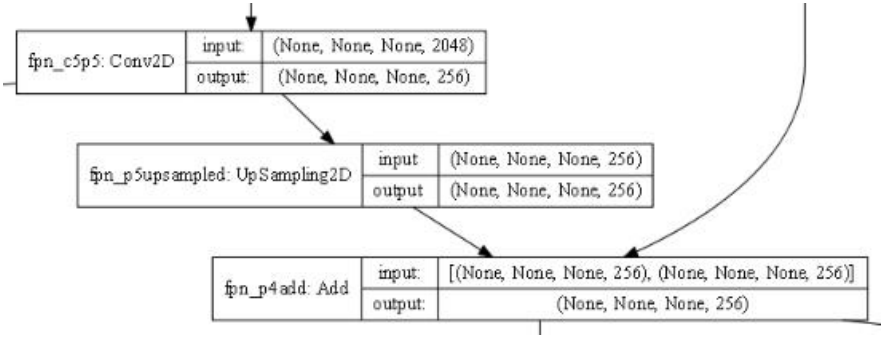
Arsitektur *Featur Pyramid Network* (FPN) memiliki 4 bagian, dengan penyusunan mulai dari bagian empat, detail dari arsitektur FPN adalah sebagai berikut:

1. Bagian empat dari FPN berupa *layer Convolutional 2D* yang disambung ke tahap lima dari arsitektur Resnet101, lalu ditambah dengan 1 *layer Convolutional 2D.*
2. Bagian tiga dari FPN berupa *layer UpSampling2D* yang disambungkan ke bagian empat dari FPN dan *layer Convolutional2D* yang disambungkan ke tahap empat dari arsitektur Resnet101, lalu kedua *layer* tersebut dijadikan 1 menggunakan *layer add*, lalu ditambah dengan *layer Convolutional2D.*
3. Bagian dua dari FPN berupa *layer UpSampling2D* yang disambungkan ke bagian tiga dari FPN dan *layer Convolutional2D* yang disambungkan ke tahap tiga

dari arsitektur Resnet101, lalu kedua *layer* tersebut dijadikan 1 menggunakan *layer add*, lalu ditambah dengan *layer Convolutional2D.*

4. Bagian satu dari FPN berupa *layer UpSampling2D* yang disambungkan ke tahap dua dari FPN dan *layer Convolutional2D* yang disambungkan ke tahap dua dari arsitektur Resnet101, lalu kedua *layer* tersebut dijadikan 1 menggunakan *layer add*, lalu ditambah dengan *layer Convolutional2D.*
5. Lalu ada bagian tambahan yang berupa *layer MaxPooling2D* yang disambungkan ke bagian empat dari FPN.

Ilustrasi FPN dapat dilihat pada Gambar 3.9



Gambar 3.9 Ilustrasi arsitektur bagian FPN

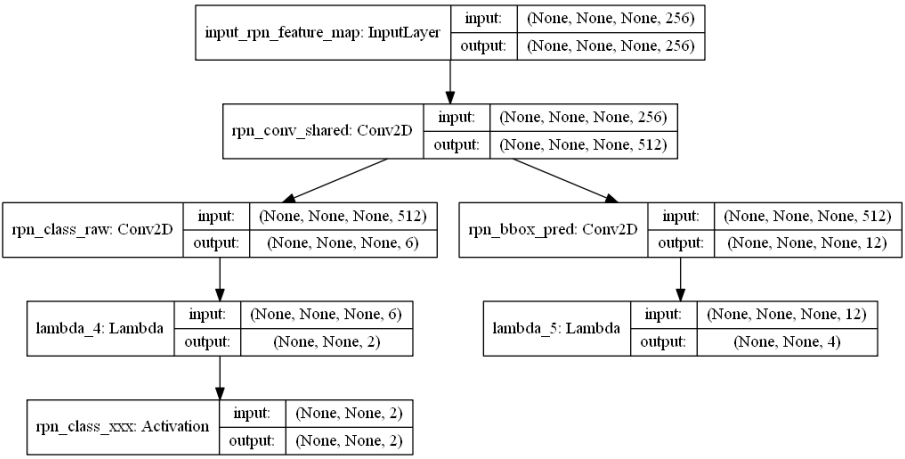
3.2.1.3 *Region Proposal Network*

Untuk bagian *Region Proposal Network* (RPN) terdapat 2 bagian, yaitu bagian RPN dan bagian Proposal Layer, Berikut rincian dari arsitektur RPN yang digunakan:

1. Masukan berupa *feature maps* hasil dari FPN bagian satu sampai 4
2. Semua input akan melalui dua *layer Convolutional2D*
3. Lalu hasilnya akan di reshape untuk menjadi rpn_class_logits

4. Rpn_class_logits akan dilewatkan *layer* Aktivasi *softmax* untuk menjadi rpn_probs
5. Hasil dari point ke-2 akan dilewatkan satu *layer Convolutional2D* lalau akan di reshape untuk menjadi rpn_bbox
6. Semua hasil dari bagian – bagian yang ada di FPNakan menghasilkan rpn_class_logits, rpn_probs, dan rpn_bbox setelah melalui RPN
7. Lalu semua hasil dari RPN akan dilewatkan ke *layer Concatentate*

Ilustrasi arsitektur RPN dapat dilihat pada Gambar 3.10



Gambar 3.10 Ilustrasi arsitektur bagian RPN

Untuk bagian Proposal Layer inputnya adalah hasil dari *layer Concatenate* pada hasil RPN, fungsi dari bagian Proposal Layer adalah untuk menyeleksi ROI hasil dari RPN, seleksi didasarkan pada skor ROI, dan memakai non-max suppression untuk memfilter ROI yang *overlap.*

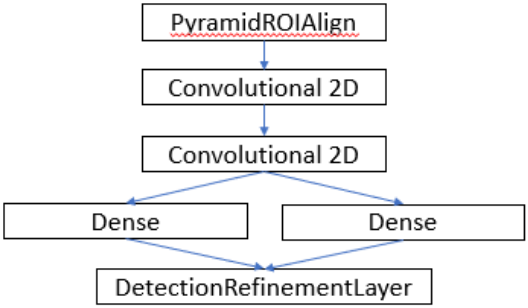
3.2.1.4 Klasifikasi

Berikut adalah detail dari tahap klasifikasi:

1. Masukan untuk klasifikasi berupa ROI hasil dari tahap RPN, *feature maps* dari Resnet101, dan image_meta. Image_meta berisi *metadata* dari gambar yaitu: id gambar, ukuran gambar original, ukuran gambar setelah melalui tahap *resize* dan *padding*, koordinat posisi gambar asli di gambar yang telah diproses, factor skala *resize*, dan semua label yang ada dalam dataset yang dipakai.
2. Lalu masukan akan melalui *layer PyramidROIAlign*, layer ini bertujuan untuk menyamakan ukuran fitur yang didapat dari FPN dengan cara melakukan pooling terhadap fitur yang bersangkutan.
3. Lalu dilanjutkan dengan *layer TimeDistributed Convolutional2D*
4. Lalu dilanjutkan dengan *layer TimeDistributed Batch Normalization.*
5. Lalu dilanjutkan dengan *layer* aktivasi relu.
6. Lalu dilanjutkan dengan layer *TimeDistributed Convolutional2D*
7. Lalu dilanjutkan dengan *layer TimeDistributed Batch Normalization*
8. Lalu dilanjutkan dengan *layer* aktivasi relu
9. Lalu dilanjutkan dengan *layer TimeDistributed Dense* untuk mendapatkan mrcnn_class_logits lalu dilanjutkan dengan *layer TimeDistributed Activation softmax* untuk mendapatkan mrcnn_probs, kedua parameter ini berisi sekumpulan angka probabilitas untuk setiap kelasnya
10. Dari poin 8 akan dilanjutkan melalui sebuah *layer TimeDistributed Dense* untuk mendapatkan mrcnn_box sebagai koordinat bounding box objek
11. Lalu mrcnn_class_logits, mrcnn_probs dan mrcnn_box, ROI dari RPN, dan *metadata* gambar

akan dilewatkan sebuah *Detection Layer* yang akan menghasilkan hasil akhir berupa ROI, Kelas dan Skor dari gambar yang dideteksi

Hasil dari bagian klasifikasi merupakan hasil akhir dari proses klasifikasi pada Mask RCNN, ilustrasi dari bagian Klasifikasi dapat dilihat pada Gambar 3.11



Gambar 3.11 Ilustrasi arsitektur bagian klasifikasi

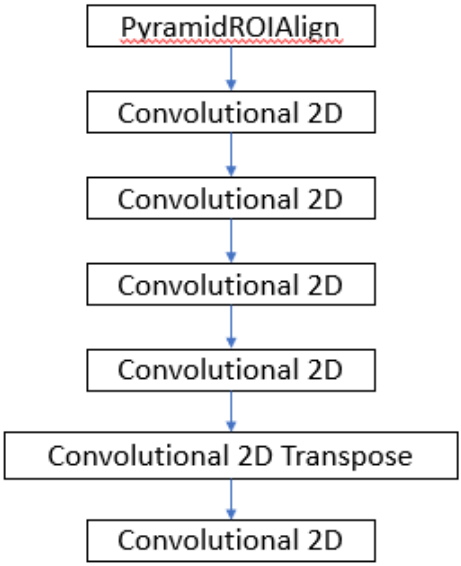
3.2.1.5 Mask

Berikut adalah detail dari tahap Mask:

1. Dimulai dengan sebuah *Detection Layer* yang berfungsi untuk memproses bounding box agar sesuai dengan ukuran objek. Dengan masukan berupa ROI dari RPN, kelas yang dihasilkan oleh bagian Klasifikasi, bounding box yang dihasilkan oleh bagian Klasifikasi dan *metadata* gambar
2. Lalu dilanjutkan dengan *layer PyramidROIAlign*
3. Lalu dilanjutkan dengan *layer TimeDistributed Batch Normalization*
4. Lalu dilanjutkan dengan *layer* Aktivasi relu
5. Lalu dilanjutkan dengan *layer TimeDistributed Convolutional2D, TimeDistributed Batch Normalizatin,* dan *layer* Aktivasi relu, diulang sebanyak 3 kali

6. Lalu dilanjutkan dengan *layer TimeDistributed Convolutional2DTranspose* dan *layer TimeDistributed Convolutional2D*

Ilustrasi dari bagian Mask dapat dilihat pada Gambar 3.12



Gambar 3.12 Ilustrasi Arsitektur bagian Mask

3.2.2 Tahap Pelatihan dan Pengujian

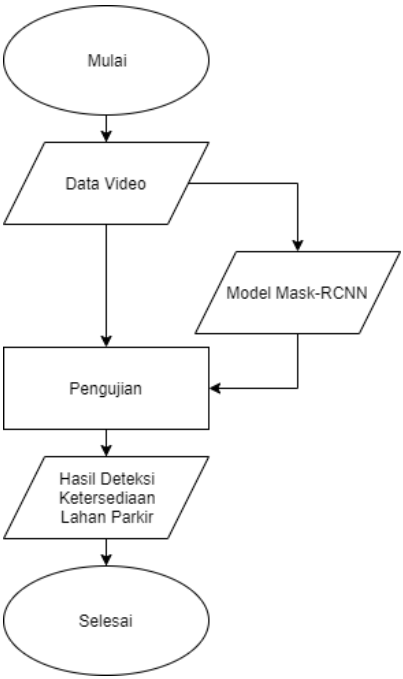
Model yang digunakan adalah model *pretrained* yang disediakan oleh Mask RCNN, model dilatih dengan menggunakan 8 GPU dengan 2 gambar per GPU dengan *learning rate* 0.001, *weight decay* 0.0001 dan *momentum* 0.9, pelatihan menggunakan dataset "Common Object in Context" (COCO) dengan data latih sebanyak 123.287 gambar dengan 80 jenis objek.

Tahap pengujian menggunakan konfigurasi 1 GPU dengan 1 gambar per GPU, proses pengujian model terhadap data uji berguna untuk mengetahui seberapa baik model dilatih. Lalu

setelah dilakukan pengujian akan mendapatkan nilai akurasi, *precision*, dan *recall.*

3.3 Desain Sistem Deteksi Ketersediaan Lahan Parkir

Model yang telah dilatih akan diimplementasi ke sebuah sistem yang berfungsi untuk mendeteksi ketersediaan lahan parkir, sistem memiliki fitur untuk membuat lahan parkir pada data video dari cctv secara otomatis.



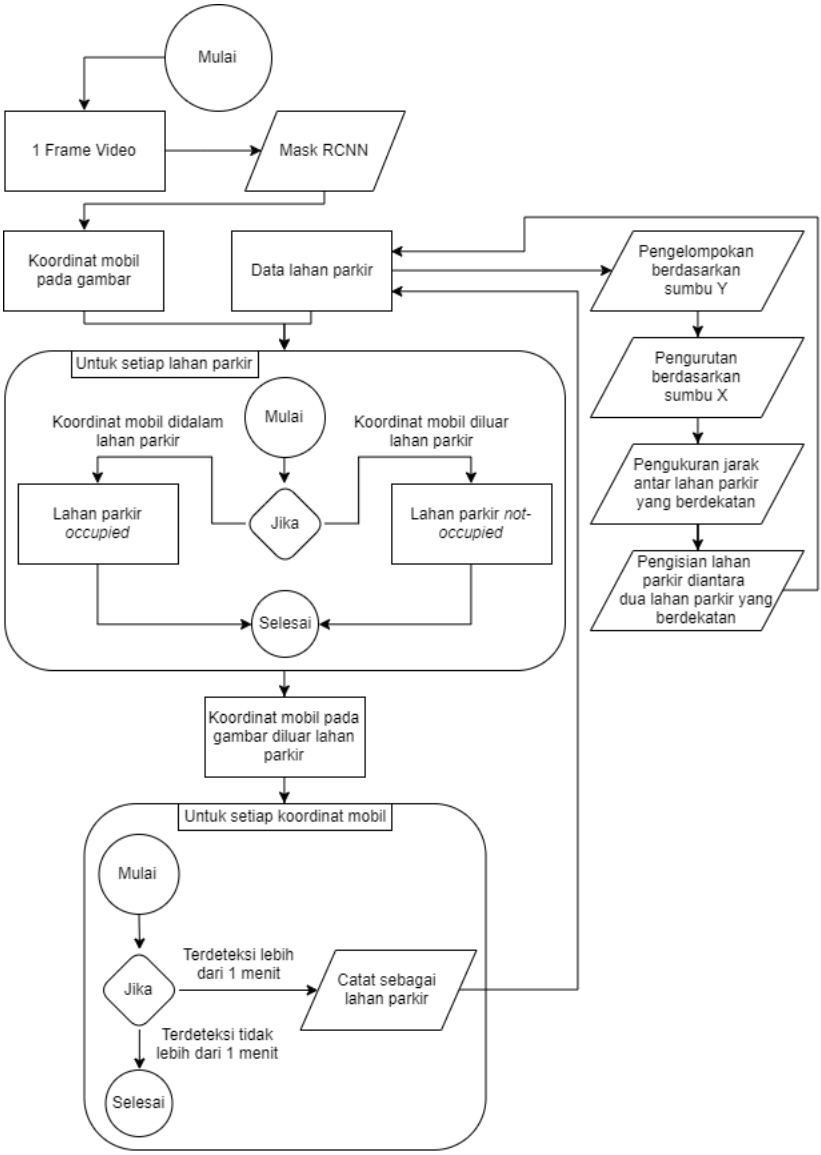
Gambar 3.13 Diagram Alir Sistem Deteksi Ketersediaan Lahan Parkir

Secara umum diagram alir dari sistem deteksi ketersediaan lahan parkir dapat dilihat pada Gambar 3.13.

