DETEKSI SEPEDA MOTOR DI JALAN RAYA MENGGUNAKAN *FASTER R-CNN* BERBASIS *VGG16*

SKRIPSI

Diajukan untuk Memenuhi bagian dari Syarat Memperoleh Gelar Sarjana Komputer Pada Departemen Pendidikan Ilmu Komputer Program Studi Ilmu Komputer

oleh:

Mochamad Dian Lazuardi Yudha

1403206

PROGRAM STUDI ILMU KOMPUTER

DEPARTEMEN PENDIDIKAN ILMU KOMPUTER

FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PENDIDIKAN INDONESIA BANDUNG

2020

DETEKSI SEPEDA MOTOR DI JALAN RAYA MENGGUNAKAN *FASTER R-CNN* BERBASIS *VGG16*

Oleh
Moch Dian Lazuardi Yudha

Sebuah skripsi yang diajukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana pada Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam

Universitas Pendidikan Indonesia
Agustus 2020

MOCH DIAN LAZUARDI YUDHA

DETEKSI SEPEDA MOTOR DI JALAN RAYA MENGGUNAKAN *FASTER R-CNN* BERBASIS *VGG16*

disetujui dan disahkan oleh pembimbing:

Pembimbing I

Prof. Dr. H. Wawan Setiawan, M. Kom.

NIP. 196601011991031005

Pembimbing II

Yaya Wihardi, S. Kom., M. Kom.

NIP. 198903252015041001

Mengetahui,

Ketua Departemen Pendidikan Ilmu Komputer

Dr. Lala Septem Riza, M.T.

NIP. 197809262008121001

PERNYATAAN

Saya menyatakan bahwa skripsi yang berjudul "Deteksi Sepeda Motor di Jalan Raya Menggunakan *Faster R-CNN* Berbasis VGG16" ini sepenuhnya karya sendiri. Tidak ada plagiat dari orang lain di dalamnya dan saya tidak melakukan penyalinan atau pengutipan dengan caracara yang tidak sesuai etika keilmuwan yang berlaku dalam masyarakat keilmuwan. Atas pernyataan ini, saya siap menanggung sanksi yang dijatuhkan kepada saya apabila ditemukan adanya pelanggaran terhadap etika keilmuwan di karya ini atau ada klaim dari pihak lain terhadap keaslian karya saya ini.

Bandung, Agustus 2020

Pembuat pernyataan,

Moch Dian Lazuardi Yudha

NIM 1403206

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena berkat rahmat dan hidayah-Nya, penelitian yang berjudul "**Deteksi Sepeda Motor di Jalan Raya Menggunakan *Faster R-CNN* Berbasis *VGG16***" dapat diselesaikan. Skripsi ini disusun untuk memenuhi salah satu syarat dalam menempuh gelar Sarjana Komputer di Program Studi Ilmu Komputer, Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam, Universitas Pendidikan Indonesia. Penulis sadari bahwa skripsi yang disusun dengan segala usaha dari awal sampai selesai ini masih jauh dari kata sempurna. Oleh karena itu, penulis mengharapkan berbagai kritik dan saran dari para pecinta ilmu pengetahuan yang bersifat positif supaya skripsi ini dapat dibangun dengan lebih baik. Penulis juga berharap skripsi ini dapat memberikan manfaat, baik untuk penulis sendiri, umumnya bagi para pengembang teknologi dan pecinta ilmu pengetahuan.

Bandung, Agustus 2020

Penulis

Moch Dian Lazuardi Yudha

UCAPAN TERIMA KASIH

Skripsi ini dapat diselesaikan tidak lepas dari dukungan, dorongan, serta bantuan dari berbagai pihak. Sehingga pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada beberapa pihak yang telah berperan serta dalam penyelesaian laporan ini, diantaraya adalah:

1. Orang tua yang selalu mendo'akan penulis sehingga penulisan skripsi dapat terselesaikan.
2. Dr. Lala Septem Riza, M.T. selaku Ketua Departemn Pendidikan Ilmu Komputer Universitas Pendidikan Indonesia.
3. Dr. Rani Megasari, M.T. selaku Ketua Program Studi Ilmu Komputer Universitas Pendidikan Indonesia.
4. Bapak Prof. Dr. H. Wawan Setiawan, M.Kom selaku dosen pembimbing I dan pembimbing akademik yang telah membimbing dalam penyelesaian skripsi dan penyusunan skripsi.
5. Bapak Yaya Wihardi, M.Kom selaku dosen pembimbing II yang telah membimbing dalam penyelesaian skripsi dan penyusunan skripsi.
6. Wina Yulinar selaku orang yang selalu memberikan dorongan untuk menyelesaikan skripsi dan penyusunan skripsi.
7. Sahabat dan teman-teman serta semua pihak yang tidak dapat penulis sebutkan satu-persatu yang telah memberikan dukungan selama proses pengerjaan skripsi.

Penulis menyadari bahwa penyusunan skripsi ini masih terdapat banyak kekurangan dan keterbatasan yang perlu disempurnakan, oleh karena itu penulis sangat mengharapkan saran maupun kritik yang membangun agar tidak terjadi kesalahan yang sama di kemudian hari dan dapat meningkatkan kualitas ke tahap yang lebih baik. Akhir kata, semoga skripsi ini dapat bermanfaat dan memberikan wawasan yang luas kepada para pembaca.

Bandung, Agustus 2020

Penulis

Abstrak

Menurut Badan Pusat Statistika (BPS) pada tahun 2016, pengendara motor di Indonesia mencapai 98.9 Juta jiwa atau 81.5% dari keseluruhan pengendara di Indonesia. Dan hanya sedikit orang yang menggunakan Mobil Penumpang dibandingkan dengan sepeda motor. Ini tandanya, Indonesia wajib menguji pengendara sepeda motor agar bisa layak mengedara di jalanan. Oleh karena itu harus dibuatkan sebuah system untuk mendeteksi sepeda motor. Banyak algoritma yang telah di *publish* dengan berbagai arsitektur, penelitian ini menggunakan arsitektur yang sedang popular yaitu *faster rcnn*. *Faster rcnn* telah diuji dibanyak penelitian untuk mendeteksi objek dalam gambar dan video. Penelitian ini memiliki beberapa tahapan yaitu pelabelan objek pada data training, pra proses, *training*, dan *testing*. *Faster RCNN* dalam penelitian ini menggunakan dasar model algoritma vgg16 yang memiliki total 16 layer. Hasil dari percobaan dengan menggunakan gambar dari CCTV menunjukan nilai *loss* 6.01% dan 12.11%. Hasil tersebut menunjukan bahwa sistem sudah cukup baik dalam mendeteksi sepeda motor di jalan raya.

Kata Kunci : Deteksi Sepeda Motor, *Faster R-CNN, VGG16, Deep Learning, Convolutional Neural Network*

Abstract

According to *Badan Pusat Statistika (BPS)* on 2016, motorcycle riders in Indonesia reached 98.9 million people or 81.5% from all riders in Indonesian. Only a few people using pubic transportation. This is why Indonesian need to make rule for people who using private vehicle. Therefore, a system must be made do detect motorcycle. Many algorithms have been published with various architectures. This research uses the currently popular architecture, namely the faster rcnn. RCNN Faster has been tested in many studies to detect objects in images and videos. This research has several stages, namely object labeling on training data, pre-processing, training, and testing. The RCNN Faster in this study uses the basic VGG16 algorithm model which has a total of 16 layers. The results of the experiment using images from CCTV showed a loss value of 6.01% and 12.11%. These results indicate that the system is quite good at detecting motorbikes on the highway.

Key Words : Motorcycle Detection, *Faster R-CNN, VGG16, Deep Learning, Convolutional Neural Network*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar belakang

Menurut Badan Pusat Statistika (BPS) pada tahun 2016, pengendara motor di Indonesia mencapai 98.9 *Juta* jiwa atau 81.5% dari keseluruhan pengendara di Indonesia. Dan hanya sedikit orang yang menggunakan angkutan umum dibandingkan dengan sepeda motor. Ini tandanya, Indonesia wajib menguji pengendara sepeda motor agar bisa layak mengedara di jalanan. Selain itu, banyak kecelakaan yang dialami oleh pengendara motor yang disebabkan kesalahan pengendaranya sendiri. Maka dari itu lebih baik dibuat sistem untuk mendeteksi orang yang melanggar peraturan lalu lintas di jalan raya. Sebelum mendeteksi pelanggaran, tahap pertama yang harus dilakukan yaitu mendeteksi sepeda motor.

Penulis menggunakan *Convolutional Neural Network* (*CNN*) dengan arsitektur *Faster RCNN* yang telah dibundle dengan *library Tensorflow* menggunakan Bahasa pemrograman *Python*. Karena *CNN* sangat bagus performa nya pada kompilasi dengan image (Bianco, 2016). Pada waktu kompilasi, *image* harus memulai pada tahap pra-proses yang dimana image akan melalui tahap ekstraksi, diubah ke bentuk *gray-scale*, dll. Tahap pra proses tersebut akan sangat berguna pada bagian pembuatan model dengan menggunakan *CNN*. Karena dapat meningkatkan tingkat *accuracy* (Jiang, 2016). Arsitektur *Faster RCNN* telah dibuktikan akurasinya pada penelitian buah kiwi dengan keadaan pagi, siang, dan malam hari dan mendapatkan hasil yang bagus, selain itu buah kiwi dapat terlihat jelas pada waktu testing (Song, 2019)

Teknik *pattern recognition* yaitu teknik pengenalan pola. Pola yang dikenali bisa bermacam-macam, seperti wajah, bola mata, sidik jari, dll. Tetapi dalam kasus ini, pola orang yang memakai helm lah yang akan diproses. Hal ini dibuktikan sudah banyak peneliti yang meneliti *pattern recognition* tersebut (Han,Chin-chuan, dkk, 2001). Penggunaan *machine learning* pada kasus ini akan memudahkan pihak yang berwenang pada saat di lampu merah. Karena teknik ini, akan otomatis melihat siapa orang yang melanggar atau orang yang tidak memakai helm di lampu merah.

Dalam pembuatan sistem, citra akan diuji dalam hal *contrast* citra,

misalnya bagaimana jika citra / gambar tersebut dipengaruhi dengan cahaya yang sangat terang atau bahkan tidak terdapat cahaya sama sekali pada malam hari. Tahapan tersebut bisa dilakukan di *pre-processing* dan ekstraksi pada citra. Mengapa menggunakan beberapa test karena bisa saja citra tersebut tidak berjalan ditahap *learning* jika contrastnya sangat terang ataupun sangat rendah nantinya (Han, 2016) *Recognition* sendiri terdiri dari image dan video yang dimana akan dirangkum pada topic *computer vision*. Pada proses *learning*, tentu saja dengan menggunakan video proses akan berjalan dengan sangat lambat dikarenakan *training data* yang dipakai berkali-kali masuk kedalam *training data*. Berbeda dengan *image* proses yang dilakukan hanyalah ruang lingkup dari *image* itu sendiri (Qi, 2016) oleh karena itu penulis akan menggunakan *Image* pada *testing* tugas akhir ini.

1.2 Rumusan masalah penelitian

Rumusan masalah pada penilitian ini adalah:

1. Bagaimana mendeteksi sepeda motor di jalan raya?
2. Bagaimana kemampuan algoritma arsitektur *Faster R-CNN* berbasis *VGG16* dalam mengklasifikasi sepeda motor di persimpangan jalan?

1.3 Tujuan penelitian

Tujuan yang ingin dicapai pada penilitian ini adalah:

1. Implementasi algoritma arsitektur *Faster R-CNN Base VGG16* pada deteksi pengendara yang tidak memakai helm
2. Analisis kemampuan algoritma arsitektur *Faster R-CNN Base VGG16* pada deteksi pengendara yang tidak memakai helm

1.4 Manfaat penelitian

1. Dapat mempelajari bagaimana arsitektur *Faster R-CNN (VGG16)* bekerja.

1.5 Batasan masalah penilitian

Berikut beberapa batasan masalah dari penelitian ini:

1. Dalam pengujian, data sample diambil dari website dishub Kab. Sukoharjo.
2. Sistem tidak akan mendeteksi jika citra yang diberikan berupa gambar

bergerak atau video.

3. Perangkat lunak yang akan dibangun untuk mendukung penelitian ini menggunakan pemrograman *Python*.

1.6 Sistematika penulisan

Sistematika penulisan yang akan disampaikan pada penelitian ini, yaitu:

BAB I PENDAHULUAN

Bab 1 merupakan pendahuluan dari skripsi ini yang terdiri dari sub bab latar belakang, rumusan masalah, manfaat, batasan masalah, dan sistematika penulisan.

BAB II KAJIAN PUSTAKA

Pada kajian Pustaka akan diuraikan materi-materi yang berhubungan dengan penelitian. Materi ini yang mendasari penulis dalam melakukan penelitian. Materi yang disampaikan meliputi citra, pengerian citra, *color*, metode sistem, pengeriaan dan sejarah *machine learning*, *supervised learning*, *unsupervised learning*, *convolutional neural network (CNN)*, python, *faster r-cnn berbasis vgg16.*

BAB III

Bab ini berisi tahapan penilitian yang digambarkan pada desain penelitian, alat dan bahan yang digunakan dalam penilitian. Pada bab ini juga menjelaskan mengenai data penelitian baik itu *input* maupun *output.*

BAB IV PEMBAHASAN

Bab pembahasan menjelaskan tentang bagaimana penelitian dilakukan sesuai tahapan yang telah dibuat pada desain penelitian. Setiap alur yang ada pada desain penelitian dijelaskan secara berurutan. Pada bab ini juga dijelaskan scenario pengujian yang dilakukan beserta hasil pengujian.

BAB V KESIMPULAN

Bab kesimpulan berisi tentang rangkuman dari hasil penelitian yang telah dilakukan. Selain kesimpulan, pada bab 5 disampaikan saran untuk penelitian selanjutnya.

LAMPIRAN

BAB II
KAJIAN PUSTAKA

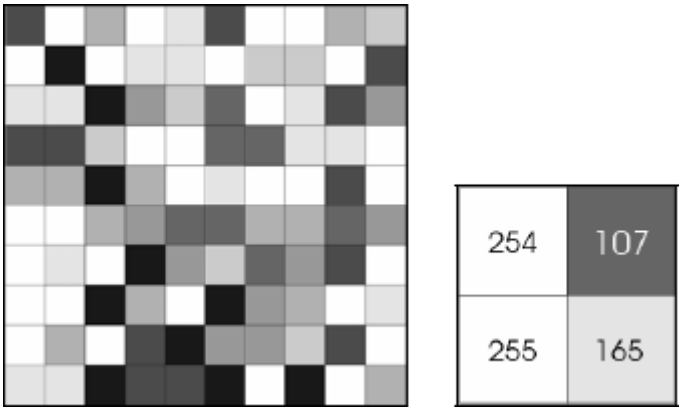
2.1 Penelitian Terkait

Beberapa hasil penelitian sebelumnya yang relevan dengan penelitian ini adalah:

1. Song, Zhenzhen. (Song, 2019) mengusulkan metode algoritma *faster rcnn* berbasis *vgg16* untuk mendeteksi buah kiwi dalam keadaan pagi, siang, dan malam. Pada penelitian ini terdapat error yaitu buah kiwi yang tidak terdeteksi. Pada penelitian ini, *Faster RCNN* berbasis *VGG16* mendapatkan *average-precision* sebesar 87.61%. Angka tersebut sudah menunjukan bahwa algoritma ini dapat dipakai untuk mendeteksi suatu objek pada citra.
2. Chahyati, Dina. (Chahyati, 2017) mengusulkan metode algoritma *faster rcnn* untuk *tracking* manusia menggunakan *cctv*.

2.2 Citra

Citra adalah sebuah array atau matrix yang memiliki kolom dan baris. Jika dilihat pada 8-bit citra *grayscale*, element di assign pada intensitas 0 sampai 255. Orang normal memanggil gray scale tersebut dengan sebutan hitam- putih. Citra *grayscale* normal memiliki 8-bit kedalaman warna atau sama dengan 256 *grayscale*. Dan untuk citra yang memiliki warna, mempunyai 24-bit kedalaman warna dengan 8x8x8 bits atau 256x256x256 *colors* atau setara dengan 16 juta *colors* (Young, 2007).



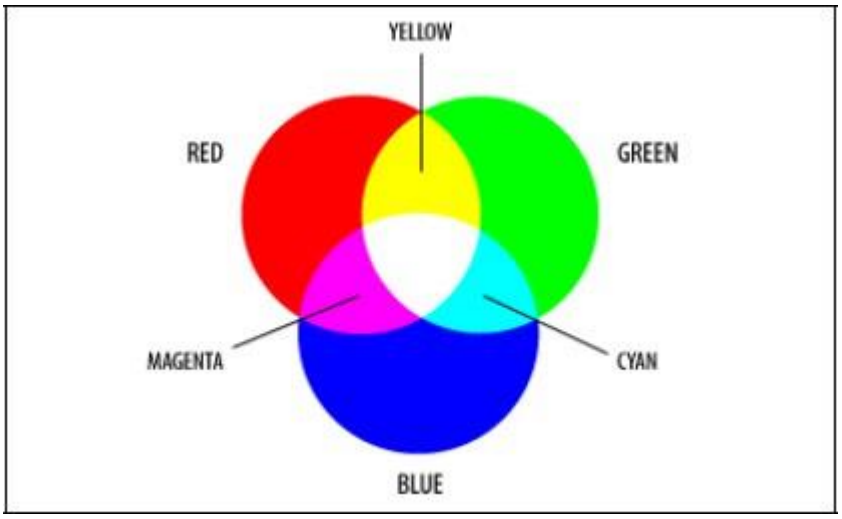
Gambar 2.1 *Array of Image* (Young, 2007)

Terdapat 2 *group* pada citra yaitu graphic vector dan bitmaps dengan beberapa format seperti GIF, JPEG, TIFF, dll.

2.3 Color

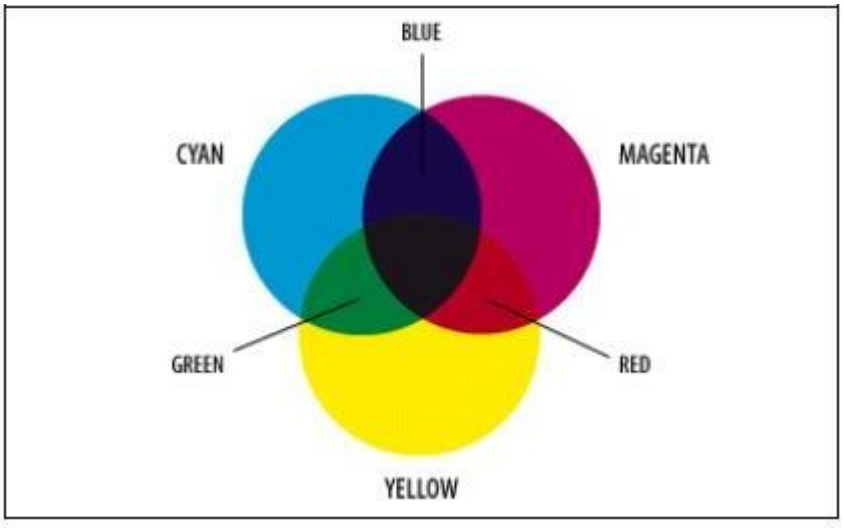
Pada setiap gambar memiliki warna yang bermacam-macam. Tetapi warna inti hanyalah *Red*, *Green*, dan *Blue*. Didalam ilmu sains, terdapat 2 bentuk yaitu *RGB* dan *CMYK* (Young, 2007).

RGB terdiri dari dari warna *Red* (Merah), *Green* (Hijau), dan *Blue* (B). *RGB* juga telah digunakan di perangkat-perangkat yang telah ada seperti layar pada monitor dan pembuatan *website*. Tetapi belum digunakan untuk aktivitas printing pada *printer*. Warna secondary pada group ini yaitu *RGB – Cyan*, *Magenta*, dan *Yellow* (Kuning). Ketiga warna ini adalah campuran dari warna dasar yaitu *RGB* (Young, 2007).



Gambar 2.2 *Color RGB* (Young, 2007)

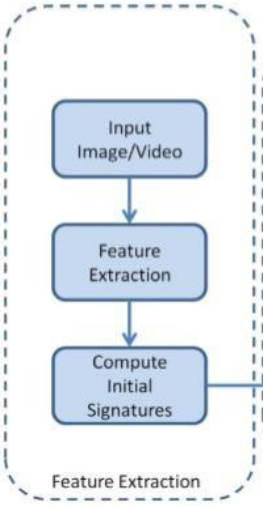
CMYK memilik 4 warna dasar yaitu *Cyan*, *Magenta*, *Yellow*, dan *K* (warna tambahan hitam). *CMYK* biasa dipakai pada aktivitas *printing* pada printer sebuah gambar (Young, 2007).



Gambar 2.3 *Color CMYK* (Young, 2007)

2.4 Metode Sistem

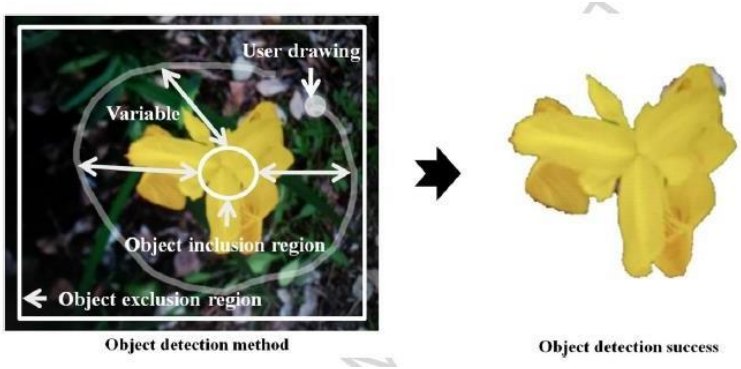
Praproses adalah suatu cara sebelum ke tahap *Active Learning* (Liaw, 2002) yang mana berfungsi agar pada tahap Activer Learning akan menghasilkan hasil yang memuaskan (Suzuki, 2016).



Gambar 2.4 Tahap Eksperimen (Lee, 2015)

Fitur *segmentation* dapat dilakukan pada sebuah object yang memiliki banyak *noise* atau object tersebut memiliki object lain di sekelilingnya. Maka dari itu dengan fitur *segmentation*, *object* yang kita ingin deteksi akan dipisah dari object disekitarnya. Caranya yaitu, pada tiap object yang penulis ingin deteksi

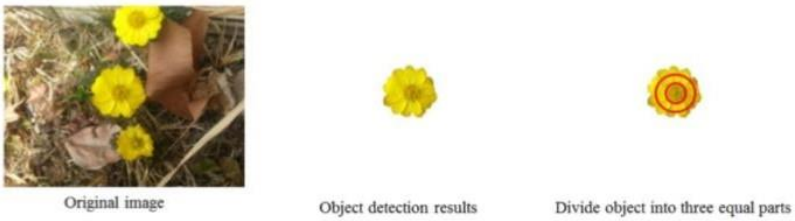
pasti memiliki warna *pixel* yang berbeda. Maka dari itu, jika warna pixel tersebut makin lebar makin jauh dari warna *pixel object* aslinya, maka bisa dipastikan bahwa *object* yang diluar tersebut sudah bukan dari *object* yang kita ingin deteksi.



Gambar 2.5 *Segmentation Feature* (Lee, 2015)

Hal ini akan lebih efektif untuk menentukan object yang ingin kita deteksi / pakai pada aktivitas *training model*. Karena kemungkinan besar *object* yang kita deteksi, intensitasnya akan berbeda dari object disekeliliingnya (Lee, 2015).

Setelah objek tersebut berhasil di *Segmentation*, objek tersebut di ekstraksi berdasarkan fitur-fitur yang terkandung didalamnya. Misalkan pada Gambar 2.5 yang memiliki beberapa fitur seperti part-part dalam bunganya, warna yang dimiliki oleh bunganya (karena warna kuning tersebut memiliki intensitas cahaya yang berbeda), dll. Hal tersebut dilakukan untuk proses training selanjutnya pada pembuatan model.



Gambar 2.6 Ekstraksi Fitur (Lee, 2015)

2.5 Pengertian dan Sejarah *Machine Learning*

Machine Learning adalah bagian dari ilmu komputer yang dapat membelajarkan komputer sehingga memiliki kemampuan untuk belajar tanpa

diprogram secara eksplisit (Arthur, 1959). *Machine Learning* merupakan bagian dari kecerdasan buatan yang berfokus dalam mempelajari, mendesain, dan membuat sebuah algoritma yang memiliki kemampuan untuk belajar dari data yang ada. Agar sebuah perangkat memiliki kecerdasan, maka komputer atau mesin tersebut harus dapat belajar. Dengan kata lain, *Machine Learning* berisi tentang keseluruhan proses pembelajaran komputer atau mesin menjadi cerdas dan dapat belajar dari data. *Machine Learning* sudah ada dan mulai digunakan sejak 50 tahun yang lalu dan sudah banyak digunakan di berbagai bidang. Contohnya pada bidang ekonomi, keilmuan, industri dan sebagainya.

Salah satu implementasi *Machine Learning* yang pernah dilakukan oleh Arthur Samuel sekitar 57 tahun yang lalu yaitu pembuatan permainan catur dengan komputer. Catur dipilih karena permainan sangat mudah tetapi memerlukan strategi yang bagus. Samuel membuat permainan catur ini berdasarkan pohon penyelesaian. Pencarian penyelesain dilakukan dengan menyusuri pohon permasalahan sampai mendapatkan solusinya.