Gambar yang dideteksi akan diambil dari data video setiap 20 *frame* sekali, gambar tersebut akan dilewatkan model Mask RCNN untuk mendapatkan hasil segmentasi dan klasifikasi, hasil

tersebut akan dicatat, lalu setelah 200 frame (sekitar 1 menit) hasil tersebut akan dicatat sebagai lahan parkir yang nantinya akan dideteksi tersedia atau tidaknya. Hasil lahan parkir dari tahap pertama hanya mendapatkan lahan parkir yang pada saat itu terdapat mobil saja, untuk itu dibutuhkan proses lanjutan untuk membuat lahan parkir secara otomatis pada tempat – tempat yang saat itu tidak ada mobil, selanjutnya data akan dikelompokkan berdasarkan sumbu vertikalnya, jika sumbu vertikal berdekatan maka lahan parkir tersebut dianggap satu kelompok, setelah itu tiap kelompok lahan parkir akan diurutkan berdasarkan sumbu horizontal, lalu setiap dua lahan parkir yang berdekatan akan diukur jarak antara keduanya, jika jarak melebihi lebar dari ukuran lahan parkir yang sudah di-set maka jarak tersebut akan dibagi dengan lebar dari ukuran lahan parkir untuk mendapatkan jumlah dari lahan parkir yang dapat dimuat diantara dua lahan parkir yang sudah ada, selanjutnya lahan parkir buatan akan diletakkan diantara dua lahan parkir tersebut.

Hasil koordinat ROI dari deteksi mobil menggunakan Mask RCNN akan diambil titik tengahnya lalu semua lahan parkir yang sudah didapat akan dicek apakah titik tengah tersebut ada di dalam lahan parkir, jika iya maka lahan parkir tersebut akan ditandai sebagai *occupied*. Untuk diagram alir dapat dilihat pada Gambar 3.14



Gambar 3.14 Diagram alir untuk pembuatan lahan parkir

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 deteksi ketersediaan lahan parkir diperlukan beberapa perangkat pendukung sebagai berikut.

4.1.1 Perangkat Keras

Implementasi tugas akhir ini menggunakan desktop *personal computer* (PC) HP Pavilion Gaming Desktop 690-00xx. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi Intel Core i7-8700 dengan kecepatan 3,2 GHz, *Random Access Memory* (RAM) sebesar 16 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA GeForce GTX 1060 sebesar 6 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, Keras, Numpy, Matplotlib dan Scikit-learn.

4.2 Implementasi Pembangunan Arsitektur

Pada subbab ini akan dijabarkan implementasi fungsi-fungsi pada tahap pembangunan model. Arsitektur Mask RCNN memiliki lima bagian utama, yaitu:

1. Ekstraksi Fitur
2. FPN
3. RPN
4. Klasifikasi
5. Mask

4.2.1 Ekstraksi Fitur

Seperti yang sudah dijelaskan pada Bab 3, ekstraksi fitur pada Tugas Akhir kali ini adalah menggunakan Resnet101, Resnet101 memiliki 5 tahapan arsitektur untuk tahapan pertama tercantum pada Kode Sumber 4.1, lalu untuk tahap 2 sampai 5 menggunakan fungsi conv_blovk yang tercantum pada Kode Sumber 4.2 dan fungsi identity_block yang tercantum pada Kode Sumber 4.3, untuk tahap 2 sampai 5 bisa dilihat pada Kode Sumber 4.4

```
1.  x = KL.ZeroPadding2D((3, 3))(input_image)
2.  x = KL.Conv2D(64, (7, 7), strides=(2, 2), name='con
    v1', use_bias=True)(x)
3.  x = BatchNorm(name='bn_conv1')(x, training=train_bn
    )
4.  x = KL.Activation('relu')(x)
5.  C1 = x = KL.MaxPooling2D((3, 3), strides=(2, 2), pa
    dding="same")(x)
```

Kode Sumber 4.1 Kode tahap satu Resnet101

```
1.  nb_filter1, nb_filter2, nb_filter3 = filters
2.  conv_name_base = 'res' + str(stage) + block + '_bra
    nch'
3.  bn_name_base = 'bn' + str(stage) + block + '_branch
    '
4.
5.  x = KL.Conv2D(nb_filter1, (1, 1), strides=strides,

6.                name=conv_name_base + '2a', use_bias=
    use_bias)(input_tensor)
```

```
7.  x = BatchNorm(name=bn_name_base + '2a')(x, training
    =train_bn)
8.  x = KL.Activation('relu')(x)
9.
10. x = KL.Conv2D(nb_filter2, (kernel_size, kernel_size
    ), padding='same',
11.               name=conv_name_base + '2b', use_bias=
    use_bias)(x)
12. x = BatchNorm(name=bn_name_base + '2b')(x, training
    =train_bn)
13. x = KL.Activation('relu')(x)
14.
15. x = KL.Conv2D(nb_filter3, (1, 1), name=conv_name_ba
    se +
16.               '2c', use_bias=use_bias)(x)
17. x = BatchNorm(name=bn_name_base + '2c')(x, training
    =train_bn)
18.
19. shortcut = KL.Conv2D(nb_filter3, (1, 1), strides=st
    rides,
20.                      name=conv_name_base + '1', use
    _bias=use_bias)(input_tensor)
21. shortcut = BatchNorm(name=bn_name_base + '1')(short
    cut, training=train_bn)
22.
23. x = KL.Add()([x, shortcut])
24. x = KL.Activation('relu', name='res' + str(stage) +
     block + '_out')(x)
```

Kode Sumber 4.2 Fungsi conv_block

Baris 1 berguna untuk spesifikasi *filter* untuk *layer Convolutional2D*, baris 2 dan 3 berfungsi untuk penamaan *layer*, baris 4 – 24 masing – masing adalah penambahan *layer* yang dibutuhkan.

```
1.  nb_filter1, nb_filter2, nb_filter3 = filters
2.  conv_name_base = 'res' + str(stage) + block + '_bra
    nch'
3.  bn_name_base = 'bn' + str(stage) + block + '_branch
    '
```

```
4.
5.  x = KL.Conv2D(nb_filter1, (1, 1), name=conv_name_ba
    se + '2a',
6.                use_bias=use_bias)(input_tensor)
7.  x = BatchNorm(name=bn_name_base + '2a')(x, training
    =train_bn)
8.  x = KL.Activation('relu')(x)
9.
10. x = KL.Conv2D(nb_filter2, (kernel_size, kernel_size
    ), padding='same',
11.               name=conv_name_base + '2b', use_bias=
    use_bias)(x)
12. x = BatchNorm(name=bn_name_base + '2b')(x, training
    =train_bn)
13. x = KL.Activation('relu')(x)
14.
15. x = KL.Conv2D(nb_filter3, (1, 1), name=conv_name_ba
    se + '2c',
16.               use_bias=use_bias)(x)
17. x = BatchNorm(name=bn_name_base + '2c')(x, training
    =train_bn)
18.
19. x = KL.Add()([x, input_tensor])
20. x = KL.Activation('relu', name='res' + str(stage) +
     block + '_out')(x)
```

Kode Sumber 4.3 Fungsi identity_block

Baris 1 berguna untuk spesifikasi *filter* untuk *layer Convolutional2D*, baris 2 dan 3 berfungsi untuk penamaan *layer*, baris 4 – 20 masing – masing adalah penambahan *layer* yang dibutuhkan.

```
1.  # Stage 2
2.  x = conv_block(x, 3, [64, 64, 256], stage=2, block=
    'a', strides=(1, 1), train_bn=train_bn)
3.  x = identity_block(x, 3, [64, 64, 256], stage=2, bl
    ock='b', train_bn=train_bn)
4.  C2 = x = identity_block(x, 3, [64, 64, 256], stage=
    2, block='c', train_bn=train_bn)
5.  # Stage 3
```

```
x = conv_block(x, 3, [128, 128, 512], stage=3, block='a', train_bn=train_bn)
x = identity_block(x, 3, [128, 128, 512], stage=3, block='b', train_bn=train_bn)
x = identity_block(x, 3, [128, 128, 512], stage=3, block='c', train_bn=train_bn)
C3 = x = identity_block(x, 3, [128, 128, 512], stage=3, block='d', train_bn=train_bn)
# Stage 4
x = conv_block(x, 3, [256, 256, 1024], stage=4, block='a', train_bn=train_bn)
block_count = {"resnet50": 5, "resnet101": 22}[architecture]
for i in range(block_count):
    x = identity_block(x, 3, [256, 256, 1024], stage=4, block=chr(98 + i), train_bn=train_bn)
C4 = x
# Stage 5
x = conv_block(x, 3, [512, 512, 2048], stage=5, block='a', train_bn=train_bn)
x = identity_block(x, 3, [512, 512, 2048], stage=5, block='b', train_bn=train_bn)
C5 = x = identity_block(x, 3, [512, 512, 2048], stage=5, block='c', train_bn=train_bn)
```

Kode Sumber 4.4 Tahap 2 sampai 5 Resnet101

4.2.2 Feature Pyramid Network

Sesuai dengan yang telah dijelaskan pada Bab 3, *Feature Pyramid Network* (FPN) memiliki 4 bagian dan 1 tambahan, kode untuk semua bagian FPN bias dilihat pada Kode Sumber 4.5

```
P5 = KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (1, 1), name='fpn_c5p5')(C5)
P4 = KL.Add(name="fpn_p4add")([
    KL.UpSampling2D(size=(2, 2), name="fpn_p5upsampled")(P5),
    KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (1, 1), name='fpn_c4p4')(C4)])
P3 = KL.Add(name="fpn_p3add")([
```

```
6.      KL.UpSampling2D(size=(2, 2), name="fpn_p4upsamp
    led")(P4),
7.      KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (1, 1),
     name='fpn_c3p3')(C3)])
8.  P2 = KL.Add(name="fpn_p2add")([
9.      KL.UpSampling2D(size=(2, 2), name="fpn_p3upsamp
    led")(P3),
10.     KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (1, 1),
     name='fpn_c2p2')(C2)])
11. P2 = KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (3, 3)
    , padding="SAME", name="fpn_p2")(P2)
12. P3 = KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (3, 3)
    , padding="SAME", name="fpn_p3")(P3)
13. P4 = KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (3, 3)
    , padding="SAME", name="fpn_p4")(P4)
14. P5 = KL.Conv2D(config.TOP_DOWN_PYRAMID_SIZE, (3, 3)
    , padding="SAME", name="fpn_p5")(P5)
15. P6 = KL.MaxPooling2D(pool_size=(1, 1), strides=2, n
    ame="fpn_p6")(P5)
```

Kode Sumber 4.5 Implementasi FPN

4.2.3 Region Proposal Network

Seperti yang telah dijelaskan pada Bab 3, *Region Proposal Network* (RPN) memiliki 2 bagian, bagian pertama dapat dilihat pada Kode Sumber 4.6, dan bagian kedua dapat dilihat pada Kode Sumber 4.7

```
1.  input_feature_map = KL.Input(shape=[None, None, dep
    th], name="input_rpn_feature_map")
2.  shared = KL.Conv2D(512, (3, 3), padding='same', act
    ivation='relu', strides=anchor_stride, name='rpn_co
    nv_shared')(feature_map)
3.
4.  # Anchor Score. [batch, height, width, anchors per
    location * 2].
5.  x = KL.Conv2D(2 * anchors_per_location, (1, 1), pad
    ding='valid', activation='linear', name='rpn_class_
    raw')(shared)
6.
7.  # Reshape to [batch, anchors, 2]
```

```
8.  rpn_class_logits = KL.Lambda(lambda t: tf.reshape(t
    , [tf.shape(t)[0], -1, 2]))(x)
9.
10. # Softmax on last dimension of BG/FG.
11. rpn_probs = KL.Activation("softmax", name="rpn_clas
    s_xxx")(rpn_class_logits)
12.
13. # Bounding box refinement. [batch, H, W, anchors pe
    r location * depth]
14. # where depth is [x, y, log(w), log(h)]
15. x = KL.Conv2D(anchors_per_location * 4, (1, 1), pad
    ding="valid", activation='linear', name='rpn_bbox_p
    red')(shared)
16.
17. # Reshape to [batch, anchors, 4]
18. rpn_bbox = KL.Lambda(lambda t: tf.reshape(t, [tf.sh
    ape(t)[0], -1, 4]))(x)
```

Kode Sumber 4.6 Bagian pertama RPN

```
1.  # Box Scores. Use the foreground class confidence.
    [Batch, num_rois, 1]
2.  scores = inputs[0][:, :, 1]
3.  # Box deltas [batch, num_rois, 4]
4.  deltas = inputs[1]
5.  deltas = deltas * np.reshape(self.config.RPN_BBOX_S
    TD_DEV, [1, 1, 4])
6.  # Anchors
7.  anchors = inputs[2]
8.
9.  # Improve performance by trimming to top anchors by
     score
10. # and doing the rest on the smaller subset.
11. pre_nms_limit = tf.minimum(self.config.PRE_NMS_LIMI
    T, tf.shape(anchors)[1])
12. ix = tf.nn.top_k(scores, pre_nms_limit, sorted=True
    , name="top_anchors").indices
13. scores = utils.batch_slice([scores, ix], lambda x,
    y: tf.gather(x, y), self.config.IMAGES_PER_GPU)
14. deltas = utils.batch_slice([deltas, ix], lambda x,
    y: tf.gather(x, y), self.config.IMAGES_PER_GPU)
```

```
15. pre_nms_anchors = utils.batch_slice([anchors, ix],
    lambda a, x: tf.gather(a, x), self.config.IMAGES_PE
    R_GPU, names=["pre_nms_anchors"])
16.
17. # Apply deltas to anchors to get refined anchors.
18. # [batch, N, (y1, x1, y2, x2)]
19. boxes = utils.batch_slice([pre_nms_anchors, deltas]
    , lambda x, y: apply_box_deltas_graph(x, y), self.c
    onfig.IMAGES_PER_GPU, names=["refined_anchors"])
20.
21. # Clip to image boundaries. Since we're in normaliz
    ed coordinates,
22. # clip to 0..1 range. [batch, N, (y1, x1, y2, x2)]

23. window = np.array([0, 0, 1, 1], dtype=np.float32)
24. boxes = utils.batch_slice(boxes, lambda x: clip_box
    es_graph(x, window), self.config.IMAGES_PER_GPU, na
    mes=["refined_anchors_clipped"])
```

Kode Sumber 4.7 Bagian kedua RPN

Baris 1 berfungsi untuk mendapatkan skor dari bounding box, baris 4 dan 5 berfungsi untuk mendapatkan delta dari bounding box, baris 7 berfungsi untuk mengambil *anchor*, baris 11 sampai 15 befungsi untuk menambah performa dengan cara hanya mengambil 6000 *anchor* dengan skor terbaik, baris 19 berfungsi untuk mendapatkan bounding box yang sudah sesuai dengan objek, baris 23 dan 24 berfungsi untuk memotong bagian samping dari gambar.