Awal ditemukannya *Machine Learning* yaitu pada abad ke-20, seorang ilmuan dari Spanyol, Torres y Quevedo, membuat sebuah mesin catur yang dapat mengalahkan atau melakukan skakmat pada raja lawan dengan sebuah ratu dan raja. Perkembangan secara sistematis kemudian dimulai segera setelah diketemukannya komputer digital.

Artikel ilmiah pertama tentang Kecerdasan Buatan ditulis oleh Alan Turing pada tahun 1950, dan kelompok riset pertama dibentuk tahun 1954 di Carnegie Mellon University oleh Allen Newell and Herbert Simon. Namun bidang Kecerdasan Buatan baru dianggap sebagai bidang tersendiri di konferensi Dartmouth tahun 1956, di mana 10 peneliti muda memimpikan mempergunakan komputer untuk memodelkan bagaimana cara berfikir manusia. Mereka berhipotesis bahwa *Mekanisme berfikir manusia dapat secara tepat dimodelkan dan disimulasikan pada komputer digital.*

Machine Learning memiliki beberapa tipe dengan proses pembelajaran yang berbeda, tipe-tipe tersebut akan dijelaskan pada sub-bab berikutnya.

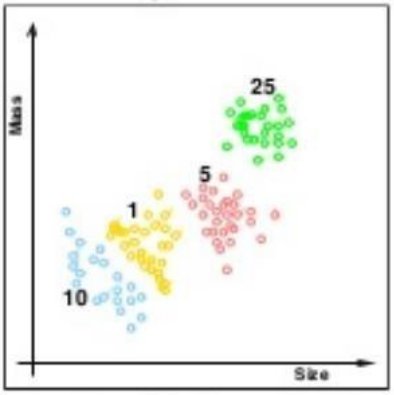
2.5.1 *Supervised Learning*

Tugas dari *Supervised Learning* terdiri dari pembangunan model yang

memetakan nilai *input* pada nilai ouput dimana *data* training tersedia (Riza, 2015). *Supervised Learning* adalah *Machine Learning* yang membutuhkan label sebagai tujuan dari pelatihan data atau data *training* (Mohri, dkk, 2012). *Supervised Learning* merupakan suatu pembelajaran yang dimana *output* tersebut diharapkan telah diketahui sebelumnya. Pada metode ini, setiap pola yang diberikan kedalam model *Machine Learning* telah diketahui *output*nya. Contoh algoritma dari salah satu bagian dari *Machine Learning* yaitu jaringan saraf tiruan yang mernggunakan metode *Supervised Learning* adalah hebbian (hebb rule), perceptron, adaline, boltzman, hapfield, dan backpropagation.

Berikut ini adalah beberapa contoh penerapan tipe *Machine Learning*, *Supervised Learning*:

1. Klasifikasi: sebuah metode untuk menyusun data secara sistematis menurut aturan-aturan yang telah ditetapkan sebelumnya (Muhammad, dkk, 2015). Dengan melakukan klasifikasi, dari data yang telah ada dapat dibuat sebuah model prediksi dengan *output* kelas.
2. Regresi: Analisis regresi adalah salah satu metode statistik untuk memprediksi nilai dari satu atau lebih variabel respon/dependen dari satu set variabel prediktor/independen (Johnson, 1982).



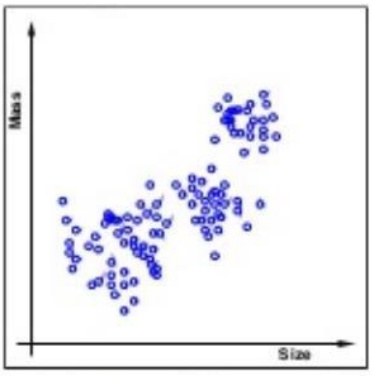
Gambar 2.7 Contoh *Supervised learning* pada pengenalan koin

Pada Gambar 2.7, diperlihatkan bagaimana klasifikasi dari pengenalan koin, terlihat sangat jelas lokasi bagian dari tiap kelas, seperti koin dengan nilai sepuluh terpisah dipaling bawah dengan warna biru, koin dengan nilai satu yang berwarna kuning tidak bercampur dengan yang lainnya, dan seterusnya.

2.5.2 *Unsupervised Learning*

Unsupervised Learning terdiri dari pembangunan model dari *data training* dengan tidak mengandung nilai *output* (Riza, 2015). *Unsupervised Learning* merupakan pembelajan yang tidak terawasi dimana tidak memerlukan target *output*. Teknik ini menggunakan prosedur yang berusaha untuk mencari partisi dari sebuah pola. *Unsupervised Learning* mempelajari bagaimana sebuah sistem dapat belajar untuk merepresentasikan pola *input* dalam cara yang menggambarkan struktur statistikal dari keseluruhan pola *input*. Berbeda dari *Supervised Learning*, *Unsupervised Learning* tidak memiliki target *output* yang eksplisit atau tidak ada pengklasifikasian *input*.

Dalam *Machine Learning*, teknik *Unsupervised* sangat penting. Hal ini dikarenakan cara kerjanya mirip dengan cara bekerja otak manusia. Dalam melakukan pembelajaran, tidak ada informasi dari contoh yang tersedia. Oleh karena itu, *Unsupervised Learning* menjadi esensial. Metode ini tidak dapat ditentukan hasilnya akan menjadi seperti yang diharapkan selama proses pembelajaran, nilai bobot yang disusun dalam proses *range* tertentu tergantung pada nilai *output* yang diberikan. Tujuan metode *Unsupervised Learning* ini agar kita dapat mengelompokkan *Unit-Unit* yang hampir sama dalam satu area tertentu. Pembelajaran ini biasanya sangat cocok untuk klasifikasi pola. Contoh algoritma jaringan saraf tiruan yang menggunakan metode *Unsupervised* ini adalah competitive, hebbian, kohonen, *Learning Vector Quantizatio (*LVQ), dan neocognitron.

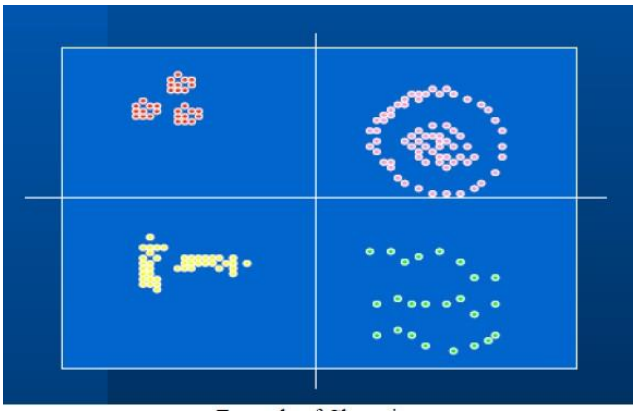


Gambar 2.8 Contoh *Unsupervised Learning* dalam pengenalan koin

Salah satu contoh dari *Unsupervised Learning* adalah clustering, sistem diharapkan mampu untuk memisahkan data serupa ke dalam kelompoknya masing-masing, seperti pada Gambar 8, belum diketahui kelas dari masing-masing data, mesinlah yang menentukan berdasarkan kedekatannya.

K-Means adalah salah satu algoritma untuk melakukan *Clustering* yang biasa digunakan dan juga simple dalam pengerjaanya. Oleh karena itu, *K- Means* tidak akan memakan banyak memory dalam proses men-Cluster sebuah data. Clustering termasuk teknik *Unsupervised Learning* (Kapugama, 2016).

Clustering adalah sebuah state untuk mempartisi sebuah data set yang diberikan menjadi *sub group* yang nantinya *sub group* tersebut akan mempunya karateristiknya sendiri. Masalah pada Clustering penting untuk dioptimasi dalam banyak field, *pattern recognition*, *learning, source coding*. Setelah menjadi *sub group*, akan dapat disimpulkan oleh penulis bahwa dari banyak data tersebut akan memiliki sebuah titik yang dimana titik tersebut yaitu karateristik dari data (Rose, 1998).



Gambar 2.9 *Clustering* (Agarwa, 2003)

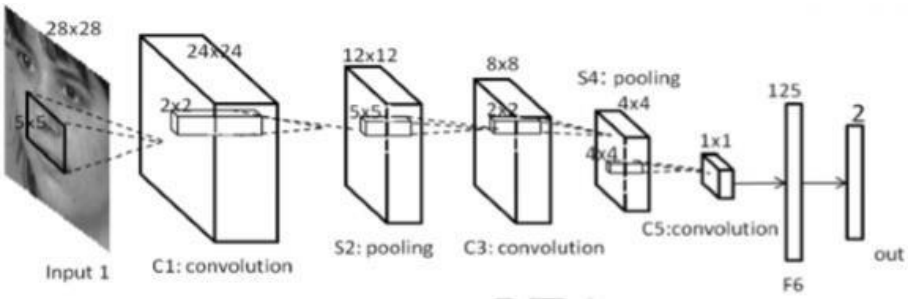
Inti dalam Clustering yaitu memutuskan untuk membuat pengelompokkan yang baik yang memiliki kesamaan yang sama percis dengan data yang lainnya. Komponen yang penting dalam *cluster* yaitu kesamaan jarak antara data-data tersebut yang bias dihitung menggunakan Algoritma seperti *K- Means* salah satunya (Agarwa, 2003).

Dalam menentukan sebuah jarak pada berbagai data, bisa menggunakan algoritma. Dan jika rumus tersebut berbeda, otomatis hasil pada perhitungan jarak akan berbeda pula. *Domain Knowledge* diperlukan untuk mengukur jarak antar data. Maka dari itu banyak algoritma-algoritma berbeda yang merupakan *upgrade*-an dari algoritma aslinya yang bertujuan agar hasil dari algoritma baru tersebut menjadi lebih baik dari algoritma versi yang sebenarnya (Agarwa, 2003).

2.6 Convolutional Neural Network (CNN)

Neural Network adalah suatu algoritma yang meniru pergerakan kerja otak manusia yang didalamnya memiliki banyak Neuron bahkan bisa mencapai milyaran Neuron. Tetapi jika didalam *Neural Network*, tidak memiliki milyaran dikarenakan terbatasnya memory komputer dan hal lainnya. (Lee, 2009).

Convolutional Neural Network adalah *Neural Network* yang memiliki banyak layer dan tiap layernya memiliki beberapa bidang dua dimensi dan masing-masing bidang terdiri dari beberapa Neuron Independen (Zhan, 2015).



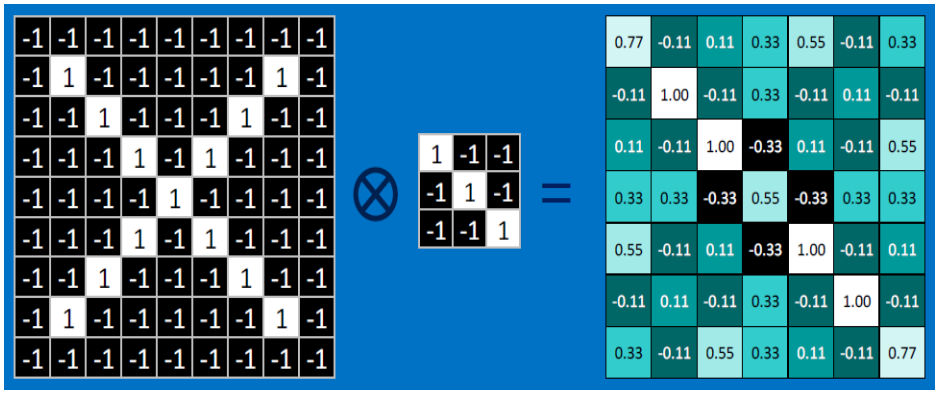
Gambar 2.10 Tahapan Algoritma *CNN* (Zhan, 2015)

Umumnya CNN biasa digunakan untuk mengklasifikasi Image maka dari itu akan menghasilkan label kelas yang paling mungkin terkatir dengan gambar Testing (Wang, 2016). CNN sendiri sangat baik untuk klasifikasi Image, bahkan tidak cocok untuk men-*Train* data lainnya. (Lee, 2009).

Suatu gambar awalnya akan diubah menjadi *array of pixel* yang dimana object akan diubah menjadi 1 dan -1. 1 untuk object yang terdeteksi, -1 untuk object selain object yang ingin dideteksi. Setelah gambar tersebut diubah menjadi *array of pixel, array* tersebut akan di train ke CNN. CNN sendiri memiliki 3 tahap

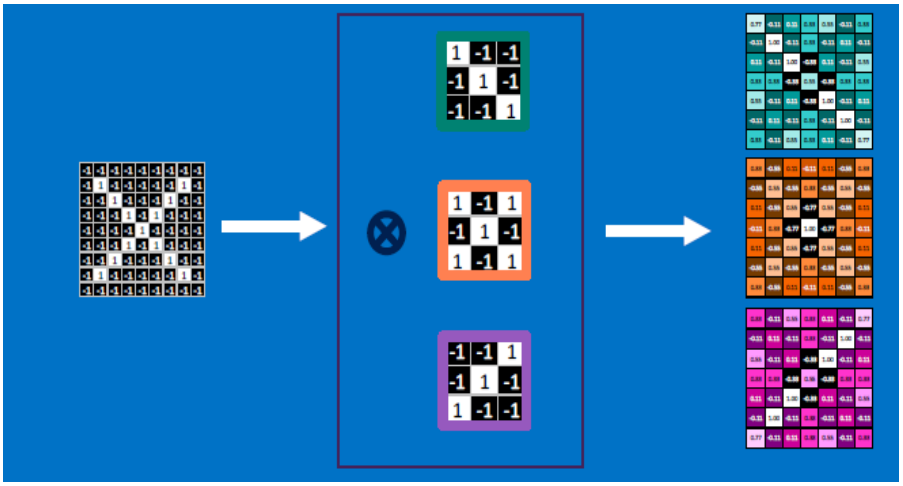
yaitu *Convolution*, *ReLU*, *Pooling Layer*. Tetapi sebelum masuk ke 3 tahap tersebut, array akan dipecah menjadi bagian- bagian yang dimana bagian tersebut memiliki ukuran 3x3 yang berfungsi untuk perhitungan pada tahap *Convolutional* (Han, 2016).

Pada tahap *Convolution*, bagian-bagian yang berbentuk 3x3 tersebut akan dipakai untuk mengkalikan pada *array of pixel* dari gambar yang ingin di- train.



Gambar 2.11 Tahapan *CNN* Pertama (Rohrer, 2016)

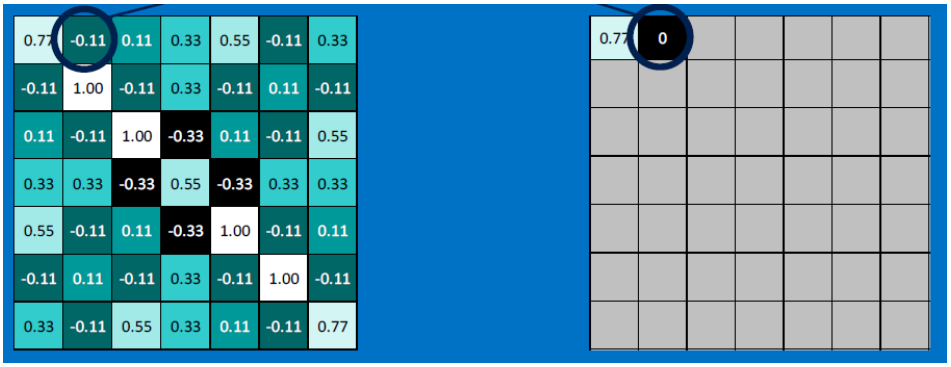
Pada Gambar 2.11, *array of pixel* dari gambar yang ingin di-train akan dikalikan dengan bagian-bagian yang telah ditentukan sebelumnya. Semua bagian harus dikalikan untuk menghasilkan data yang lebih banyak dan lebih akurat.



Gambar 2.12 Tahapan *CNN* Kedua (Rohrer, 2016)

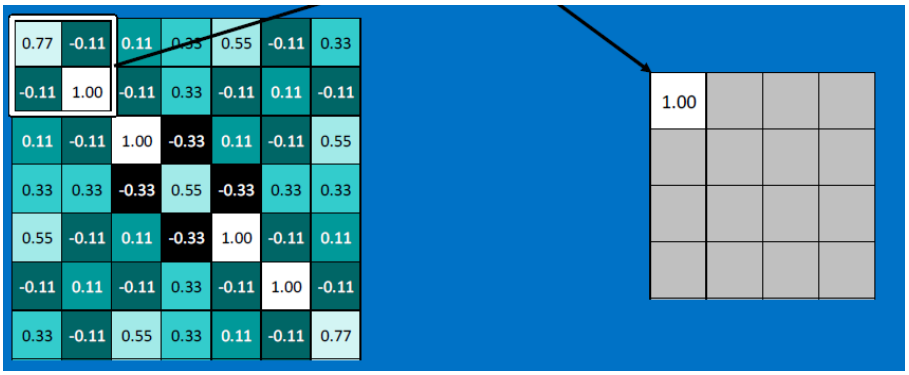
Selanjutnya memasuki tahap *ReLU*. Pada tahap ini, hasil dari *Convolutional* akan diubah pada angka yang bernilai negative menajdi 0. Tetapi, angka yang

bernilai positif akan ditulis seadanya angka tersebut *(Rohrer, 2016)*.



Gambar 2.13 Tahapan *CNN* Ketiga (Rohrer, 2016)

Setelah tahap *ReLU* selesai, hasil pada Tahap *ReLU* akan masuk pada tahap pooling. Pada tahap ini akan di dipilih nilai terbesar dari setiap array 2x2 / 3x3nya pada setiap bagian.

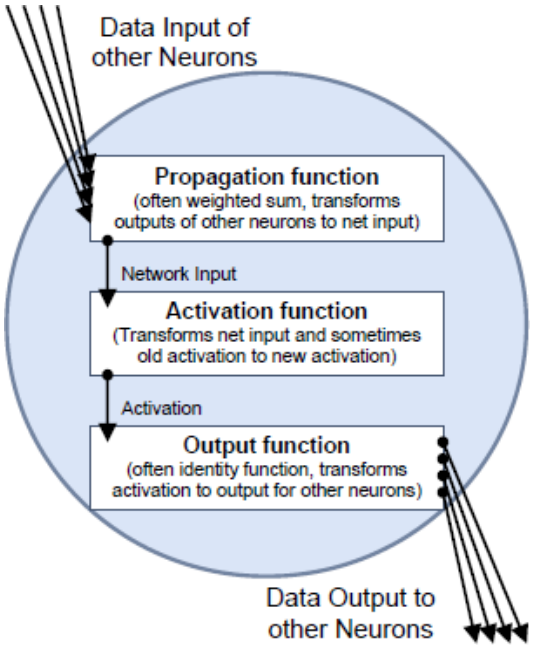


Gambar 2.14 Tahapan *CNN* Keempat (Rohrer, 2016)

Pada tahap menggunakan algoritma *Neural Network* dengan teknik *Backward Propagation*, terdapat input yang berupada bobot dari setiap inputannya, *learning rate*, *epoch* (pengulangan pada waktu training data tersebut). Epoch tersebut bias membuat *training* data menjadi lebih bagus dalam hal *accuracy* dan otomatis pada tahap *testing* akan lebih akurat untuk mengenali sebuah *image* maupun *video* (Roghganger, 2011).

Neural network memiliki sebuah fundamental yang dimana terdiri dari Input sebuah data yang nantinya akan masuk ke tahap fungsi propagation. Di fungsi propagation biasanya masukan data tersebut akan memasuki rumus terdiri dari berat (w), *neuron* yang telah ditentukan. Setelah itu akan masuk kedalam fungsi

aktivasi. Fungsi aktivasi bertujuan untuk menentukan bahwa hasil data tersebut akan menjadi seperti apa nantinya. Fungsi aktivasi yang paling sederhana yaitu mengetahui x > 0 atau x < 0 dan itu adalah tujuan dari *neural network* sendiri. Setelah memasuki tahap fungi aktivasi, akan memasuki *output* yang berupada model. Model ini akan dipakai untuk bahan testing (Kriesel, 2005).



Gambar 2.15 Tahapan Garis Besar CNN

Didalam fungsi propagation, terdapat berat (w) yang dikalikan dengan masukan data yang berupa *image of array* dan sebanyak neuron yang ditentukan oleh inputan user.

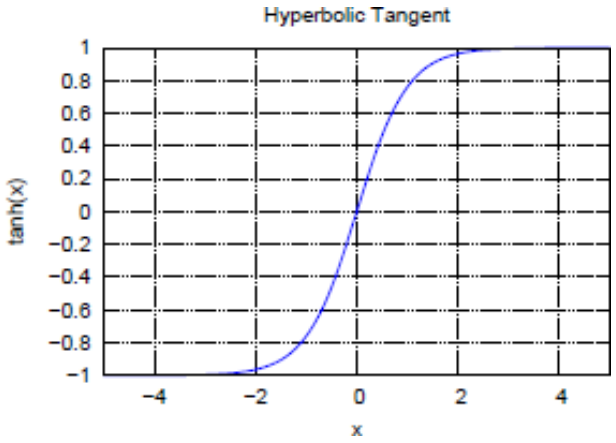
$$\mathrm{net}_j = \sum_{i \in I} (o_i \cdot w_{i,j})$$

Setelah semua dikalikan akan dijumlahkan pada setiap datanya. Tanda Sigma pada rumus tersebut adalah penjumlahan dari banyaknya data yang ada dari image of array. Biasanya pada algoritma neural network terdapat sebuah bias yang berfungsi untuk mendeklarasikan error pada perhitungannya (Rojas, 1996). Pada tahap fungsi aktivasi, akan ditentukan bahwa hasil data tersebut ingin bagaimana. Penentuan angka tersebut biasa disebut dengan Treshold value. Perhitungan yang mudah yaitu dengan menggunakan fungsi “if” sederhana pada setiap hasilnya.

$$y = f(net) = \begin{cases} 1, & net > 0{,}5 \\ 0, & -0{,}5 \le net \le 0{,}5 \\ -1, & net < 0{,}5 \end{cases}$$

Fungsi aktivasi sendiri bersifat global, maksudnya global yaitu pada setiap neuron akan terdapat fungsi aktivasi. Beberapa orang mengatakan fungsi aktivasi dengan sebutan *transfer function*.

Dalam tahap fungsi aktivasi, biasanya orang-orang menggunakan fungsi sigmoid. Karena dipercaya akan lebih akurat untuk menentukan *accuracy* pada akhirnya. Biasanya, learning tidak akan sesempurna 100% tetapi dengan *accuracy* 99% sudah dapat menentukan identitas dari gambar atau data tersebut *(Rojas, 1996)*.



Gambar 2.16 Fungsi Aktivasi Sigmoid (Rojas, 1996)

$$y = f(x) = \frac{1}{1 + e^{-x}}$$

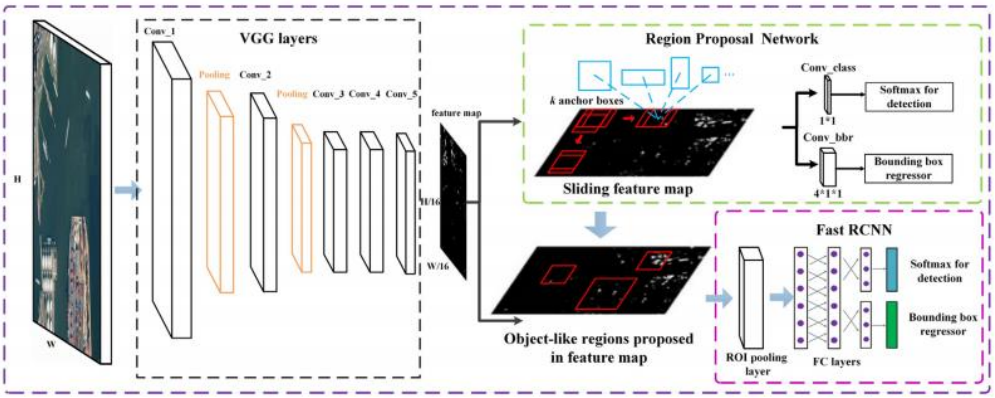
$$f'(x) = f(x)(1 - f(x))$$

Pada tahap back propagation, terdapat tambahan yaitu *backward propagation* yang bertujuan untuk mengurangi *error* pada setiap *learning* yang dilakukan *forward backpropagation*. Maka dari itu akan terdapat *update* bobot dan juga bias yang akan digunakan pada *forward backprogation* nantinya.

2.7 Arsitektur *Faster R-CNN Base VGG16*

Faster R-CNN adalah arsitektur *CNN* yang memiliki 2 tahap yaitu *RPN* (gambar diproses oleh fitur ekstraksi) dan *CNN Model* (*VGG16*). Fitur maps

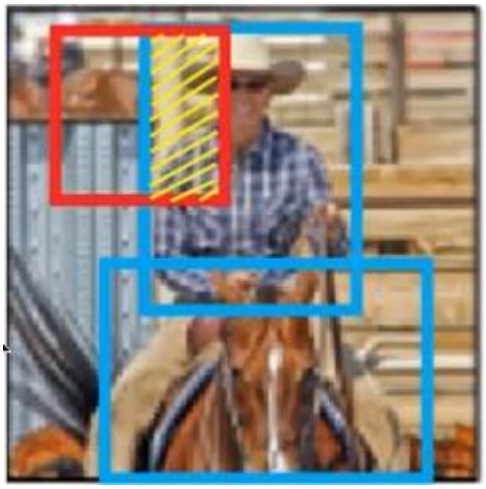
digunakan untuk memprediksi proposal *bounding box* sedangkan tahap ke dua, proposal ini digunakan untuk memotong fitur dari fitur maps yang kemudian dipakai untuk klasifikasi dan regresi objek terkait.