4.2.4 Klasifikasi

1. Seperti yang sudah dijelaskan pada Bab 3, Klasifikasi memiliki 2 bagian yaitu bagian Klasifikasi yang menghasilkan data prediksi mentah, dan bagian *Detection Layer* yang menghasilkan data prediksi akhir. Kode untuk bagian Klasifikasi dapat dilihat di
   `# Shape: [batch, num_rois, POOL_SIZE, POOL_SIZE, ch annels]`

```
2.  x = PyramidROIAlign([pool_size, pool_size], name="r
    oi_align_classifier")([rois, image_meta] + feature_
    maps)
3.  # Two 1024 FC layers (implemented with Conv2D for c
    onsistency)
4.  x = KL.TimeDistributed(KL.Conv2D(fc_layers_size, (p
    ool_size, pool_size), padding="valid"), name="mrcnn
    _class_conv1")(x)
5.  x = KL.TimeDistributed(BatchNorm(), name='mrcnn_cla
    ss_bn1')(x, training=train_bn)
6.  x = KL.Activation('relu')(x)
7.  x = KL.TimeDistributed(KL.Conv2D(fc_layers_size, (1
    , 1)), name="mrcnn_class_conv2")(x)
8.  x = KL.TimeDistributed(BatchNorm(), name='mrcnn_cla
    ss_bn2')(x, training=train_bn)
9.  x = KL.Activation('relu')(x)
10. shared = KL.Lambda(lambda x: K.squeeze(K.squeeze(x,
     3), 2), name="pool_squeeze")(x)
11. mrcnn_class_logits = KL.TimeDistributed(KL.Dense(nu
    m_classes), name='mrcnn_class_logits')(shared)
12. mrcnn_probs = KL.TimeDistributed(KL.Activation("sof
    tmax"), name="mrcnn_class")(mrcnn_class_logits)
13. x = KL.TimeDistributed(KL.Dense(num_classes * 4, ac
    tivation='linear'),name='mrcnn_bbox_fc')(shared)
14. s = K.int_shape(x)
15. mrcnn_bbox = KL.Reshape((s[1], num_classes, 4), nam
    e="mrcnn_bbox")(x)
```

Kode Sumber 4.8 dan kode untuk bagian *Detection Layer* dapat dilihat pada Kode Sumber 4.9

```
16. # Shape: [batch, num_rois, POOL_SIZE, POOL_SIZE, ch
    annels]
17. x = PyramidROIAlign([pool_size, pool_size], name="r
    oi_align_classifier")([rois, image_meta] + feature_
    maps)
18. # Two 1024 FC layers (implemented with Conv2D for c
    onsistency)
19. x = KL.TimeDistributed(KL.Conv2D(fc_layers_size, (p
    ool_size, pool_size), padding="valid"), name="mrcnn
    _class_conv1")(x)
```

```
x = KL.TimeDistributed(BatchNorm(), name='mrcnn_class_bn1')(x, training=train_bn)
x = KL.Activation('relu')(x)
x = KL.TimeDistributed(KL.Conv2D(fc_layers_size, (1, 1)), name="mrcnn_class_conv2")(x)
x = KL.TimeDistributed(BatchNorm(), name='mrcnn_class_bn2')(x, training=train_bn)
x = KL.Activation('relu')(x)
shared = KL.Lambda(lambda x: K.squeeze(K.squeeze(x, 3), 2), name="pool_squeeze")(x)
mrcnn_class_logits = KL.TimeDistributed(KL.Dense(num_classes), name='mrcnn_class_logits')(shared)
mrcnn_probs = KL.TimeDistributed(KL.Activation("softmax"), name="mrcnn_class")(mrcnn_class_logits)
x = KL.TimeDistributed(KL.Dense(num_classes * 4, activation='linear'),name='mrcnn_bbox_fc')(shared)
s = K.int_shape(x)
mrcnn_bbox = KL.Reshape((s[1], num_classes, 4), name="mrcnn_bbox")(x)
```

Kode Sumber 4.8 Bagian Klasifikasi

```
rois = inputs[0]
mrcnn_class = inputs[1]
mrcnn_bbox = inputs[2]
image_meta = inputs[3]

# Get windows of images in normalized coordinates. Windows are the area
# in the image that excludes the padding.
# Use the shape of the first image in the batch to normalize the window
# because we know that all images get resized to the same size.
m = parse_image_meta_graph(image_meta)
image_shape = m['image_shape'][0]
window = norm_boxes_graph(m['window'], image_shape[:2])

# Run detection refinement graph on each item in the batch
```

```
15. detections_batch = utils.batch_slice([rois, mrcnn_c
    lass, mrcnn_bbox, window], lambda x, y, w, z: refin
    e_detections_graph(x, y, w, z, self.config), self.c
    onfig.IMAGES_PER_GPU)
16.
17. # Reshape output
18. # [batch, num_detections, (y1, x1, y2, x2, class_id
    , class_score)] in
19. # normalized coordinates
20. return tf.reshape(detections_batch, [self.config.BA
    TCH_SIZE, self.config.DETECTION_MAX_INSTANCES, 6])
```

Kode Sumber 4.9 Bagian *DetectionLayer*

Seperti yang telah dijelaskan pada Bab 3, bagian Klasifikasi akan menghasilkan mrcnn_class_logits pada baris 16, mrcnn_probs pada barsi 17, dan mrcnn_box pada baris 25

4.2.5 Mask

Kode untuk bagian Mask sesuai dengan yang sudah dijelaskan pada Bab 3 dapat dilihat pada Kode Sumber 4.10

```
1.  # Shape: [batch, num_rois, MASK_POOL_SIZE, MASK_POO
    L_SIZE, channels]
2.  x = PyramidROIAlign([pool_size, pool_size],name="ro
    i_align_mask")([rois, image_meta] + feature_maps)
3.  # Conv layers
4.  x = KL.TimeDistributed(KL.Conv2D(256, (3, 3), paddi
    ng="same"), name="mrcnn_mask_conv1")(x)
5.  x = KL.TimeDistributed(BatchNorm(), ame='mrcnn_mask
    _bn1')(x, training=train_bn)
6.  x = KL.Activation('relu')(x)
7.  x = KL.TimeDistributed(KL.Conv2D(256, (3, 3), paddi
    ng="same"), name="mrcnn_mask_conv2")(x)
8.  x = KL.TimeDistributed(BatchNorm(), name='mrcnn_mas
    k_bn2')(x, training=train_bn)
9.  x = KL.Activation('relu')(x)
10. x = KL.TimeDistributed(KL.Conv2D(256, (3, 3), paddi
    ng="same"), name="mrcnn_mask_conv3")(x)
```

```
11. x = KL.TimeDistributed(BatchNorm(), name='mrcnn_mas
    k_bn3')(x, training=train_bn)
12. x = KL.Activation('relu')(x)
13. x = KL.TimeDistributed(KL.Conv2D(256, (3, 3), paddi
    ng="same"), name="mrcnn_mask_conv4")(x)
14. x = KL.TimeDistributed(BatchNorm(), name='mrcnn_mas
    k_bn4')(x, training=train_bn)
15. x = KL.Activation('relu')(x)
16. x = KL.TimeDistributed(KL.Conv2DTranspose(256, (2,
    2), strides=2, activation="relu"), name="mrcnn_mask
    _deconv")(x)
17. x = KL.TimeDistributed(KL.Conv2D(num_classes, (1, 1
    ), strides=1, activation="sigmoid"), name="mrcnn_ma
    sk")(x)
```

Kode Sumber 4.10 Bagian Mask

4.3 Implementasi Evaluasi CNN

Setelah membangun arsitektur *Mask Region-based Convolutional Neural Network*, dilakukan proses pengujian. Data uji diambil dari dataset CNRPark.

CNRPark menyediakan file csv yang berisi koordinat lahan parkir dan file txt yang berisi label dari lahan parkir tersebut, langkah pertama adalah memproses file csv dan file txt agar dapat digunakan dengan mudah saat pengujian.

```
1.  lot = {}
2.  coord = {}
3.  base_path = 'datasets/CNREXT/'
4.  images = glob(base_path + '*/*/*/*/*')
5.  csvs = glob(base_path + '*.csv')
6.  txts = glob(base_path + 'labels/camera*.txt')
```

Kode Sumber 4.11 *Load* file csv dan txt

Baris 1 – 6 berfungsi untuk menyiapkan *variable* yang akan digunakan untuk memproses data csv dan txt.

```
1.  for i in range(len(txts)):
```

```
2.      txt = txts[i]
3.      txt_data = open(txt, 'r').read()
4.      txt_data = txt_data.split('\n')
5.
6.      for item in txt_data:
7.          try:
8.              path, label = item.split(' ')
9.              weather, date, camera, name = path.spli
    t('/')
10.             name = name.split('.jpg')[0]
11.             _, name_date, name_time, _, slotId = na
    me.split('_')
12.             name_time = name_time.split('.')
13.             name_time = ''.join(name_time)
14.             true_name = name_date + '_' + name_time
     + '.jpg'
15.             true_path = base_path + 'image/' + weat
    her + '/' + date + '/' + camera + '/' + true_name
16.
17.             if true_path not in lot:
18.                 lot[true_path] = []
19.             temp = {
20.                 'camera': str(int(camera.split('cam
    era')[-1])-1),
21.                 'slotid': str(slotId),
22.                 'label': str(label)
23.             }
24.             lot[true_path].append(temp)
25.         except:
26.             pass
```

Kode Sumber 4.12 Proses file txt

Baris 1 – 26 berfungsi untuk memproses data txt, data txt akan di dimuat ke dalam sebuah *variable* sebagai data teks pada baris 2 – 3, lalu data teks akan dipisahkan dengan pemisah '\n' untuk mendapatkan data perbaris, lalu data perbaris akan dipisahkan menjadi kode lokasi dan kelas, kode lokasi berisi cuaca, tanggal kamera, dan kode file yang akan dipisahkan, dari kode file akan diambil tanggal, waktu, dan kode identifikasi, lalu dari tanggal dan waktu akan dibuat menjadi nama file yang sesuai

dengan dataset yang ada, lalu akan digabungkan dengan cuaca, tanggal, dan kamera untuk mendapatkan letak file yang sebenarnya, lalu letak file, label, dan kamera akan disimpan ke sebuah *variable*.

```
1.  for i in range(len(csvs)):
2.      csv = csvs[i]
3.      csv_data = pd.read_csv(csv)
4.
5.      coord[str(i)] = {}
6.
7.      for j in range(len(csv_data)):
8.          slotId = str(csv_data.iloc[j]['SlotId'])
9.          x = int(csv_data.iloc[j]['X'])
10.         y = int(csv_data.iloc[j]['Y'])
11.         w = int(csv_data.iloc[j]['W'])
12.         h = int(csv_data.iloc[j]['H'])
13.         coord[str(i)][slotId] = {
14.             "x": x,
15.             "y": y,
16.             "w": w,
17.             "h": h,
18.             'x1': int(x * 1000/2592),
19.             'y1': int(y * 750/1944),
20.             'x2': int((x + w) * 1000/2592),
21.             'y2': int((y + h) * 750/1944)
22.         }
```

Kode Sumber 4.13 Proses file csv

Baris 1 – 22 berfungsi untk memproses data csv, pada baris 2 – 3 data csv akan dimuat ke dalam sebuah *variable* dengan menggunakan *library* pandas, data akan diiterasi untuk mendapatkan titik tengah dari bounding box dan lebar dan tinggi dari bounding box tersebut, dikarenakan gambar yang disediakan CNRPark sudah di rubah menjadi ukuran 1000x750 namun informasi bounding box masih mengacu pada ukuran 2592x1944 maka diperlukan proses untuk merubah informasi bounding box

menjadi sesuai dengan gambar dengan ukuran 1000x750, lalu semua informasi akan disimpan kedalam sebuah *variable*.

Langkah selanjutnya adalah menyiapkan konfigurasi yang akan digunakan untuk pengujian model

```
1.  class InferenceConfig(coco.CocoConfig):
2.      GPU_COUNT = 1
3.      IMAGES_PER_GPU = 1
4.
5.  config = InferenceConfig()
6.  config.display()
```

Kode Sumber 4.14 Konfigurasi untuk pengujian model

Untuk pengujian model konfigurasinya adalah menggunakan 1 GPU dan gambar yang diproses adalah 1 gambar per GPU, hal ini dilakukan dikarenakan GPU yang tersedia hanya 1 dan cukup 1 gambar yang dideteksi pada 1 waktu.

Selanjutnya model akan dibentuk dan *pretrained weights* yang sudah disediakan akan dimuat kedalam model tersebut.

```
1.  model = modellib.MaskRCNN(mode = "inference", model
    _dir = 'logs/', config = config)
2.  model.load_weights('mask_rcnn_coco.h5', by_name = T
    rue)
```

Kode Sumber 4.15 Membentuk model dan memuat *weights*

Selanjutnya model dapat digunakan untuk pengujian dengan memanggil fungsi *detect*

```
1.  image = cv2.imread({lokasi gambar})
2.  result = model.detect([image])
```

Kode Sumber 4.16 Deteksi objek Mask RCNN

Pada baris 1 gambar akan dimuat menggunakan *library opencv* lalu akan dideteksi objek yang ada dalam gambar tersebut menggunakan fungsi *detect* yang akan menghasilkan label, skor,

dan bounding box untuk setiap objek yang ada dalam gambar tersebut.

Semua data gambar akan melalui proses deteksi, hasil deteksi dan label data asli akan dicatat, cara untuk menghubungkan antara hasil deteksi dan data asli adalah dengan cara mencatat bounding box dari data asli lalu mencatat titik tengah dari semua hasil deteksi lalu jika ada hasil deteksi yang ada di dalam bounding box maka lokasi bounding box tersebut akan ditandai sebagai label 1 pada hasil deteksi yang akan dicatat, hal ini dilakukan berulangkali untuk semua bounding box dari data asli.

```
report = classification_report(true, detected)
print(report)
```

Kode Sumber 4.17 Membuat laporan hasil uji

Lalu menggunakan fungsi classification_report untuk menghasilkan akurasi, presisi dan *recall*.

4.4 Implementasi Sistem Deteksi Ketersediaan Lahan Parkir

Sistem deteksi ketersediaan lahan parkir diimplementasi menggunakan bahasa pemrograman *python* dan menggunakan arsitektur *Mask Region-based Convolutional Neural Network*.