Gambar 2.17 Arsitektur *Faster R-cnn Base VGG16* (Deng, 2018)

Dari gambar diatas, terdapat beberapa tahap pada arsitektur *Faster R-CNN*

1. Gambar yang telah dipraproses akan masuk ke CNN Model (penulis menggunakan VGG16 untuk arsitektur layer CNN nya) yang akan menghasilkan fitur map untuk image tersebut. *VGG16* memiliki total 16 layer. Citra yang dimasukan pada arsitektur *Faster RCNN* memiliki resolusi *height* 600.
2. Fitur map tersebut akan masuk ke tahap RPN. Hasil dari tahap RPN yaitu penulis sudah dapat object yang diproposal. *Regional Proposal Network* memiliki tujuan untuk mendapatkan fitur map yang telah di *propose* objeknya. Pada tahap ini memiliki yang namanya *anchor*. *Anchor* tersebut berfungsi untuk men-*justify* suatu objek dengan bantuan metode sliding window. *Anchor* memiliki *scale* 128, 256, 512 dengan *ratio* 1:1, 1:2, dan 2:1.



Gambar 2.18 Anchor pada *Regional Proposal Network*

Pada gambar 2.18 bisa dilihat terdapat *box* warna merah (*anchor)* dan *box* warna biru (*ground truth box*). Pada kedua *box* tersebut terdapat irisan warna kuning yang dinamakan *intersect of union* atau *IoU*. *IoU* adalah perhitungan dari kotak warna biru dibagi kotak warna merah yang dimana jika hasil dari *IoU* tersebut >0.7, sistem akan men-*justify* kalo kotak tersebut adalah objek yang akan dideteksi. Untuk menghitung nilai *loss* pada tahap *regional proposal network* yaitu:

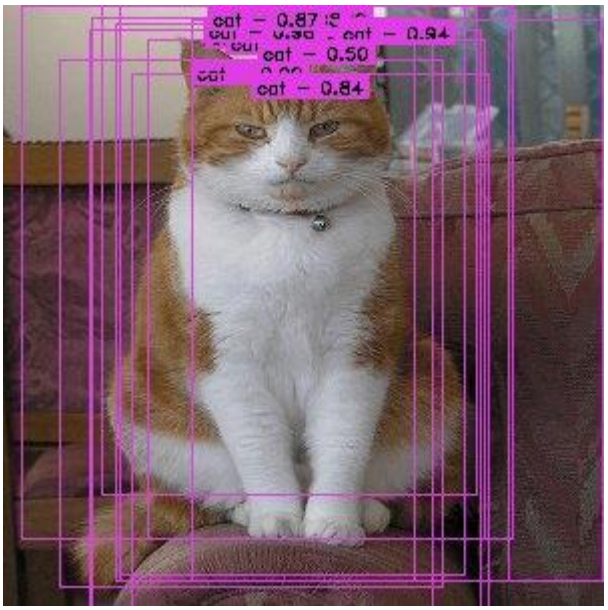
$$\mathcal{L}(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i \mathcal{L}_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{box}} \sum_i p_i^* \, . \, L_1^{smooth}(t_i - t_i^*)$$

Catatan :

- Pi = prediksi probabilitas anchor i menjadi object
- Pi* = ground truth label, anchor i adalah sebuah object
- Ncls = minibatch size
- Lcls = softmax loss function
- Lamba = constant value
- Nbox = total anchor
- Lreg = smooth function
- Ti = prediksi box
- Ti* = ground truth box

3. Setelah mengetahui proposal setiap objek, masuk ke tahap *ROI Pooling Layer* yang menghasilkan ukuran yang sama pada tiap objek yang

diproposal. Fitur map yang didapatkan pada tahapan *Regional Proposal Network* akan memiliki banyak *box*.



Gambar 2.20 Contoh *Ouput Regional Proposal Network*

Dari hasil tahapan *regional proposal network*, maka harus ada tahapan untuk mem-*fixing box* yang sebenernya pada 1 objek. Maka dari itu, fungsi pada tahapan ini yaitu untuk menyimpulkan *box* yang sebenernya pada hasil dari *regional proposal network*.

4. Tahap terakhir, proposal pada tahap ke tiga masuk ke *fully connected layer* yang memiliki *softmax layer* dan linear regresi layer. Hasil dari tahap ini akan bisa diuji coba untuk mengklasifikasi suatu objek.

RPN pada arsitektur ini memiliki 9 anchor yang berfungsi untuk menandai kemungkinan suatu objek ada pada gambar yang ditraining. Jadi, pada 1 objek akan terdapat banyak kotak yang nantinya akan ditentukan pada tahap *ROI Pooling Layer*.

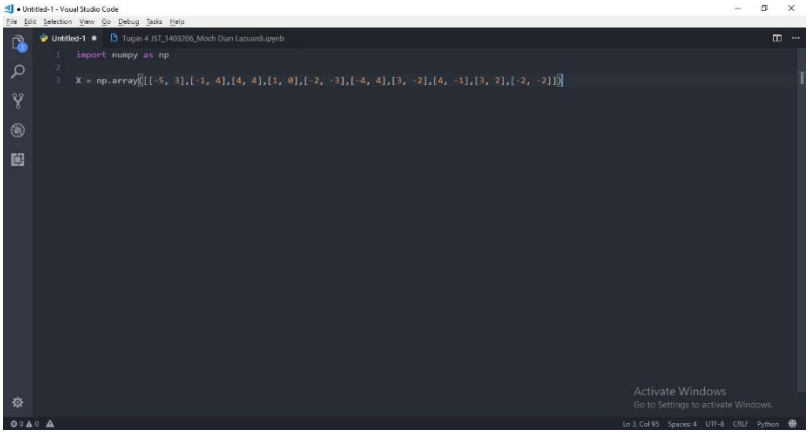
2.8 Python

Python adalah Programming Language yang fleksibel, simple coding, dll. Selain itu juga, python sudah men-support untuk bahasa *Object-Oriented*. Dengan kelebihan python sendiri yaitu:

1. Python sangatlah cepat dan *powerful*

   Python memiliki banyak *library* didalamnya yang dapat digunakan untuk membuat aplikasi. Dengan *Library* tersebut, pengguna dapat membuat aplikasi

dengan hanya 3 baris code saja.

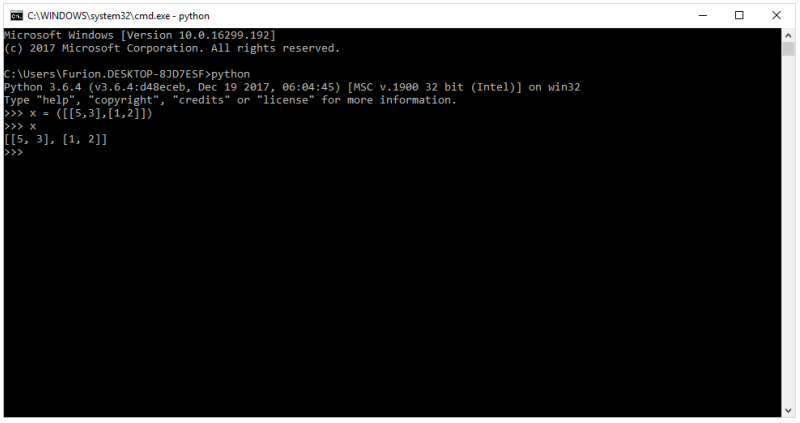


Gambar 2.18 Contoh *Syntax Python* Pembuatan *Array*

2. Python sudah dapat men-*support* banyak teknologi terbaru

Bahasa python dapat ditulis dengan *syntax* C++ atau Java. Java sendiri sudah populer di masyarakat. Maka dari itu dengan menggunakan python, pengguna tidak perlu berpikir panjang untuk belajar dari awal.

3. Coding menggunakan *python* sangatlah sederhana



Gambar 2.19 Contoh *Syntax Python* Pembuatan *Array*

4. Python bersifat *Open-Source*

Python bersifat *Open-Source.* Yang artinya, masyarakat dapat menggunakannya dengan bebas dan gratis selain untuk analisis data, Python dapat digunakan oleh aplikasi lain seperti:

5. Network dan *Internet Programming*

Banyak *library python* yang dapat digunakan untuk pemrograman

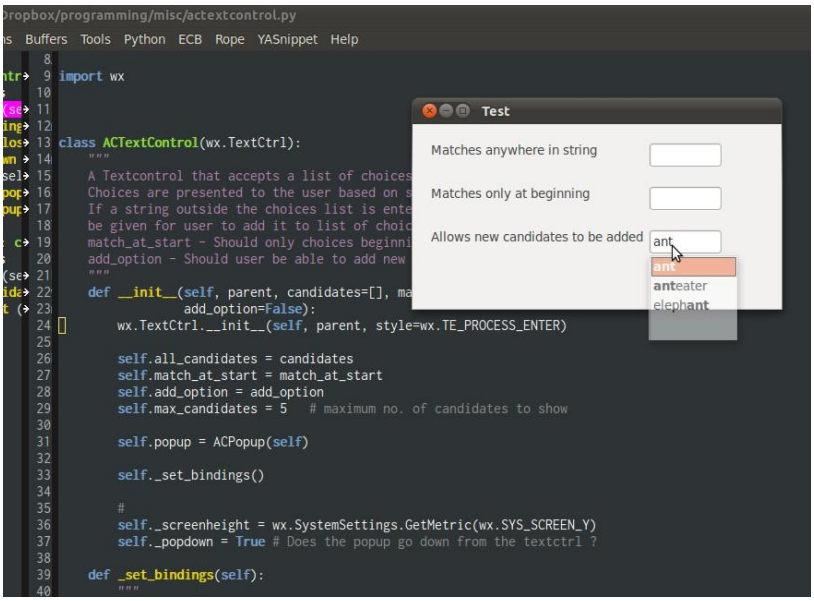
jaringan. Misalnya, koneksi *Client-Server*, *FTP*, *Telnet*, dll. Beberapa *tool* lainnya juga seperti mod-python mengizinkan server web seperti apache menjalankan script Bahasa *python*. Selanjutnya, beberapa program popular seperti *Django* dapat mendukung *script* Bahasa *python* juga.

6. *Numeric Programming*

Python memiliki *library Numpy* yang didalamnya dapat men-*handle* fungsi yang sama dengan *software numeric* lainnya seperti *Fortran*. Dengan *syntax* yang sederhana, *python* dapat meningkatkan performanya pada hal kecepatan komputasi. Ditambah *library Numpy* dapat dikolaborasikan dengan *library* lainnya.

7. Pembuatan *GUI*

Tkinter adalah *software* berorientasi objek standar yang didistribusikan dengan penerjemah python. *Software* ini menyediakan *tool* untuk pembuatan GUI dalam Bahasa python. Selain itu, terdapat *software* lain seperti *wxPython* yang berbasis pada *library C++*. Kedua *software* tersebut dapat merancang GUI dengan Bahasa *python*.