Langkah pertama adalah mempersiapkan arsitektur Mask RCNN untuk digunakan.

```
class InferenceConfig(coco.CocoConfig):
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

config = InferenceConfig()
config.display()

# modelling
model = modellib.MaskRCNN(mode = "inference", model
_dir = 'logs/', config = config)
```

```
10.
11. model.load_weights('mask_rcnn_coco.h5', by_name = T
    rue)
12.
13. class_names = ['BG', 'person', 'bicycle', 'car', 'm
    otorcycle', 'airplane', ...]
```

Kode Sumber 4.18 Mempersiapkan arsitektur Mask RCNN

Baris 1-3 adalah konfigurasi yang digunakan oleh Mask RCNN, konfigurasi yang digunakan adalah menggunakan 1 GPU dan 1 gambar untuk setiap GPU, baris 9 berfungsi untuk membuat model Mask RCNN, baris 11 berfungsi untuk memuat *pretrained weights* ke dalam memori, baris 13 berisi kelas yang ada dalam model yang telah di latih.

```
1.  parking_lot = Tracker()
2.  capture = cv2.VideoCapture(video)
3.  counter = 0
4.  while True:
5.      ret, frame = capture.read()
6.      # time.sleep(1/30)
7.      if not ret:
8.          break
9.      counter += 1
10.     if counter == 1:
11.         worker(frame)
12.
13.     if counter % 20 == 0:
14.         cloned = deepcopy(frame)
15.         cloned[:,:big_ROI[0]] = 0
16.         cloned[:,big_ROI[2]:] = 0
17.         cloned[:big_ROI[1],:] = 0
18.         cloned[big_ROI[3]:,:] = 0
19.         t = Thread(target = worker, args = (cloned,
    ))
20.         t.start()
```

Kode Sumber 4.19 Mengambil gambar video

Pada baris 1 di inisialisasi sebuah *module* yang berguna untuk mencatat lahan parkir yang menjadi tujuan utama dari Tugas Akhir ini, barsi 2 berfungsi untuk membuat sebuah objek yang berguna untuk mengambil data dari kamera, baris 5 berfungsi untuk mengambil gambar dari objek pada baris 2, baris 12 befungsi untuk inisialisasi data untuk lahan parkir dengan cara memanggil fungsi worker, baris 14 berfungsi untuk mengambil gambar setiap 20 frame 1 kali, baris 15 – 19 befungsi untuk menghitamkan area diluar ROI utama yang sudah diset sehingga dapat mempercepat proses segmentasi dan klasifikasi, baris 20 dan 21 berfungsi untuk membuat sebuah *thread* untuk menjalankan fungsi worker yang akan menjalankan model yang telah dibuat pada *thread* tersendiri agar tidak mengganggu proses penampilan video.

```
1.  def worker(frame):
2.      global master
3.      results = model.detect([frame])
4.      master = results[0]
```

Kode Sumber 4.20 Fungsi worker

Pada fungsi worker baris 3 berfungsi untuk menjalankan model Mask RCNN lalu hasilnya akan disimpan pada *global variable* master.

```
1.  sub = deepcopy(master)
2.  current = []
3.  for i in range(len(sub['rois'])):
4.      if class_names[sub['class_ids'][i]] == 'car':
5.          y1, x1, y2, x2 = sub['rois'][i]
6.          current.append([x1, y1, x2, y2])
7.  space, temporary = parking_lot.update(current)
```

Kode Sumber 4.21 Menjalankan *module* tracker

Pada baris 1 berfungsi untuk menyalin isi dari *variable* master selanjutnya hasil dari Mask RCNN di filter hanya diambil yang dideteksi sebagai mobil, lalu koordinatnya dicatat ke *variable* current, lalu *variable* tersebut akan digunakan sebagai parameter untuk fungsi update pada *module* tracker, fungsi ini akan dipanggil untuk setiap frame.

4.4.1 Implementasi *module* Tracker

Module Tracker berfungsi untuk mencatat dan memproses semua data hasil deteksi dari Mask RCNN.

Untuk *module* Tracker ada beberapa *variable* yang perlu di inisialisasi

```
1.  def __init__(self):
2.      self.space = []
3.      self.temporary = []
4.
5.      self.same_space_threshold = 25
6.      self.max_distance = 50
7.      self.square_size = 50
8.
9.      self.fps = 20
10.     self.frame_detect = 60 * self.fps
11.     self.max_age = 300 * self.fps
12.     self.max_occupied_age = 10 * self.fps
```

Kode Sumber 4.22 Inisialisasi module tracker

Self.space berfungsi sebagai wadah untuk lahan parkir, self.temporary berfungsi sebagai wadah untuk hasil deteksi yang masih belum berumur 1 menit, self.same_space_threshold adalah jarak antara dua titik hasil deteksi yang masih dianggap sama, self.max_distance adalah seberapa jauh jarak sumbu Y dari lahan parkir yang ada untuk dideteksi sebagai 1 kelompok, self.square_size adalah setengah dari lebar dan panjang kotak untuk lahan parkir, self.fps adalah *frame rate* dari video yang sedang dideteksi, self.frame_detect adalah berapa lama titik hasil deteksi harus dideteksi agar dapat menjadi lahan parkir,

self.max_age adalah berapa lama sebuah hasil dari Mask RCNN yang telah dicatat pada self.temporary disimpan sebelum data tersebut dihapus, self.max_occupied_age adalah jumlah frame yang dibutuhkan untuk sebuah lahan parkir menjadi *not-occupied* setelah tidak ada mobil yang terdeteksi pada lahan parkir tersebut.

```
1.  def update(self, current):
2.      self._interpolate_data()
3.      new_current = self._check_space(current)
4.      self._check_temporary(new_current)
5.      self._check_max_age_occupied()
```

Kode Sumber 4.23 Fungsi update

Fungsi update berguna untuk menjalankan semua fungsi yang perlu dijalankan pada *module* Tracker

4.4.1.1 Implementasi fungsi interpolate_data

Fungsi interpolate_data digunakan untuk mengisi lahan kosong diantara lahan parkir yang masih cukup untuk diisi dengan lahan parkir.

```
1.  clusters = []
2.
3.  for item in self.space:
4.      temp = deepcopy(item)
5.
6.      if len(clusters) == 0:
7.          clusters.append([temp])
8.          continue
9.
10.     marker = 0
11.     for cluster in clusters:
12.         rata_y = sum([x['coord'][1] for x in cluste
    r]) / len(cluster)
13.         dist = abs(rata_y - item['coord'][1])
14.
15.         if dist < self.max_distance:
```

```
            cluster.append(temp)
            marker = 1
            break

    if marker == 1:
        continue
    else:
        clusters.append([temp])
```

Kode Sumber 4.24 Pengelompokan berdasarkan sumbu Y

Pada bagian ini lahan parkir akan dikelompokkan berdasarkan sumbu Y dengan maksimum jarak sumbu Y sesuai dengan yang telah diset pada *variable* self.max_distance.

```
for item in clusters:
    item = sorted(item, key=lambda x: x['coord'][0]
)

    for i in range(1, len(item)):
        if((item[i]['coord'][0] - item[i-
1]['coord'][0]) >= (4*self.square_size)):
            empty_space = (item[i]['coord'][0] - se
lf.square_size) - (item[i-
1]['coord'][0] + self.square_size)
            # print(item[i-
1]['id'], item[i]['id'], empty_space)
            space_avail = math.floor(empty_space /
(self.square_size*2))
            lot_space = math.floor(empty_space / sp
ace_avail)
            for j in range(1, space_avail+1):
                middle =  [item[i-
1]['coord'][0] + (lot_space * j), item[i-
1]['coord'][1]]
                temp = {
                    'id': len(self.space),
                    'coord': middle,
                    'square': [middle[0] - self.squ
are_size, middle[1] - self.square_size, middle[0] +
 self.square_size, middle[1] + self.square_size],
```

```
                    'status': 0,
                    'age': self.max_occupied_age
                }

                for l in range(len(self.space)):
                    if self.space[l]['square'][0] <
     middle[0] < self.space[l]['square'][2] and self.sp
    ace[l]['square'][1] < middle[1] < self.space[l]['sq
    uare'][3]:
                        print(str(middle) + ' insid
    e' + str(self.space[l]['square']))
                    else:
                        self.space.append(temp)
                        break
```

Kode Sumber 4.25 Membuat lahan parkir

Pada bagian ini untuk setiap kelompok lahan parkir akan diurutkan berdasarkan sumbu x dari terkecil sampai terbesar, lalu akan diukur jarak antara dua lahan parkir yang berurutan apakah cukup untuk diisi dengan lahan parkir atau tidak, jika cukup maka selanjutnya akan diukur tempat tersebut cukup diisi oleh berapa lahan parkir, setelah itu lahan parkir akan ditempatkan dengan jarak antar lahan parkir yang sama.

4.4.1.2 Implementasi fungsi check_space

Fungsi ini berguna untuk mengecek koordinat kendaraan hasil dari Mask RCNN, titik tengah dari koordinat kendaraan akan dicek apakah titik tersebut berada di salah satu lahan parkir, jika iya makan lahan parkir tersebut ditandai sebagai *occupied* lalu koordinat kendaraan tersebut dihapus dari kumpulan hasil dari Mask RCNN, jika tidak maka lahan parkir tersebut akan ditandai sebagai *not-occupied.*

```
current_tempo = deepcopy(current)
for i in range(len(self.space)):
    x1, y1, x2, y2 = self.space[i]['square']

    marker = 0 # 0 empty, 1 occupied
```

```
6.      for j in range(len(current_tempo)-1, -1, -1):
7.          x_current, y_current = self._get_center(cur
    rent_tempo[j])
8.
9.          marker = 1 if x1 < x_current < x2 and y1 <
    y_current < y2 else 0
10.
11.         if marker == 1:
12.             self.space[i]['status'] = 1
13.             current_tempo.remove(current_tempo[j])

14.
15.             break
16.
17.     if marker == 0:
18.         self.space[i]['status'] = 0
```

Kode Sumber 4.26 Fungsi check_space

4.4.1.3 Implementasi fungsi check_temporary

Fungsi ini berguna untuk mencocokkan data hasil dari Mask RCNN dengan data yang dicatat pada self.temporary.

```
1.  x_current, y_current = self._get_center(current[i])

2.  for j in range(len(self.temporary)-1, -1, -1):
3.      x_temporary, y_temporary = self.temporary[j]['c
    oord']
4.
5.      distance = self._calc_distance(x_current, y_cur
    rent, x_temporary, y_temporary)
6.      marker = 1 if distance <= self.same_space_thres
    hold else 0
```

Kode Sumber 4.27 Cek jarak

Untuk setiap data hasil dan setiap data pada self.temporary akan dicek jaraknya, jika jaraknya lebih kecil daripada self.same_space_threshold maka hasil dari Mask RCNN tersebut akan diasosiasikan dengan data self.temporary tersebut, jika tidak

maka data hasil tersebut akan dicatat sebagai data baru lalu ditambahkan ke self.temporary.

Untuk hasil yang diasosiasikan dengan data self.temporary, jika data dari self.temporary tersebut sudah terdeteksi lebih dari self.frame_detect, maka data self.temporary akan dicatat sebagai lahan parkir.

```
1.  if self.temporary[j]['detected'] >= self.frame_dete
    ct:
2.      temp = {
3.          'id': len(self.space),
4.          'coord': self.temporary[j]['coord'],
5.          'square': [self.temporary[j]['coord'][0] -
    self.square_size, self.temporary[j]['coord'][1] - s
    elf.square_size, self.temporary[j]['coord'][0] + se
    lf.square_size, self.temporary[j]['coord'][1] + sel
    f.square_size],
6.          # 'square': current[i],
7.          'status': 1,
8.          'age': self.max_occupied_age
9.      }
10.     self.temporary.remove(self.temporary[j])
```

Kode Sumber 4.28 Mencatat data dari self.temporary sebagai lahan parkir

Data yang perlu dicatat untuk lahan parkir adalah id dari lahan parkir, koordinat titik tengah dari lahan parkir, koordinat untuk kotak dari lahan parkir, status dari lahan parkir, dan umur dari lahan parkir tersebut terdeteksi *occupied*.

4.4.1.4 Implementasi fungsi check_max_age_occupied

Fungsi ini berguna untuk menghitung apakah lahan parkir yang sudah tidak ada mobil sudah melewati self.max_occupied_age, hal ini dilakukan untuk mengurangi ketidak pastian dari hasil deteksi Mask RCNN

```
1.  for i in range(len(self.space)):
2.      if self.space[i]['status'] == 0:
```

```
3.          if self.space[i]['age'] > 0:
4.              self.space[i]['status'] = 1
5.              self.space[i]['age'] -= 1
6.      else:
7.          self.space[i]['age']= self.max_occupied_age
```

Kode Sumber 4.29 Fungsi check_max_age_occupied

(Halaman ini sengaja dikosongkan)

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) HP Pavilion Gaming Desktop 690-00xx. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi perangkat keras Intel Core i7-8700 dengan kecepatan 3,2 GHz, *Random Access Memory* (RAM) sebesar 16 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA GeForce GTX 1060 sebesar 6 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 kali ini, data yang digunakan adalah data CNRPark. Dataset CNRPark memiliki gambar dari 9 kamera yang merekam lokasi parkir yang berbeda, terdapat 3 cuaca berbeda untuk setiap lokasi parkir, gambar diambil setiap 30 menit sekali selama 23 hari, dataset ini memiliki 4081 gambar, disediakan pula koordinat tempat parkir dan ketersediaannya untuk setiap gambar yang ada.

Uji coba juga akan dilakukan dengan data dari lokasi parkir departemen Informatika ITS, data ini memiliki data dari lokasi parkir saat siang dan lokasi parkir saat malam hari

5.3 Skenario Uji Coba

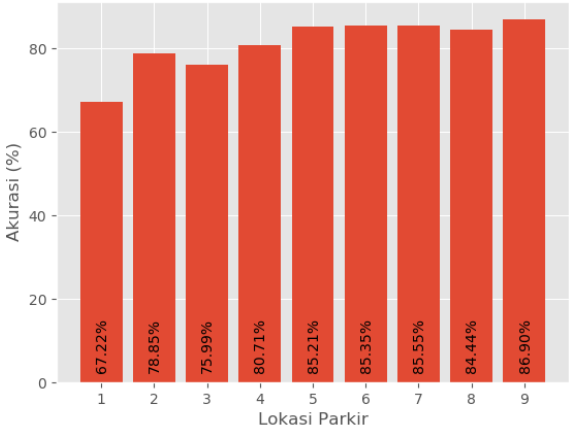
Proses uji coba berguna untuk menemukan parameter yang menghasilkan performa model yang paling optimal. Parameter yang tepat akan memberikan hasil yang lebih baik pada saat proses

uji coba. Ada 5 macam skenario uji coba dan semuanya akan dicoba pada arsitektur Mask RCNN yang telah dirancang, dan menggunakan *pretrained weights* yang dilatih dengan dataset COCO. Skenario uji coba yang akan dilakukan yaitu:

1. Uji Coba Berdasarkan Lokasi Parkir
2. Uji Coba Berdasarkan Kondisi Cuaca
3. Uji Coba Data Video Lokasi Parkir Departemen Teknik Informatika ITS
4. Uji Coba Perbandingan Arsitektur Ekstraksi Fitur
5. Uji Coba Perbandingan Metode Faster-RCNN

5.3.1 Uji Coba Berdasarkan Lokasi Parkir

Uji coba pertama adalah uji coba untuk setiap lokasi parkir yang ada dalam dataset CNRPark, uji coba ini dilakukan untuk menguji kemampuan model untuk lokasi parkir yang berbeda – beda, terdapat 9 kamera yang merekam lokasi parkir yang berbeda pada dataset CNRPark.



Gambar 5.1 Akurasi sistem berdasarkan lokasi parkir pada dataset CNRPark

Tabel 5.1 Perbandingan antara rata - rata akurasi, presisi, dan *recall* pada uji coba berdasarkan lokasi parkir

Lokasi Parkir	Akurasi	Presisi	Recall
1	67,22%	98,23%	45,47%
2	78,85%	98,89%	67,96%
3	75,99%	99,58%	57,89%
4	80,71%	99,83%	65,92%
5	85,21%	99,82%	72,74%
6	85,35%	**99,88%**	72,41%
7	85,55%	99,78%	71,14%
8	84,44%	99,59%	71,11%
9	**86,90%**	99,77%	**77,11%**

Grafik perbandingan akurasi untuk setiap lahan parkir dapat dilihat pada Gambar 5.1, tabel perbandingan akurasi, *precision*, *recall* untuk setiap lokasi parkir dapat dilihat pada Tabel 5.1.

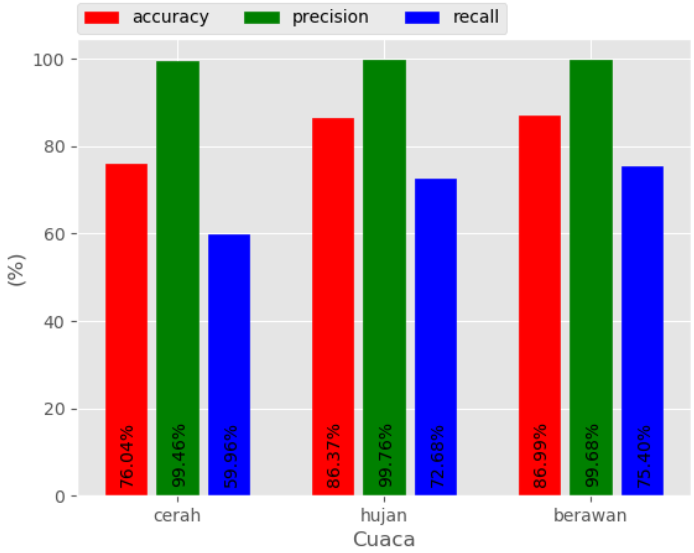
Salah satu contoh hasil deteksi dari model Mask RCNN pada tempat parkir 6 dapat dilihat pada Gambar 5.2, kotak putih adalah *groundtruth* yang disediakan oleh dataset, dan titik kuning merupakan titik tengah dari koordinat hasil deteksi dari Mask RCNN.

Gambar 5.2 Contoh hasil untuk lokasi parkir 6

5.3.2 Uji Coba Berdasarkan Kondisi Cuaca

Uji coba kedua adalah uji coba berdasarkan kondisi cuaca, uji coba ini dilakukan untuk menguji kemampuan model untuk situasi cuaca yang berbeda – beda, terdapat 3 cuaca pada dataset CNRPark, yaitu: cerah, berawan, dan hujan. Uji coba ini dilakukan terhadap semua lokasi parkir.

Grafik perbandingan akurasi sistem berdasarkan kondisi cuaca dapat dilihat pada Gambar 5.3, tabel perbandingan akurasi, presisi, *recall* untuk setiap cuaca dapat dilihat pada Tabel 5.2



Gambar 5.3 Akurasi sistem berdasarkan kondisi cuaca

Tabel 5.2 Perbandingan antara rata - rata akurasi, presisi, dan *recall* pada uji coba berdasarkan kondisi cuaca

Cuaca	Akurasi	Presisi	Recall
Cerah	76,04%	99,46%	59,96%
Hujan	86,37%	**99,76%**	72,68%
Berawan	**86,99%**	99,68%	**75,40%**

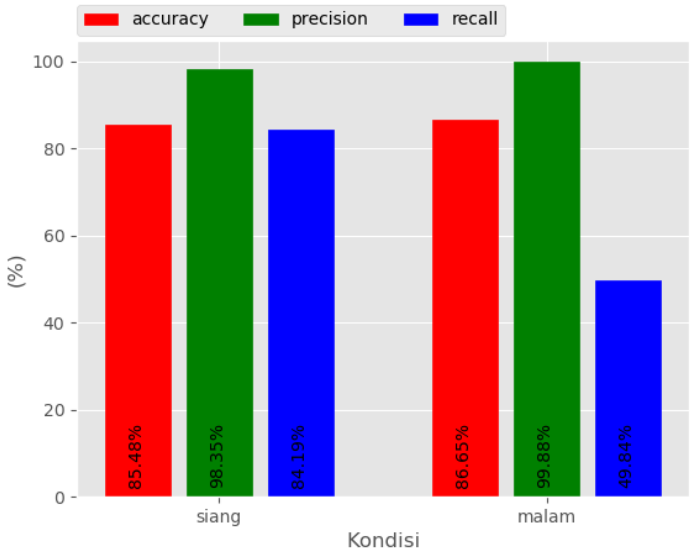
Gambar 5.4 Contoh hasil untuk cuaca hujan

Salah satu contoh hasil deteksi dari model Mask RCNN pada kondisi hujan dapat dilihat pada Gambar 5.4, kotak putih adalah *groundtruth* dari dataset, dan titik hijau adalah titik tengah dari koordinat hasil deteksi dari Mask RCNN

5.3.3 Uji Coba Data Video Lokasi Parkir Departemen Teknik Informatika ITS

Uji coba ketiga adalah uji coba pada data video dari lokasi parkir Departemen Teknik Informatika ITS, pada uji coba kali ini terdapat 2 kondisi, yaitu kondisi siang dan malam.

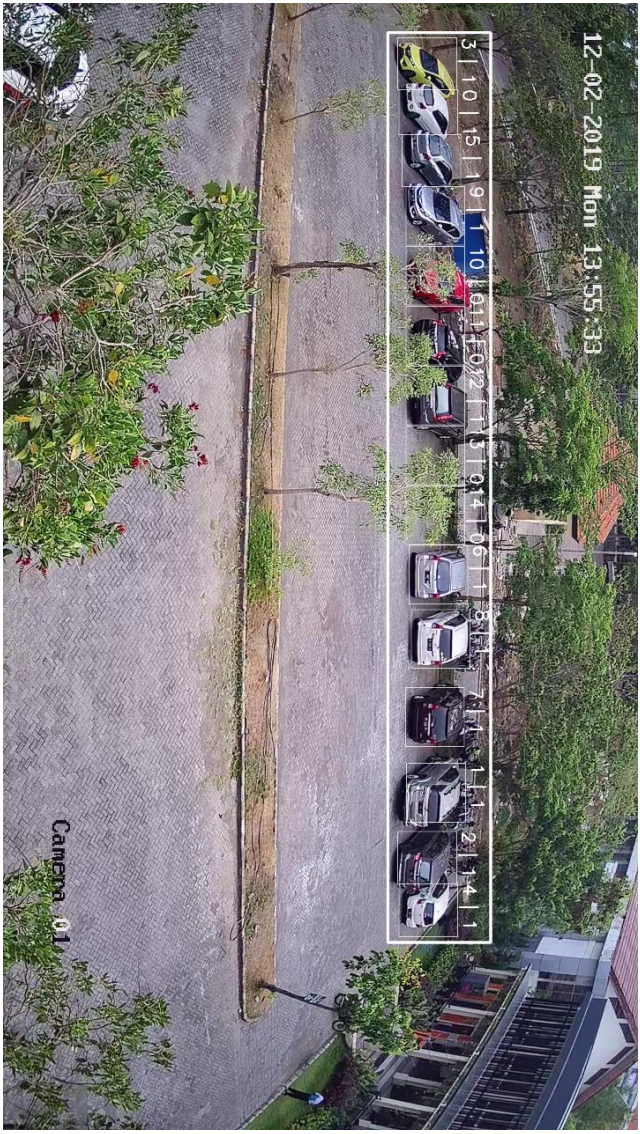
Grafik perbandingan akurasi untuk setiap kondisi dapat dilihat pada Gambar 5.5, table perbandingan akurasi, presisi, *recall* untuk setiap kondisi dapat dilihat pada Tabel 5.3.



Gambar 5.5 Akurasi untuk setiap kondisi

Tabel 5.3 Perbandingan antara rata - rata akurasi, presisi, dan *recall* pada uji coba Data Video Lokasi Parkir Departemen Informatika ITS

Kondisi	Akurasi	Presisi	Recall
Siang	85,48%	98,35%	**84,19%**
Malam	**86,65%**	**99.88%**	49.84%



Gambar 5.6 Contoh hasil untuk kondisi siang

Salah satu contoh hasil deteksi dari Mask RCNN pada data video lokasi parkir Departemen Informatika dapa dilihat pada Gambar 5.6, kotak persegi panjang besar adalah area yang dijadikan masukan ke Mask RCNN, untuk area diluar kotak persegi panjang dihitamkan, setiap kotak kecil diatasnya terdapat angka untuk *identifier* dan status dari lahan parkir tersebut, 1 untuk *occupied*, dan 0 untuk *not-occupied*.

5.3.4 Uji Coba Perbandingan Arsitektur Ekstraksi Fitur

Uji coba keempat adalah uji coba dengan mengganti arsitektur pada bagian Ekstraksi Fitur dengan menggunakan Resnet50, model dengan bagian Ekstraksi Fitur Resnet50 menggunakan *pretrained weights* yang dilatih menggunakan dataset COCO, model akan diujikan dengan dataset CNRPark dan data video lokasi parkir Departemen Informatika

Uji coba pada dataset CNRPark dibandingkan akurasi dari 9 lokasi parkir dan 3 cuaca yang ada pada dataset CNRPark. Untuk perbandingan akurasi pada data lokasi parkir CNRPark dapat dilihat pada Tabel 5.4, untuk perbandingan akurasi pada data cuaca CNRPark dapat dilihat pada Tabel 5.5

Tabel 5.4 Perbandingan akurasi uji coba arsitektur ekstraksi fitur pada data lokasi parkir CNRPark

Lokasi Parkir	Resnet101	Resnet50
1	**67,22%**	52,96%
2	**78,85%**	79,27%
3	**75,99%**	69,34%
4	**80,71%**	71,66%
5	**85,21%**	72,76%
6	**85,35%**	73,25%
7	**85,55%**	72,07%
8	**84,44%**	77,86%
9	**86,90%**	74,89%
Rata - rata	**81,14%**	71,56%

Tabel 5.5 Perbandingan akurasi uji coba arsitektur ekstraksi fitur pada data cuaca CNRPark

Cuaca	Resnet101	Resnet50
Cerah	**76,04%**	65,99%
Hujan	**86,37%**	71,12%
Berawan	**86,99%**	79,53%
Rata – rata	**83,13%**	72,21%

Selanjutnya uji coba juga dilakukan pada data video lokasi parkir Departemen Informatika, untuk perbandingan akurasi pada data video lokasi parkir Departemen Informatika dapat dilihat pada Tabel 5.6

Tabel 5.6 Perbandingan akurasi uji coba arsitektur ekstraksi fitur pada data video lokasi parkir Departemen Informatika

Kondisi	Resnet101	Resnet50
Siang	**85,48%**	65,41%
Malam	**86,65%**	81,71%
Rata – rata	**86,07%**	73,56%

5.3.5 Uji Coba Perbandingan Metode Faster-RCNN

Uji coba kelima adalah uji coba untuk membandingkan arsitektur Faster-RCNN dan arsitektur Mask-RCNN, model akan diujikan dengan dataset CNRPark. Uji coba dilakukan dengan cara membandingkan akurasi dari arsitektur Faster-RCNN dan Mask-RCNN pada 9 lokasi dan 3 cuaca dari CNRPark.

Tabel 5.7 Perbandingan akurasi uji coba arsitektur Faster-RCNN dan Mask-RCNN pada data lokasi parkir CNRPark

Lokasi Parkir	Mask-RCNN	Faster-RCNN
1	**67,22%**	44,61%
2	**78,85%**	42,74%
3	**75,99%**	47,17%
4	**80,71%**	52,69%
5	**85,21%**	57,59%
6	**85,35%**	55,63%
7	**85,55%**	58,21%
8	**84,44%**	54,15%
9	**86,90%**	52,18%
Rata - rata	**81,14%**	51,66%

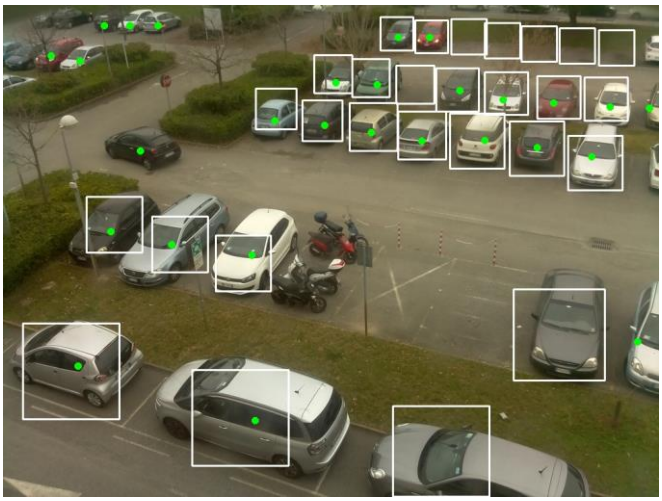
Tabel 5.8 Perbandingan akurasi uji coba arsitketur Faster-RCNN dan Mask -RCNN pada data cuaca CNRPark

Cuaca	Mask-RCNN	Faster-RCNN
Cerah	**76,04%**	47,76%
Hujan	**86,37%**	57,05%
Berawan	**86,99%**	58,05%
Rata – rata	**83,13%**	54,29%