Gambar 2.20 Pembuatan Aplikasi DesktopMenggunakan Python (wxPython)

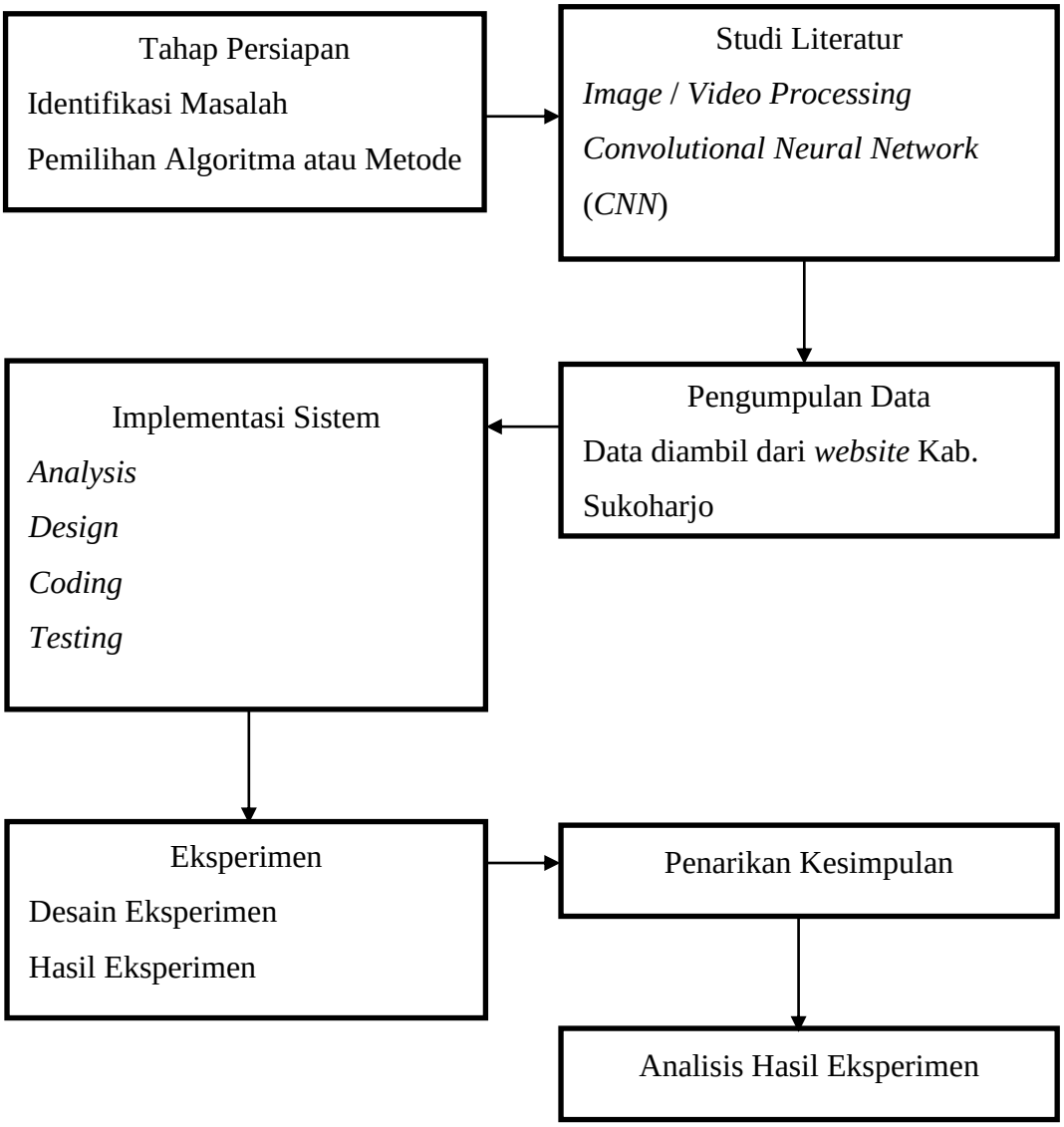
8. *System Programming*

Python menyediakan Bahasa yang cocok untuk pemrograman system. Antarmuka ini menyediakan beberapa fungsi operasi file, pemrosesan parallel, dll.

BAB III
METODE PENELITIAN

3.1 Desain penelitian

Untuk menunjang penelitian ini, dibutuhkan sebuah desain penelitian yang mewakili proses-proses yang akan dilakukan untuk mencapai hasil akhir dari penelitian tersebut.



Gambar 3.1 Desain Penelitian

Fase yang terdapat pada desain penelitian memiliki 6 fase yang dapat mengeluarkan data berbeda-beda

1. Tahap Persiapan

   Tahap persiapan adalah tahap awal dari penelitian, tahap ini dimulai dari

identifikasi masalah, kemudian merumuskan masalah, lalu menentukan algoritma atau metode yang akan digunakan untuk menyelesaikan masalah tersebut.

2. Studi Literatur

Dalam studi literatur, peneliti melakukan tahap mempelajari materi yang terkair dengan *Convolutional Neural Network*, *Image Processing*, *Video Processing* dan penggunaan Machine Learning pada Deteksi *Image*. Dalam mempelajari tentang bahasan di atas penulis mempelajari dari beberapa sumber, seperti buku, jurnal, juga *internet*, ataupun bahan bacaan lainnya yang didapat dari berbagai sumber.

3. Pengumpulan Data

Pengumpulan data dilakukan dengan mendownload video dari *website* Kab. Sukoharjo

4. Pengembangan Sistem

Tahap pengembangan *system* merupakan tahap pembuatan program dengan menggunakan Bahasa *Python* untuk mengolah sebuah *Image* / *Video* yang telah dikumpulkan melalui bagian Pengumpulan data. Data tersebut akan diproses terlebih dahulu menjadi *Array* dan akan dilakukan pra-proses terlebih dahulu sebelum ke tahap training sebuah data menggunakan Algoritma arsitektur *Faster R-CNN Base VGG16*. Pada pertama yaitu analisis, pada tahap ini peneliti menganalisis bagaimanana programat akan dibuat. Tahap selanjutnya adalah design, pada tahap ini peneliti membuat *design* kode program dengan lebih “Cantik” agar user dapat menggunakan kode program tersebut dengan mudah. Tahap ketiga yaitu coding, pada tahap ini penulis akan memulai membuat program menggunakan *Python* dengan bantuan *Library* seperti keras (untuk penggunaan *Convolutional Neural Network*),

5. Tahap Eksperimen

Setelah program dibuat, tahap selanjutnya adalah tahap eksperimen. Pada tahap ini program diuji coba sesuai dengan tujuannya.

6. Tahap Penarikan Kesimpulan

Tahap ini program akan ditarik kesimpulan berdasarkan eksperimen yang telah dilakukan.

Adapun metode yang dilakukan dalam penelitian ini dibagi kedalam dua

bagian, yaitu metode pengumpulan data dan metode perancangan perangkat lunak.

3.3 Metode Penelitian

3.3.1 Metode pengumpulan data

Penulis berusaha mendapatkan data yang akurat dan mampu menunjang penelitian, adapun dalam penelitian ini metode pengumpulan data dilakukan sebagai berikut:

1. Studi Literatur

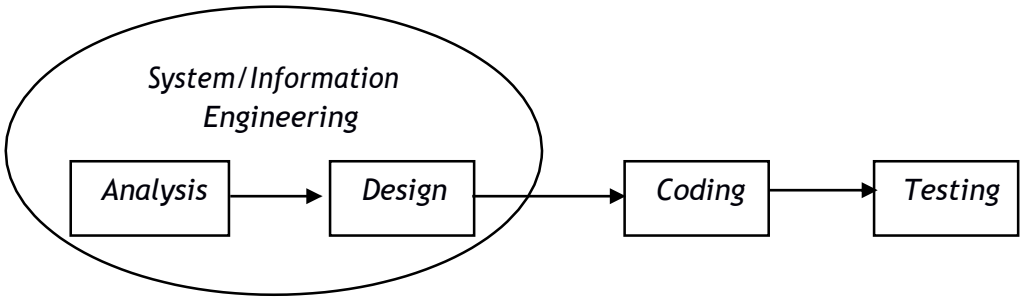
Studi literatur dilakukan dengan mempelajari teori dan konsep mengenai *Convolutional Neural Network*, *Image Processing*, penerapan *Machine Learning* pada *Image Processing*, dengan pendekatan *Neural Network* yang menjadi pendukung dalam penelitian ini melalui jurnal, *papper*, *textbook*, artikel, dan sumber-sumber ilmiah lainnya dari internet.

2. Validasi Data

Pada tahap ini, data di didapat dari website dishub Kabupaten Sukoharjo.

3.3.2 Metode pengembangan perangkat lunak

Dalam penelitian ini, dilakukan pengembangan perangkat lunak menggunakan model *Linear Sequential*. *Linear Sequential* mengusulkan sebuah pendekatan kepada pengembangan perangkat lunak yang sistematik dan sekuensial yang dimulai pada tingkat dan kemajuan sistem pada seluruh analis, desain, kode, pengujian, dan pemeliharan. Berikut adalah proses gambaran dari *Linear Sequential Mode*.



Gambar 3.2 Model Sekuensial Linear

1. *System / Informartion engineering*

Merupakan bagian dari sebuah sistem terbesar yang mana dalam pengerjaannya dimulai dengan menetapkan berbagai kebutuhan dari semua

elemen yang diperlukan sistem dan mengalokasikannya ke dalam pembentukan perangkat lunak.

2. *Analysis*

Analisis perangkat lunak merupakan tahap menganalisa hal-hal yang diperlukan dalam pembentukan sebuah perangkat lunak.

3. *Design*

Desain merupakan beberapa langkah proses yang berfokus pada empat buah atribut yang berbeda dari program, yakni struktur data, arsitektur perangkat lunak, representasi antarmuka, dan sebuah algoritma.

4. *Coding*

Hasil dari pembuatan disain haruslah diartikan ke dalam bentuk yang bisa dimengerti oleh mesin. Sehingga komputer bisa merepresentasikan ke dalam bentuk perangkat lunak.

5. *Testing*

Tes merupakan langkah paling akhir yang dikerjakan, sebuah pengetesan pada perangkat lunak yang sudah melalui beberapa tahap dan dapat dipakai oleh user, dalam tes juga dapat di lakukan pengecekan apakah perangkat lunak yang dibuat sudah memenuhi standarisasi atau belum.

3.2 Alat dan Bahan Penelitian

Untuk menunjang penelitian yang akan dilakukan, maka diperlukan alat dan bahan agar mendapatkan hasil yang baik dan terstruktur.

Adapun alat yang digunakan dalam penelitian ini, diantaranya:

a. Perangkat Keras
b. Processor 6 Core
c. Ram 56GB
d. GPU Tesla K80
e. Mouse dan Keyboard

Adapun perangkat lunak yang digunakan dalam penelitian ini, diantaranya:

a. Windows 10 Pro 64 Bit
b. Visual Studio Code
c. Browser Google Chrome

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1 Pengumpulan Data

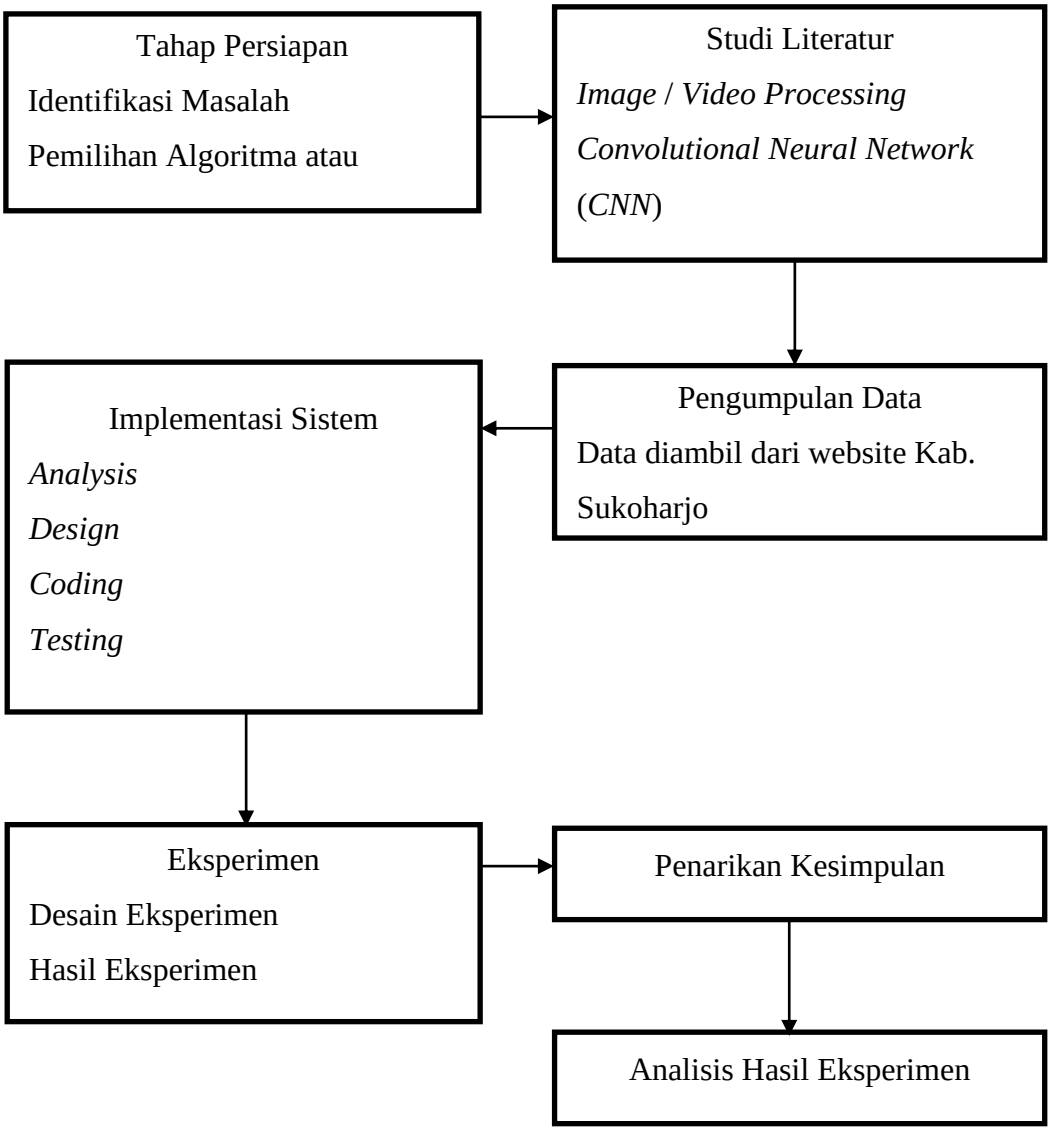
Pengumpulan data dilakukan dengan observasi secara langsung melalui *CCTV* di *website* resmi Dishub Sukoharjo (Link). Terdapat persimpangan jalan di *website* resmi Dishub Sukoharjo, tetapi hanya beberapa *CCTV* yang terlihat jelas dan memiliki kualitas resolusi 480 x 360 pixel setelah di *screenshoot* menggunakan aplikasi *VLC*.