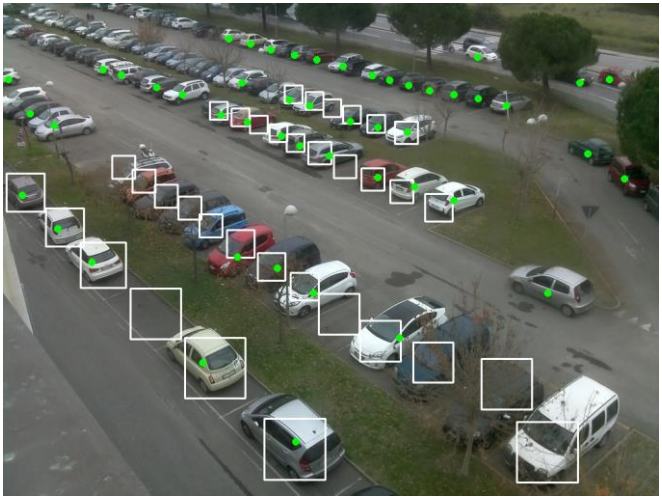
5.4 Hasil dan Evaluasi

Pada uji coba lokasi parkir pada dataset CNRPark diperoleh hasil akurasi yang paling baik pada lokasi parkir 9 dengan akurasi 86,90%, hal ini dikarenakan ROI yang disediakan dari dataset berada di tengah kendaraan dan berukuran cukup besar, ROI juga mencakup sebagian besar dari mobil yang tampak dari lokasi parkir tersebut, sedangkan hasil akurasi yang paling buruk adalah pada lokasi parkir 1 dengan akurasi 67,22%, hal ini disebabkan oleh ROI yang disediakan oleh dataset kebanyakan tidak berada tepat di tengah mobil dan ukuran ROI yang cukup kecil, disebabkan pula karena ada 2 mobil yang tertutup pohon sehingga tidak dapat dideteksi oleh model. Untuk hasil percobaan pada lokasi parkir 9

dapat dilihat pada Gambar 5.7, dan untuk hasil percobaan pada lokasi parkir 1 dapat dilihat pada Gambar 5.8



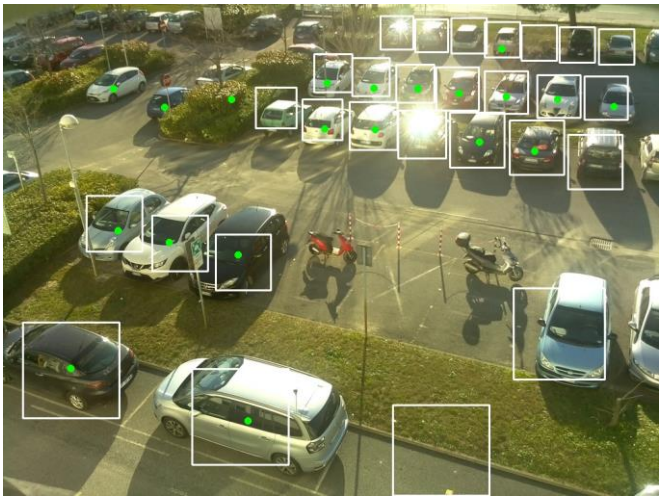
Gambar 5.7 Hasil percobaan pada lokasi parkir 9



Gambar 5.8 Hasil percobaan pada lokasi parkir 1

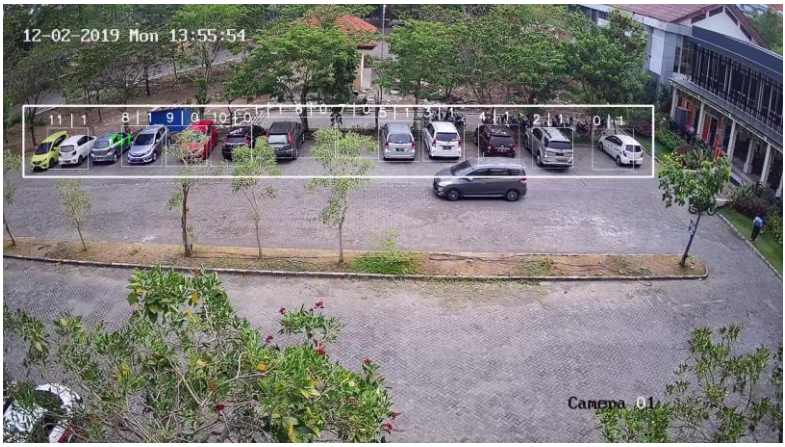
Pada uji coba perbedaan cuaca pada dataset CNRPark diperoleh hasil akurasi terbaik pada cuaca berawan dengan akurasi 86.99%, hal ini dikarenakan pada cuaca berawan dikarenakan bagian atas mobil tidak silau seperti pada cuaca cerah dan kamera tidak tertutup butiran air sehingga menghasilkan gambar yang sedikit tidak jelas seperti pada cuaca hujan, untuk akurasi terburuk ada pada cuaca cerah dengan akurasi 76,04%, hal ini dikarenakan bagian atas mobil silau dan membuat bagian mobil lainnya tertutup bayangan sehingga sulit untuk dideteksi, untuk hasil percobaan pada cuaca berawan dapat dilihat pada Gambar 5.9, dan untuk hasil percobaan pada cuaca cerah dapat dilihat pada Gambar 5.10

Gambar 5.9 Hasil percobaan pada cuaca berawan



Gambar 5.10 Hasil percobaan pada cuaca cerah

Selanjutnya dilakukan uji coba pada data video yang didapat dari CCTV yang dipasang pada parkiran Departemen Informatika ITS, pada kondisi siang hari dengan akurasi 85,48%, dengan presisi untuk tempat kosong sebesar 51,97% dan recall untuk tempat kosong sebesar 92,38%, presisi untuk tempat kosong pada kondisi siang hari kecil dikarenakan model tidak dapat mendeteksi mobil yang tertutup oleh pohon, dapat dilihat pada Gambar 5.11 terdapat 3 mobil yang tertutup pohon sehingga tidak dapat dideteksi oleh model, sedangkan presisi untuk tempat yang tidak kosong sebesar 98,35% dan recall sebesar 84,19%, dapat disimpulkan model dapat mendeteksi mobil dengan baik jika mobil tersebut tidak tertutup pohon. Untuk kondisi malam hari model mendapat akurasi sebesar 86,65% dengan presisi 99,88% dengan recall sebesar 49,84%, hal ini dikarenakan jumlah mobil yang hanya sedikit yaitu hanya 4 mobil dan 2 mobil diantaranya adalah mobil yang tertutup pohon sehingga tidak bisa dideteksi



Gambar 5.11 Hasil percobaan pada data video kondisi siang hari

Gambar 5.12 Hasil percobaan pada data video kondisi malam hari

Pada uji coba arsitektur ekstraksi fitur dapat dilihat bahwa akurasi dari Resnet101 mengungguli Resnet50 pada semua bidang, hal ini dikarenakan Resnet50 memiliki jumlah layer yang lebih sedikit sehingga kurang bisa untuk mengambil fitur dari gambar.

Pada uji coba arsitektur Faster-RCNN dapat dilihat pada perbandingan akurasinya Mask-RCNN jauh lebih baik daripada

Faster-RCNN hal ini dikarenakan Faster-RCNN menggunakan Resnet50 sebagai tahap ekstraksi fitur dan tidak menggunakan Feature Pyramid Network sehingga kurang dapat mendeteksi objek jika objek tersebut berukuran kecil.

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 terhadap lokasi parkir yang berbeda dengan akurasi tertinggi sebesar 86,90% pada lokasi parkir 9 dan akurasi terendah sebesar 67,22% pada lokasi parkir 1 dapat disimpulkan bahwa model yang dibangun dapat lebih baik dalam mendeteksi mobil jika mobil tersebut tidak tertutup oleh pohon atau tertutup oleh mobil lain, sehingga tempat peletakan kamera yang paling baik adalah menghadap bagian belakang atau bagian depan dari mobil.
2. Berdasarkan uji coba terhadap cuaca dengan akurasi tertinggi sebesar 86,99% pada cuaca berawan dan akurasi terendah sebesar 76,04% pada cuaca cerah dapat disimpulkan bahwa model dapat mendeteksi mobil dengan baik jika mobil dapat terlihat dengan jelas, dengan kata lain tidak terkena pantulan matahari dan kamera tidak tertutup oleh butiran hujan.
3. Berdasarkan uji coba terhadap data video lokasi parkir Departemen Informatika dengan akurasi sebesar 86,65% pada kondisi malam hari dan 85,48% pada kondisi siang hari dapat disimpulkan bahwa model yang dibangun dapat dengan baik mendeteksi mobil jika mobil tersebut tidak tertutup oleh objek lain.

4. Pada uji coba pembandingan arsitektur ekstraksi fitur Resnet101 dan Resnet50, Resnet101 dengan rata – rata akurasi pada data lokasi parkir CNRPark 81,14%, rata – rata akurasi pada data cuaca CNRPark 83,13% dan rata – rata pada data video lokasi parkir Departemen Informatika 86,07% jauh mengungguli hasil dari Resnet50.
5. Pada uji coba pembandingan metode Faster-RCNN dan Mask-RCNN, Mask-RCNN jauh mengungguli metode Faster-RCNN dengan akurasi pada data lokasi parkir CNRPark 81,14% dan rata – rata akurasi data cuaca CNRPark 83,13%.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem deteksi ketersediaan lahan parkir menggunakan *Mask Region-based Convolutional Neural Network*, yaitu:

1. Pengembangan sistem yang dapat mendeteksi mobil yang tertutup pohon
2. Melakukan eksplorasi untuk arsitektur ekstraksi fitur yang lebih baik
3. Melakukan eksplorasi parameter yang dapat menambah performa arsitektur seperti *activation function*, jumlah dan ukuran *filter* konvolusi, dan ukuran *max pooling layer*.

DAFTAR PUSTAKA

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

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

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

[4] 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.

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

[6] "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].

[7] "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].

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

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

[10 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].

[11 "TensorFlow," TensorFlow, [Online]. Available: https://www.tensorflow.org/. [Diakses 20 Desember 2019].

[12 J. Hui, "Medium," 27 Maret 2018. [Online]. Available: https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c. [Diakses 20 Desember 2019].

[13 S. K, "towardsdatascience," 1 Oktober 2019. [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c. [Diakses 24 Desember 2019].

[14 T. Karmarkar, "medium," 19 Agustus 2018. [Online]. Available: https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9. [Diakses 24 Desember 2019].

[15 "Wikipedia," 26 Mei 2019. [Online]. Available: https://en.wikipedia.org/wiki/Region_of_interest. [Diakses 20 Desember 2019].

[16 K. He, G. Gkioxari, P. Dollár dan R. Girshick, “Mask R-CNN,” vol. 3, 24 Januari 2017.

[17 “mc.ai,” 1 April 2018. [Online]. Available: https://mc.ai/instance-embedding-instance-segmentation-without-proposals/. [Diakses 24 Desember 2019].

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

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

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

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

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

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

(Halaman ini sengaja dikosongkan)

LAMPIRAN

L.1 Confusion Matrix Hasil Uji Coba Lokasi Parkir 1

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	4.233	76
	Kosong	5.075	6.331

L.2 Confusion Matrix Hasil Uji Coba Lokasi Parkir 2

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	1.795	20
	Kosong	846	1.434

L.3 Confusion Matrix Hasil Uji Coba Lokasi Parkir 3

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	3.109	13
	Kosong	2.261	4.088

L.4 Confusion Matrix Hasil Uji Coba Lokasi Parkir 4

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	6.169	10
	Kosong	3.188	7.209

L.5 Confusion Matrix Hasil Uji Coba Lokasi Parkir 5

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	8.188	14
	Kosong	3.068	9.568

L.6 Confusion Matrix Hasil Uji Coba Lokasi Parkir 6

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	7.709	9
	Kosong	2.937	9.453

L.7 Confusion Matrix Hasil Uji Coba Lokasi Parkir 7

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	7.484	16
	Kosong	3.035	10.579

L.8 Confusion Matrix Hasil Uji Coba Lokasi Parkir 8

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	9.136	37
	Kosong	3.711	11.200

L.9 Confusion Matrix Hasil Uji Coba Lokasi Parkir 9

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	5.678	13
	Kosong	1.685	5.588

L.10 Confusion Matrix Hasil Uji Coba Cuaca Cerah

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	22.494	121
	Kosong	15.019	25.544

L.11 Confusion Matrix Hasil Uji Coba Cuaca Berawan

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	17.475	55
	Kosong	5.701	21.012

L.12 Confusion Matrix Hasil Uji Coba Cuaca Hujan

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	13.532	32
	Kosong	5.086	18.894

L.13 Confusion Matrix Hasil Uji Coba Data Video Lokasi Parkir Departemen Informatika Kondisi Siang

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	11.350	190
	Kosong	2.130	2.305

L.14 Confusion Matrix Hasil Uji Coba Data Video Lokasi Parkir Departemen Informatika Kondisi Malam

		Label asli	
		Ditempati	**Kosong**
Label deteksi	**Ditempati**	8.395	1
	Kosong	8.448	46.516

L.15 Hasil Uji Coba Berdasarkan Lokasi Parkir

Lahan Parkir	Akurasi	Kelas	Data Uji	Presisi	Recall
1	67,22%	Kosong	6407	55,50%	98,81%
		Ditempati	9308	98,23%	45,47%
2	78,85%	Kosong	1454	62,89%	98,62%
		Ditempati	2641	98,89%	67,96%
3	75,99%	Kosong	4101	64,38%	99,68%
		Ditempati	5370	99,58%	57,89%
4	80,71%	Kosong	7219	69,33%	**99,86%**
		Ditempati	9357	99,83%	65,92%
5	85,21%	Kosong	9582	75,72%	99,85%
		Ditempati	11256	99,82%	72,74%
6	85,35%	Kosong	9462	76,29%	99,90%
		Ditempati	10646	**99,88%**	72,41%
7	85,55%	Kosong	10595	**77,71%**	99,84%
		Ditempati	10519	99,78%	71,14%
8	84,44%	Kosong	11237	75,11%	99,67%
		Ditempati	12847	99,59%	71,11%
9	**86,90%**	Kosong	5601	76,83%	99,76%
		Ditempati	7363	99,77%	**77,11%**

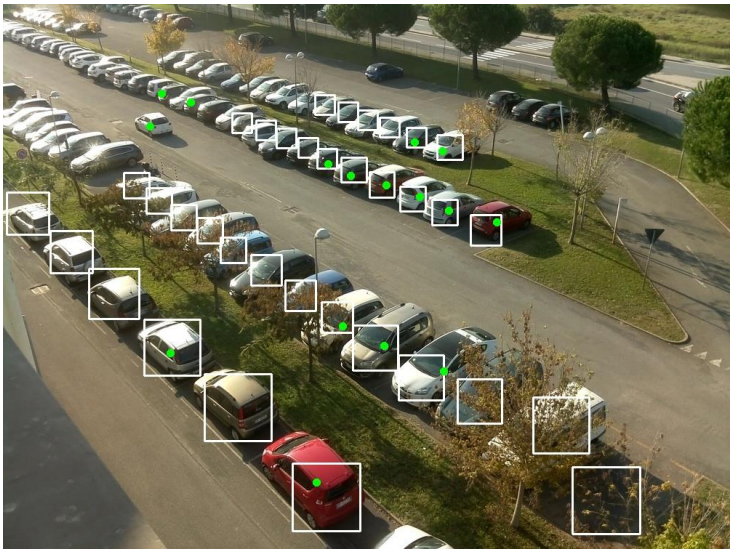
L.16 Hasil Uji Coba Berdasarkan Kondisi Cuaca