Pihak *ACTS* menyiarkan video secara *live* di *website* resminya. Oleh karena itu, penulis menyimpan video tersebut dengan bantuan *software VLC* yang dapat disimpan di *hardisk* penulis.

Gambar 4.1 Contoh dari *video* rekaman *CCTV*

Video yang digunakan pada penelitian ini diambil pada tanggal 12 September 2018 sekitar jam 08.25 sampai dengan jam 08.40. Dengan begitu penulis mendapatkan 7 video yang terpisah dan memiliki ekstensi .avi. Penulis mengambil video pada pagi hari dikarenakan penulis berpikir bahwa pada saat itu banyak orang yang ingin berpergian semisal ke kantor, sekolah, dll. Selain itu, pagi hari memiliki cahaya yang cerah yang menyebabkan kualitas *video* semakin terlihat.

4.2 Implementasi



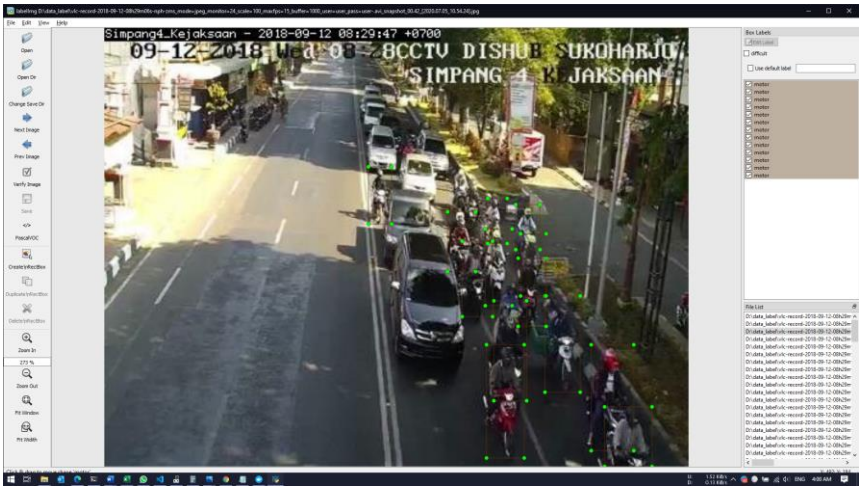
Gambar 4.2 Alur Proses Sistem Pendeteksi Sepeda Motor

Pada penelitian ini, sistem pendeteksi motor dibangun dengan menggunakan metode *Faster R-CNN* (*Base CNN* dengan arsitektur *VGG16*) yang digunakan untuk melatih data pada proses pengenalan citra. Gambar 4.2 menunjukan alur proses dari sistem pendeteksi sepeda motor.

4.2.1 Pengambilan dan transformasi citra

Data input hasil rekaman *CCTV* berupa *video* dengan formap .avi maka dari itu data *input* harus diubah menjadi data citra terlebih dahulu agar bisa diproses ke tahap selanjutnya. Proses pengambilan dan

transformasi citra dilakukan secara manual dengan menggunakan fitur *screenshot* pada *software VLC*. Setelah mendapatkan potongan citra, penulis menggunakan citra tersebut menjadi konversi *Pascal-VOC* dengan bantuan tool *labelImg*. Penulis hanya mengambil 60 citra (bukan motor) yang mana 1 citra tersebut memiliki lebih dari 1 objek didalamnya.



Gambar 4.3 Konversi Citra Menggunakan *Tool LabelImg*

4.2.2 Pra-Proses

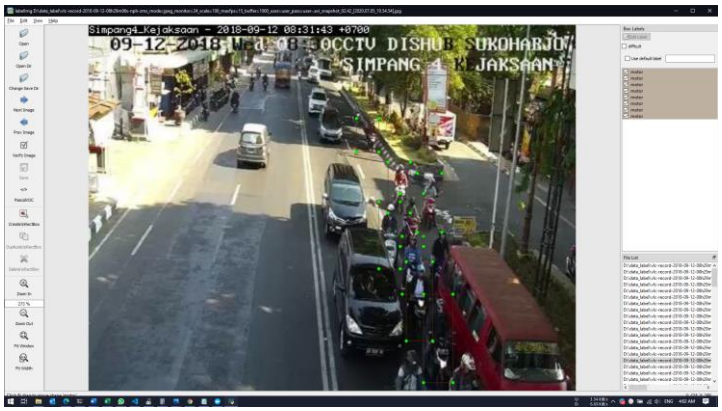
Citra yang diperoleh pada tahap sebelumnya tidak dapat langsung digunakan untuk training dan testing. Citra tersebut harus diolah terlebih dahulu agar citra siap untuk diproses oleh sistem sehingga informasi yang dihasilkan dapat digunakan pada tahap selanjutnya. Proses pengolahan citra tersebut ada pada tahap pra proses.

Pra proses yang digunakan terdiri dari pelabelan data dan *scalling* citra. Dengan kedua cara itu, citra sudah bisa ditraining menggunakan algoritma *Faster RCNN*.

1. Proses pelabelan object pada sebuah citra

Data set terdiri dari 1 kelas yaitu kelas motor. Tujuan pelabelan ini yaitu sebagai penanda masing-masing objek dalam sebuah citra sehingga memudahkan sistem pada proses *training*. Label yang digunakan dikonversi kedalam bentuk vektor. Gambar 4.5 menunjukan contoh pelabelan menggunakan aplikasi *labelimg* yang dimana label gambar

tersebut akan memiliki file *.xml* atau bisa dibilang *pascal VOC*.



Gambar 4.4 Contoh Pelabelan pada Citra

2. Proses *scalling*

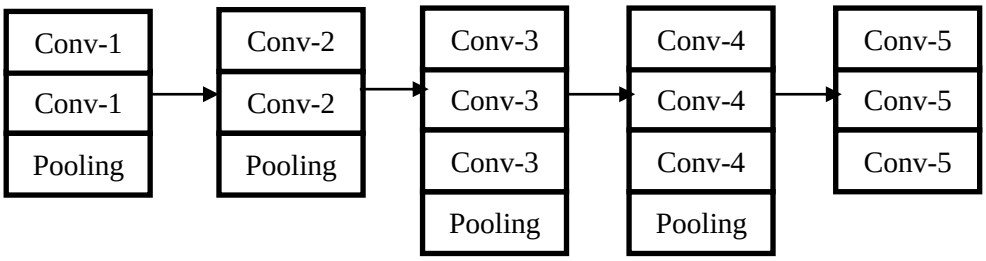
Data set yang ada sebelumnya memiliki dimensi 480 x 360, oleh karena itu data harus di *resize* agar bisa masuk ke tahap training menggunakan *Faster RCNN*. Proses untuk mengubah dimensi citra ini dinamakan *scalling*. Pada proses ini setiap citra yang awalnya memiliki dimensi 480 x 360 akan diubah menjadi 600 x 800. Pada python menggunakan *library opencv*. Perintah untuk mengubah dimensi dapat dilakukan dengan menggunakan perintah *cv2.resize()*.

4.2.3 *Training*

Pada tahap ini akan dilakukan dua proses yaitu *training* dan *testing*. Pada proses *training* dilakukan pengambilan model hasil *training*, sedangkan pada proses testing yaitu proses pengujian seberapa baik model tersebut melakukan klasifikasi pada sepeda motor. Proses pengambilan ciri citra ini menggunakan metode *Faster RCNN* (*Base VGG16*). Ada beberapa tahapan dalam menggunakan metode ini, yaitu:

1. Langkah pertama membuat layer arsitektur *VGG16* yang akan digunakan. *VGG16* memiliki 16 *layer convolutional* dan 4 *layer max pooling* pada setiap bagian yang bisa dilihat pada gambar 4.5. Ukuran citra yang masuk pada *VGG16* yaitu 600 x

800 pixel.

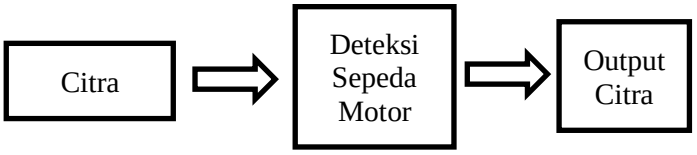


Gambar 4.5 Arsitektur *VGG16*

2. Langkah kedua membuat proses *RPN*. Hasil dari Langkah pertama akan dimasukan ke proses *RPN*.
3. Langkah ketiga membuat proses *ROI Pooling Layer*. Hasil dari tahap kedua akan masuk ke tahap ini untuk menghasilkan objek yang lebih proposional.

4.2.4 Testing

Setelah melakukan proses *training* dan *testing*, maka sistem telah menyimpan ciri dari *dataset training*. Langkah selanjutnya adalah menggunakan ciri citra tersebut untuk mendeteksi sepeda motor. Alur proses untuk mendeteksi sepeda motor digambarkan pada gambar 4.6.



Gambar 4.6 Alur Deteksi Sepeda Motor

1. Citra

   Tahapan ini yaitu memasukan citra berukuran bebas kedalam sistem.
2. Deteksi Sepeda Motor

   Dalam tahapan ini, citra yang telah diinputkan akan diproses oleh sistem untuk dideteksi objek sepeda motor.
3. *Output* Citra

Dalam tahapan ini, sistem akan menyimpan gambar yang telah diproses dan telah memiliki *bounding box* jika terdapat suatu objek sepeda motor yang terdeteksi. Untuk hasilnya bisa dilihat pada gambar 4.7.

4.3 Pengujian

4.3.1 Skenario pengujian

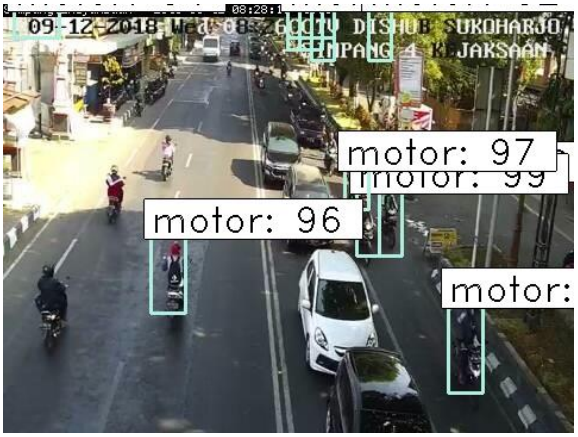
Tujuan penulis hanya mendeteksi motor pada lalu lintas, maka dari itu penulis hanya memakai 1 klasifikasi pada pengujian. Data training yang digunakan oleh penulis sebanyak 284 objek motor pada percobaan pertama dan 620 objek motor pada percobaan kedua dan dilakukan 2 kali percobaan yaitu 35 *epoch* dan 37 epoch. Penulis menginginkan tahap training ini mencapai 100 *epoch*, tetapi karena kendala device yang membuat training lama maka dari itu penulis memberhentikan di *epoch* 35 dan 37.

Aplikasi yang dibuat oleh penulis menggunakan *CNN* dengan arsitektur *Faster R-CNN (Base VGG16 Model)*. Pada penelitian ini ada 2 skenario pengujian yang dilakukan, yaitu sebagai berikut :

1. Pengujian pertama menggunakan 280 objek motor sebagai data *training* dan dilakukan 37 *epoch*.
2. Pengujian kedua menggunakan 620 objek motor sebagai data *training* dan dilakukan 35 *epoch*.