Cuaca	Akurasi	Kelas	Data Uji	Presisi	Recall
Cerah	76,04%	Kosong	25665	62,97%	99,52%
		Ditempati	37513	99,46%	59,96%
Hujan	86,37%	Kosong	18926	**78,79%**	**99,83%**
		Ditempati	18618	**99,76%**	72,68%
Berawan	**86,99%**	Kosong	21067	78,65%	99,73%
		Ditempati	23176	99,68%	**75,40%**

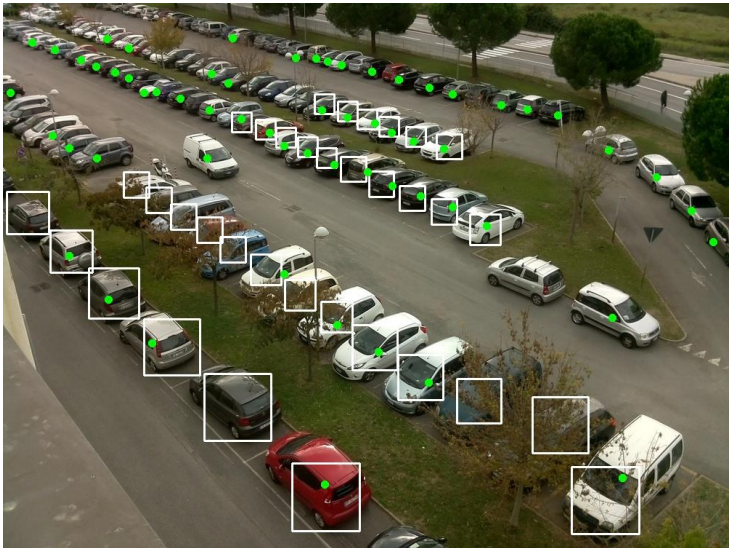
L.17 Hasil Uji Coba Data Video Lokasi Parkir Departemen Informatika

Kondisi	Akurasi	Kelas	Data Uji	Presisi	Recall
Siang	85,48%	Kosong	2495	51,97%	92,38%
		Ditempati	13480	98,35%	**84,19%**
Malam	**86,65%**	Kosong	46517	**84,63%**	**99,99%**
		Ditempati	16843	**99.88%**	49.84%

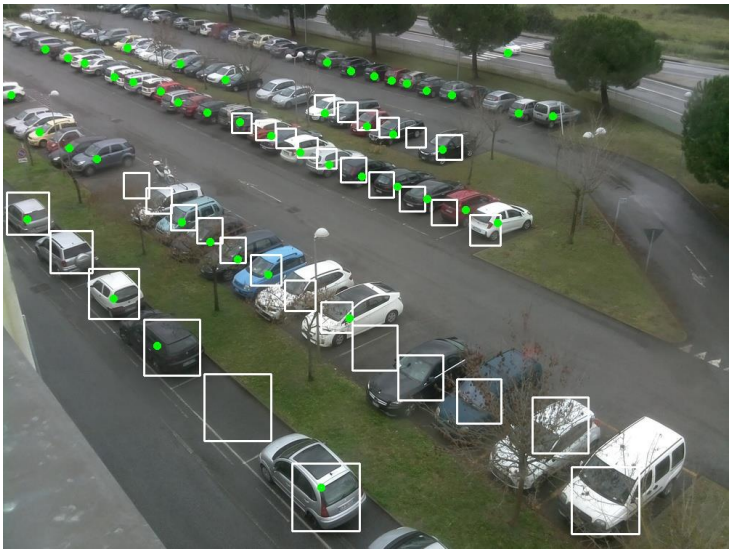
L.18 Hasil Uji Coba Lokasi Parkir 1 Pada Cuaca Cerah



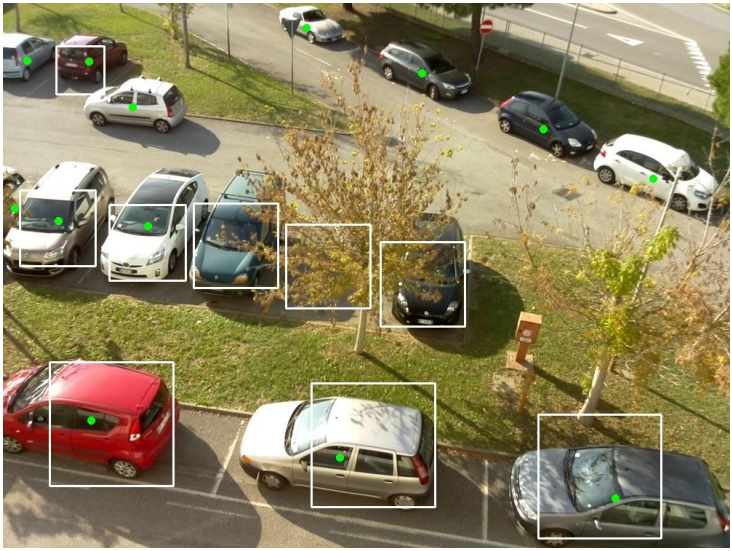
L.19 Hasil Uji Coba Lokasi Parkir 1 Pada Cuaca Berawan



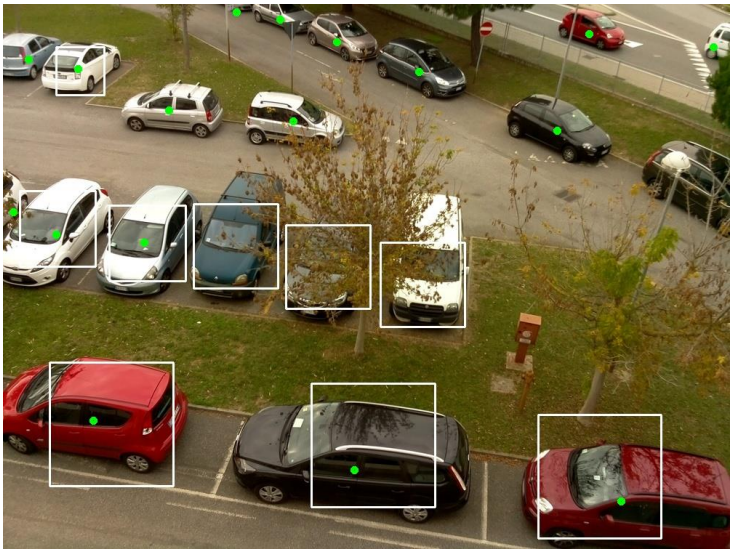
L.20 Hasil Uji Coba Lokasi Parkir 1 Pada Cuaca Hujan



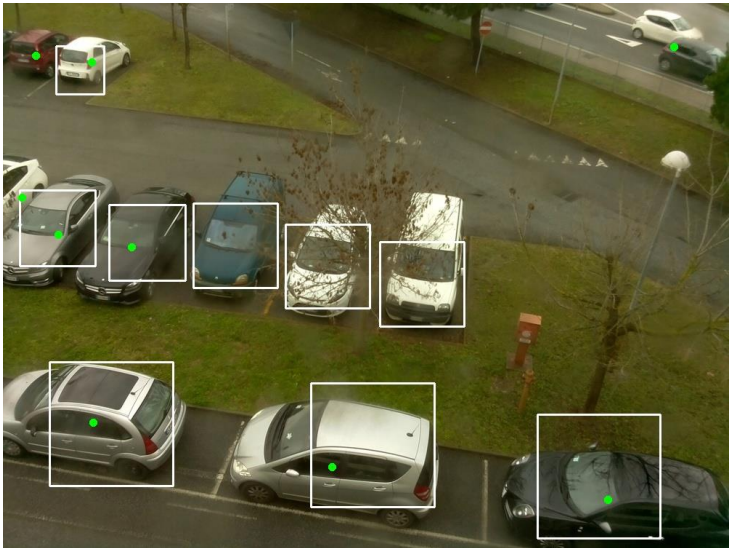
L.21 Hasil Uji Coba Lokasi Parkir 2 Pada Cuaca Cerah



L.22 Hasil Uji Coba Lokasi Parkir 2 Pada Cuaca Berawan

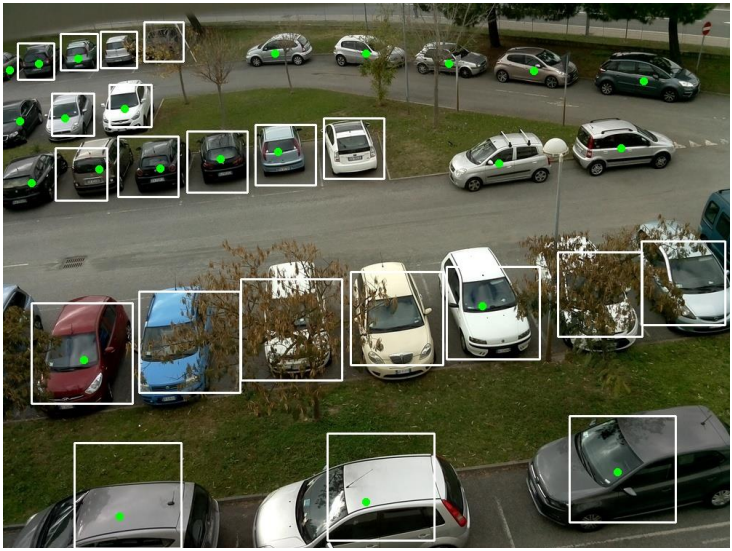


L.23 Hasil Uji Coba Lokasi Parkir 2 Pada Cuaca Hujan

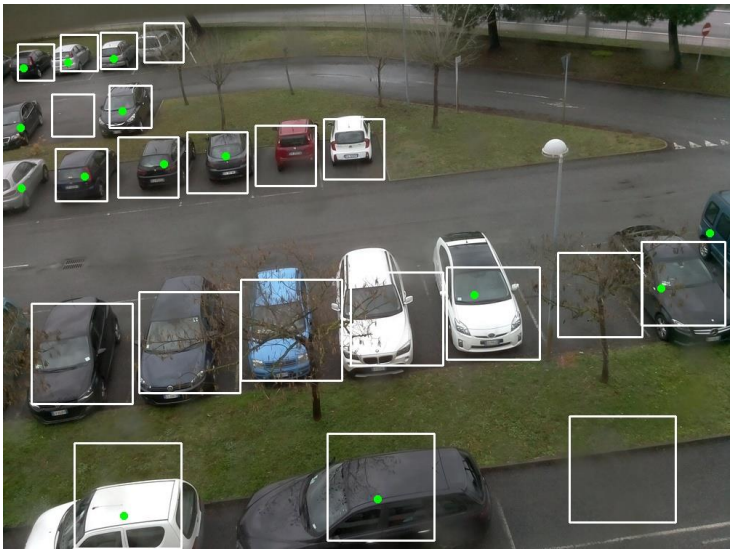


L.24 Hasil Uji Coba Lokasi Parkir 3 Pada Cuaca Cerah

L.25 Hasil Uji Coba Lokasi Parkir 3 Pada Cuaca Berawan



L.26 Hasil Uji Coba Lokasi Parkir 3 Pada Cuaca Hujan



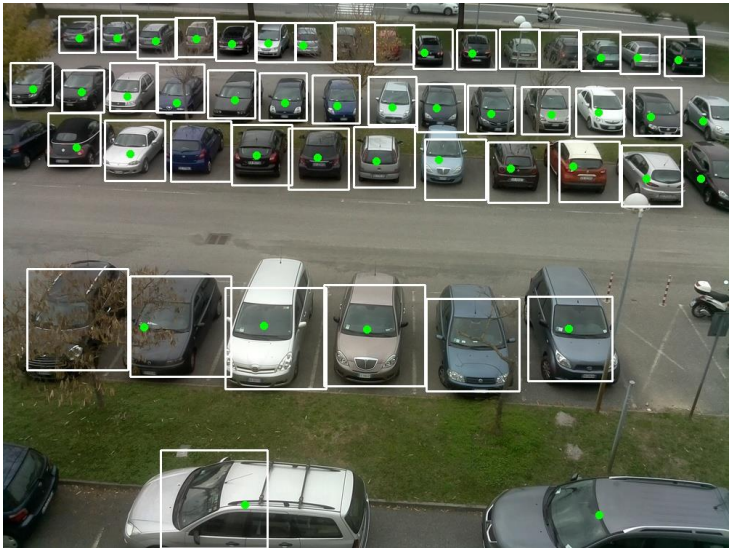
L.27 Hasil Uji Coba Lokasi Parkir 4 Pada Cuaca Cerah

L.28 Hasil Uji Coba Lokasi Parkir 4 Pada Cuaca Berawan

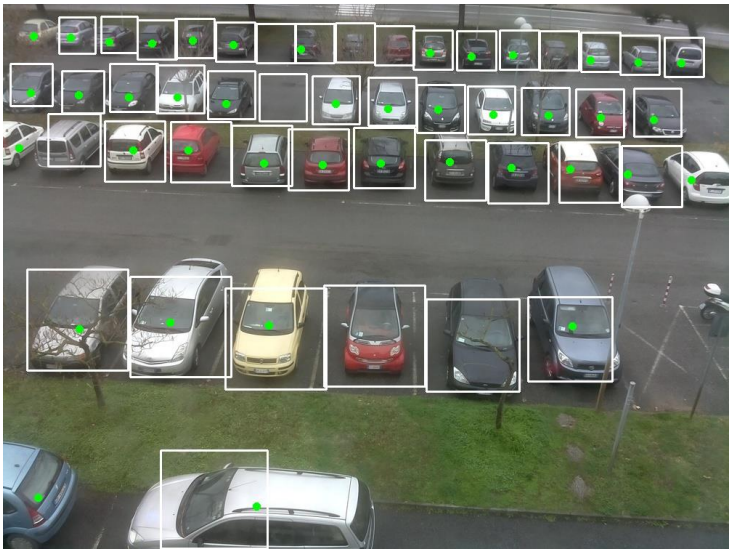
L.29 Hasil Uji Coba Lokasi Parkir 4 Pada Cuaca Hujan

L.30 Hasil Uji Coba Lokasi Parkir 5 Pada Cuaca Cerah

L.31 Hasil Uji Coba Lokasi Parkir 5 Pada Cuaca Berawan

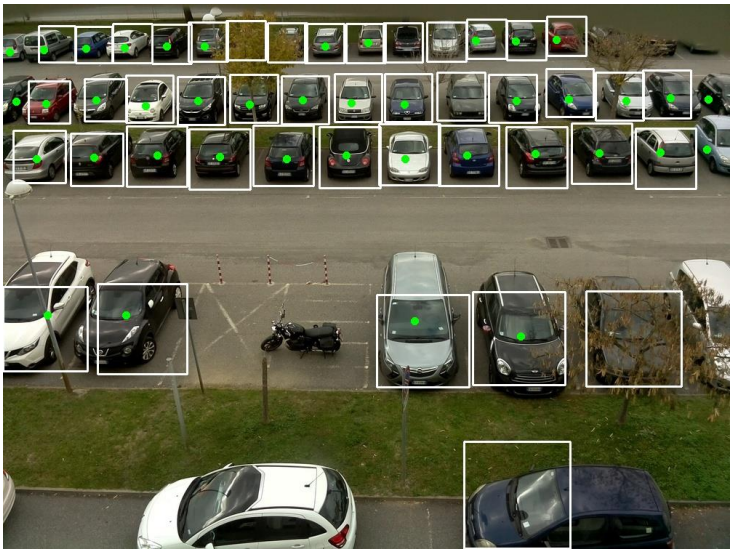


L.32 Hasil Uji Coba Lokasi Parkir 5 Pada Cuaca Hujan

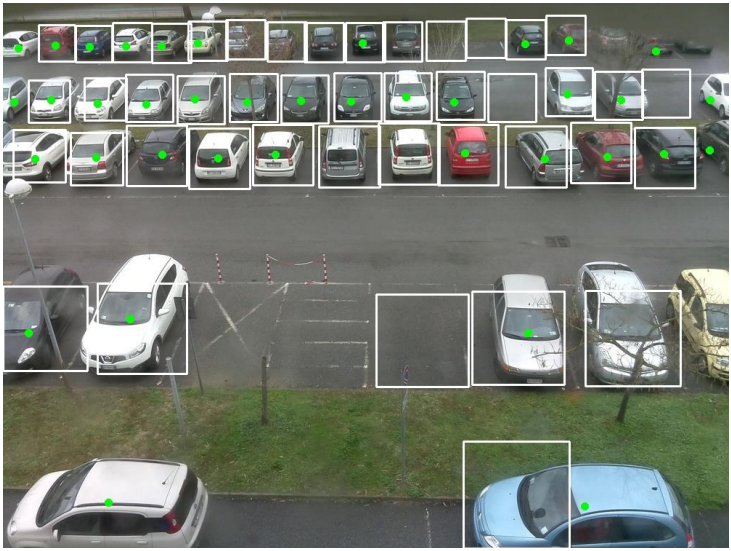


L.33 Hasil Uji Coba Lokasi Parkir 6 Pada Cuaca Cerah

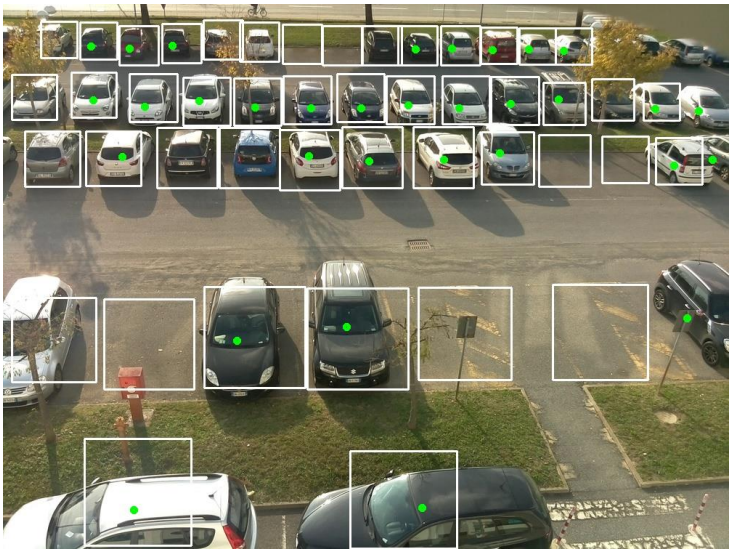
L.34 Hasil Uji Coba Lokasi Parkir 6 Pada Cuaca Berawan



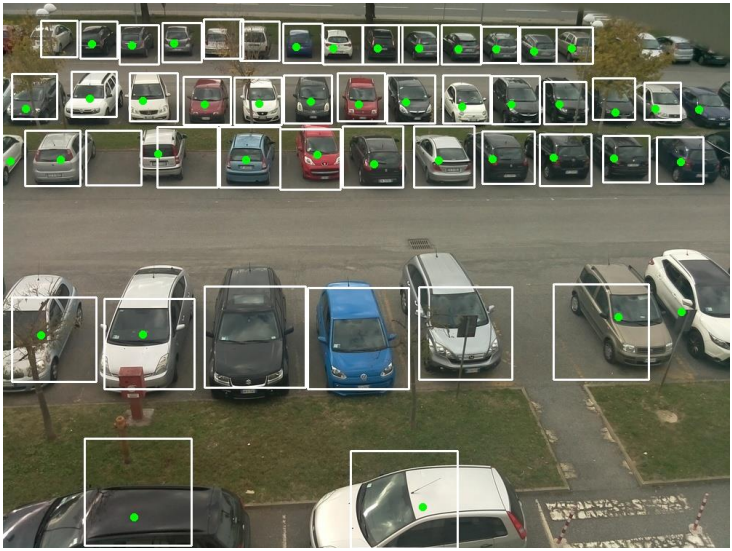
L.35 Hasil Uji Coba Lokasi Parkir 6 Pada Cuaca Hujan



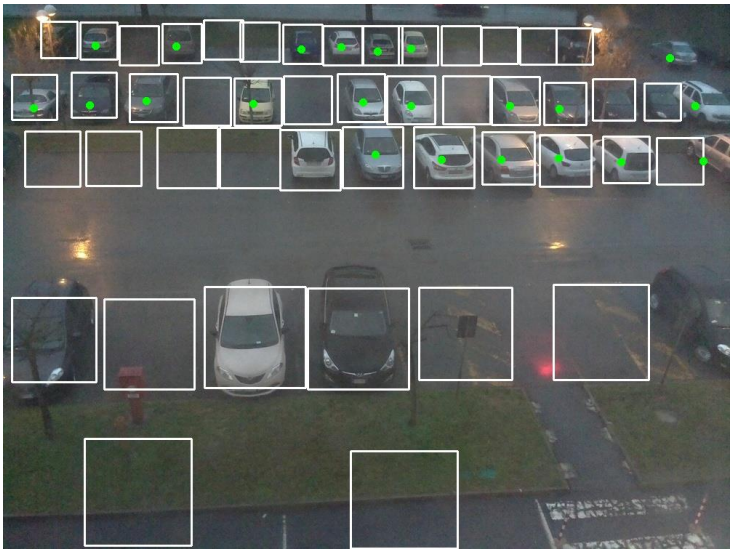
L.36 Hasil Uji Coba Lokasi Parkir 7 Pada Cuaca Cerah



L.37 Hasil Uji Coba Lokasi Parkir 7 Pada Cuaca Berawan

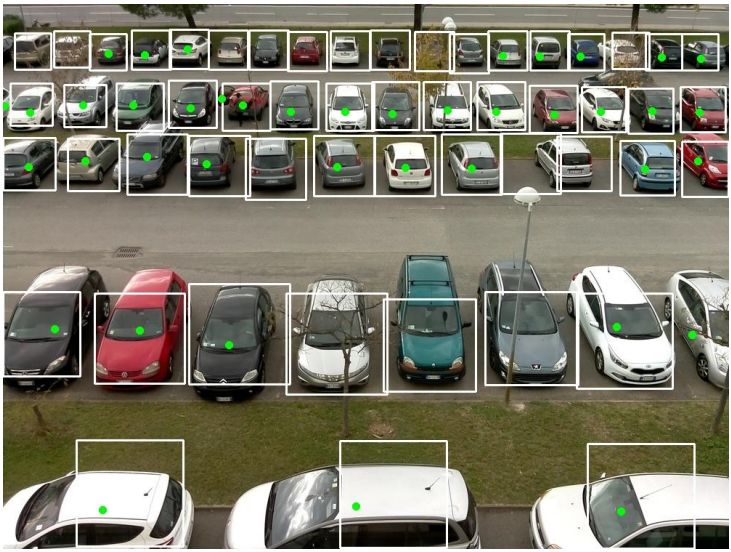


L.38 Hasil Uji Coba Lokasi Parkir 7 Pada Cuaca Hujan

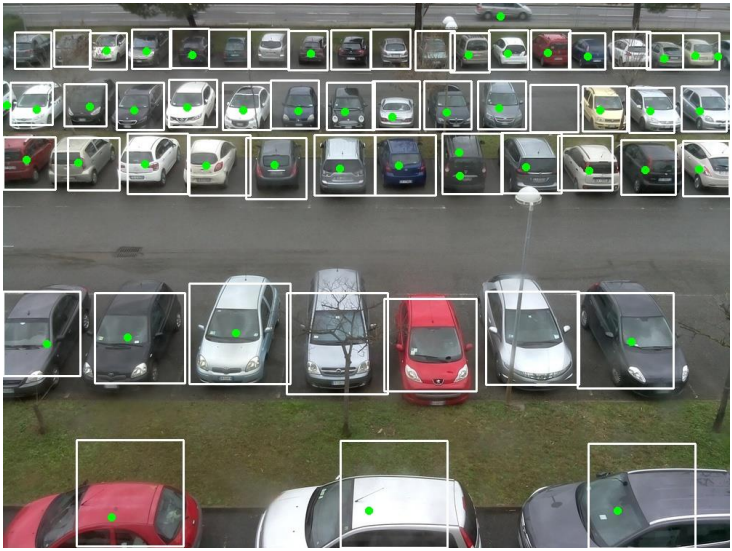


L.39 Hasil Uji Coba Lokasi Parkir 8 Pada Cuaca Cerah

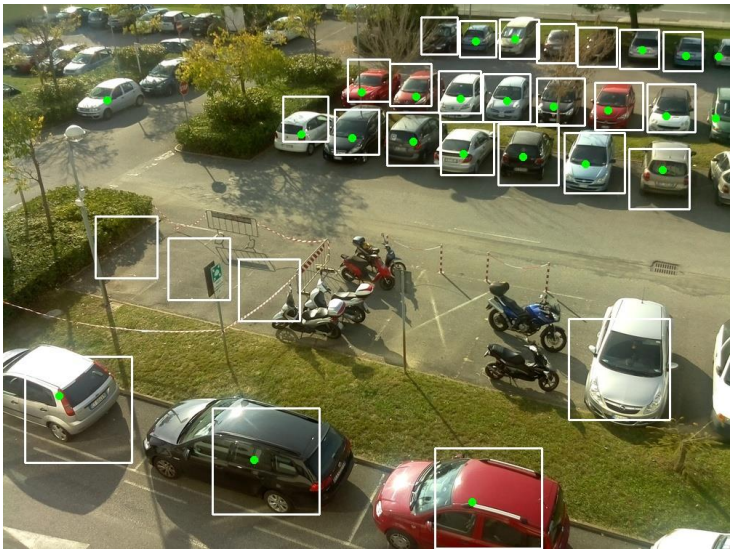
L.40 Hasil Uji Coba Lokasi Parkir 8 Pada Cuaca Berawan



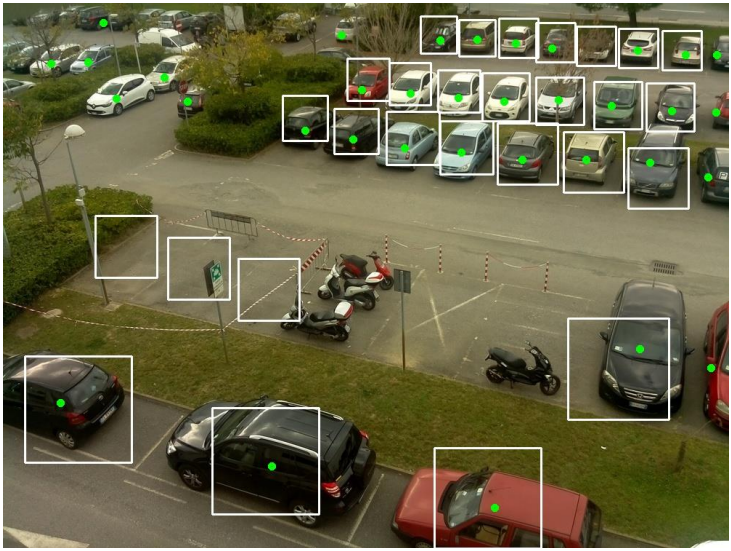
L.41 Hasil Uji Coba Lokasi Parkir 8 Pada Cuaca Hujan



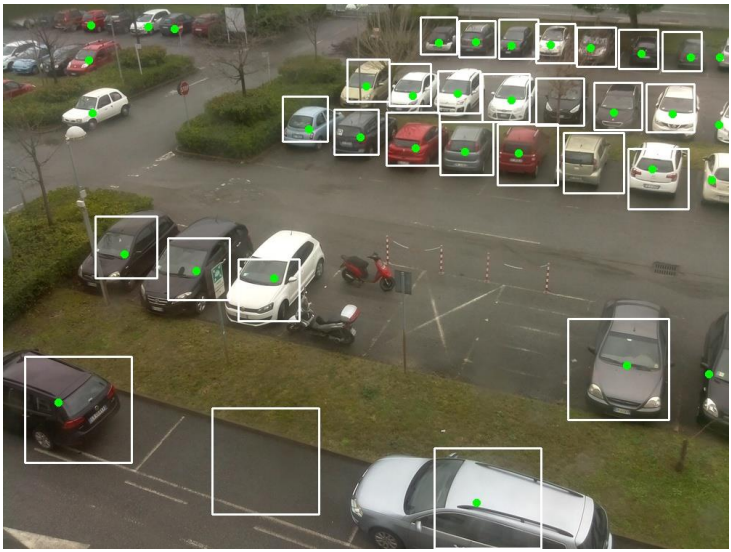
L.42 Hasil Uji Coba Lokasi Parkir 9 Pada Cuaca Cerah



L.43 Hasil Uji Coba Lokasi Parkir 9 Pada Cuaca Berawan



L.44 Hasil Uji Coba Lokasi Parkir 9 Pada Cuaca Hujan



BIODATA PENULIS

Rahandi Noor Pasha, lahir di Banyuwangi pada tanggal 21 Januari 1998. Penulis menempuh pendidikan mulai dari SDN 1 Genteng Banyuwangi (2004-2010), SMPN 1 Genteng Banyuwangi (2010-2013), SMAN 1 Genteng Banyuwangi (2013-2016), dan sekarang sedang menjalani pendidikan S1 Informatika di ITS. Penulis aktif dalam organisasi dan kepanitiaan Schematics ITS. Diantaranya adalah menjadi staff Departemen Website dan Kesekretariatan Schematics ITS, Wakil Ketua Departemen Website dan Kesekretariatan Schematics ITS 2018. Penulis juga merupakan salah satu pengurus lab Komputasi Cerdas dan Visi di departemen Informatika ITS. Komunikasi dengan penulis dapat melalui telepon: +6281217645673 dan *email*: **rahandinoor@gmail.com**.