4.3.2 Hasil penelitian

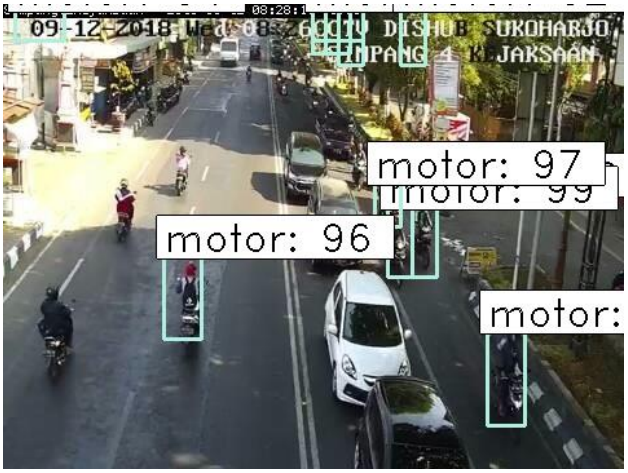
Hasil pengujian yang telah dilakukan berdasarkan scenario pengujian yang sudah dibuat sebelumnya dapat dilihat di gambar 4.5.



Gambar 4.7 Hasil Eksperimen

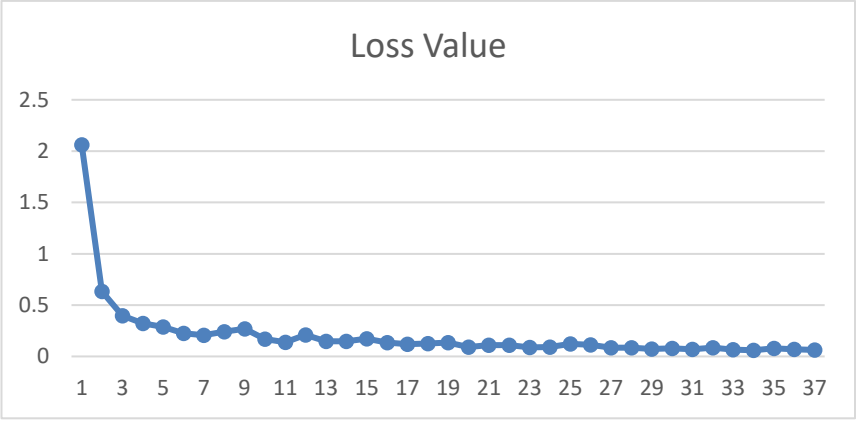
Pengujian pertama dilakukan sebanyak 37 *epoch* dengan menggunakan 284 objek motor. Dengan menggunakan skenario pertama, penulis menghasilkan nilai akurasi yang paling baik sebesar 0.9399 atau 93.9% dan menghasilkan *loss* sebesar 0.0601 atau 6.01%. Setelah penulis menguji dengan percobaan kedua yaitu dengan sebanyak 35 epoch dan 620 objek motor, penulis menghasilkan nilai akurasi yang paling baik sebesar 0.8789 atau 87.89% dan menghasilkan loss sebesar 0.1211 atau 12.11%. Terdapat perbedaan hasil deteksi antara percobaan pertama dan kedua yang bisa dilihat pada gambar 4.6 dan gambar 4.7. Penulis menghitung *average precision* dari percobaan pertama yang menghasilkan nilai *precision* 0.87 – 1.0 dan untuk percobaan kedua penulis menghasilkan nilai 0.62 – 1.0.

Gambar 4.8 Hasil Eksperimen Pertama

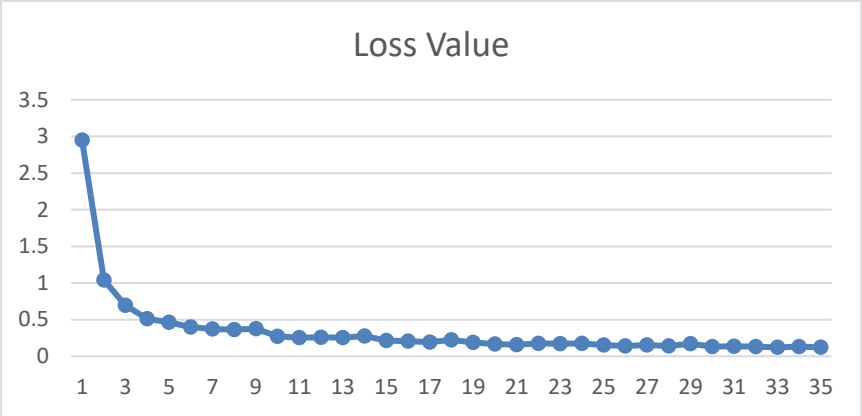


Gambar 4.9 Hasil Eksperimen Kedua

Pada pengujian pertama dan kedua, tidak selamanya epoch terakhir itu adalah akurasi yang terbaik. Maka dari itu penulis hanya menyimpan model baru jika nilai akurasi pada epoch tersebut lebih besar dari epoch sebelumnya, dan jika epoch memiliki nilai akurasi menurun, penulis tidak menyimpan model terbaru. Untuk grafik epoch pada percobaan pertama bisa dilihat pada gambar 4.8 dan untuk percobaan kedua dapat dilihat pada gambar 4.9.



Gambar 4.10 Grafik *Total Loss* Percobaan Pertama



Gambar 4.11 Grafik Total Loss Percobaan Kedua

Cara penulis mendapatkan hasil loss dari training yaitu dengan menjumlahkan Loss RPN Classifier + Loss RPN Regression + Loss Detector Classifier + Loss Detector Regression yang bisa dilihat keseluruhan hasil pada table 4.1 dan table 4.2.

Tabel 4.1 Tabel Hasil Percobaan Pertama

Epoch	Loss RPN Classifier	Loss RPN Regression	Loss Detector Classifier	Loss Detector Regression	Loss Value
1	1.4115	0.1285	0.2913	0.2323	2.0636
2	0.3683	0.0352	0.1435	0.0885	0.6355
3	0.2205	0.0206	0.0985	0.0571	0.3967
4	0.1771	0.0171	0.0816	0.0469	0.3227
5	0.1536	0.0178	0.0748	0.0419	0.2881
6	0.1248	0.0114	0.0557	0.0343	0.2262
7	0.1064	0.0125	0.0541	0.0323	0.2053
8	0.1453	0.0137	0.05	0.0315	0.2405
9	0.1727	0.0144	0.051	0.0302	0.2683
10	0.0965	0.0079	0.039	0.0261	0.1695
11	0.074	0.0081	0.0332	0.0235	0.1388
12	0.1327	0.0105	0.04	0.0256	0.2088
13	0.083	0.0098	0.0303	0.0232	0.1463
14	0.0911	0.0084	0.0272	0.0214	0.1481

Epoch	Loss RPN Classifier	Loss RPN Regression	Loss Detector Classifier	Loss Detector Regression	Loss Value
15	0.1103	0.0103	0.0306	0.022	0.1732
16	0.0814	0.0066	0.0267	0.0201	0.1348
17	0.0672	0.0078	0.0248	0.0202	0.12
18	0.0752	0.0075	0.0235	0.0192	0.1254
19	0.0771	0.0103	0.0266	0.0206	0.1346
20	0.0492	0.0051	0.021	0.0175	0.0928
21	0.0656	0.0071	0.0213	0.0172	0.1112
22	0.069	0.0059	0.0186	0.0162	0.1097
23	0.0527	0.004	0.016	0.0143	0.087
24	0.0464	0.0069	0.0211	0.0163	0.0907
25	0.0741	0.0083	0.022	0.0174	0.1218
26	0.0739	0.005	0.0178	0.0163	0.113
27	0.0487	0.0055	0.0157	0.0143	0.0842
28	0.055	0.0038	0.0134	0.0134	0.0856
29	0.0391	0.0056	0.0143	0.0147	0.0737
30	0.0361	0.0061	0.0209	0.0155	0.0786
31	0.0376	0.0037	0.0134	0.0135	0.0682
32	0.0493	0.0055	0.0151	0.0145	0.0844
33	0.0336	0.0047	0.0134	0.0141	0.0658
34	0.0332	0.0037	0.0103	0.0129	0.0601
35	0.0429	0.0052	0.0176	0.0139	0.0796
36	0.0423	0.0036	0.0117	0.0127	0.0703
37	0.0347	0.0038	0.0115	0.0127	0.0627

Tabel 4.2 Table Hasil Percobaan Kedua

Epoch	Loss RPN Classifier	Loss RPN Regression	Loss Detector Classifier	Loss Detector Regression	Loss Value
1	2.1941	0.1635	0.3279	0.2654	2.9509
2	0.5797	0.075	0.2367	0.1483	1.0397

Epoch	Loss RPN Classifier	Loss RPN Regression	Loss Detector Classifier	Loss Detector Regression	Loss Value
3	0.3757	0.0549	0.1659	0.0972	0.6937
4	0.2615	0.0435	0.1318	0.0741	0.5109
5	0.2509	0.0378	0.1093	0.0647	0.4627
6	0.2111	0.0335	0.0965	0.0575	0.3986
7	0.2007	0.0287	0.0916	0.0526	0.3736
8	0.2086	0.0258	0.0834	0.0473	0.3651
9	0.2341	0.023	0.075	0.0432	0.3753
10	0.1449	0.0209	0.0669	0.0397	0.2724
11	0.1405	0.0188	0.0607	0.0363	0.2563
12	0.1444	0.0186	0.0601	0.0362	0.2593
13	0.1538	0.016	0.0517	0.0333	0.2548
14	0.1738	0.0186	0.0518	0.0332	0.2774
15	0.1238	0.015	0.0462	0.0313	0.2163
16	0.1219	0.0135	0.0427	0.0293	0.2074
17	0.1133	0.0128	0.0387	0.0268	0.1916
18	0.1335	0.016	0.0429	0.0301	0.2225
19	0.1147	0.013	0.036	0.0264	0.1901
20	0.0906	0.011	0.0379	0.0259	0.1654
21	0.0933	0.0093	0.0328	0.0232	0.1586
22	0.1008	0.0131	0.0384	0.0253	0.1776
23	0.0958	0.0121	0.0375	0.0248	0.1702
24	0.105	0.0131	0.0336	0.0242	0.1759
25	0.0893	0.0106	0.0304	0.0225	0.1528
26	0.0856	0.0088	0.0265	0.0213	0.1422
27	0.0925	0.01	0.028	0.0216	0.1521
28	0.0917	0.008	0.0236	0.019	0.1423
29	0.1059	0.0117	0.0304	0.0211	0.1691
30	0.0788	0.0089	0.0245	0.0189	0.1311
31	0.0798	0.0108	0.0256	0.0193	0.1355

Epoch	Loss RPN Classifier	Loss RPN Regression	Loss Detector Classifier	Loss Detector Regression	Loss Value
32	0.0777	0.0096	0.0251	0.02	0.1324
33	0.0707	0.009	0.023	0.0184	0.1211
34	0.0866	0.0076	0.0192	0.0173	0.1307
35	0.0758	0.0075	0.0202	0.0176	0.1211

Untuk melihat perbandingan *average precision* pada percobaan pertama dan kedua dapat dilihat pada tabel 4.3.

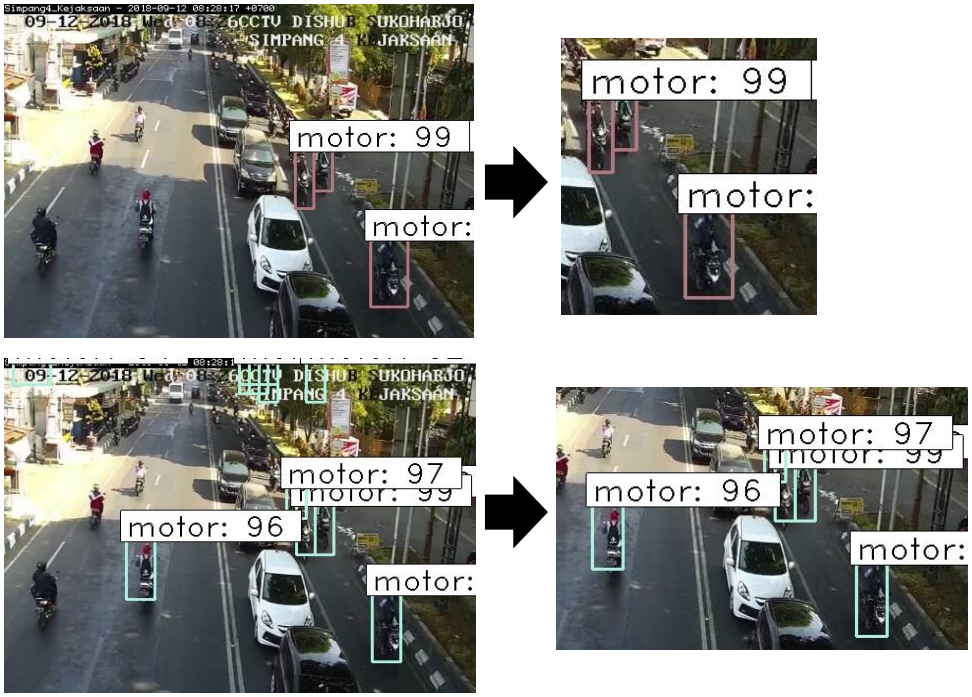
Tabel 4.3 Table perbandingan *average precision*

Gambar	Percobaan 1	Percobaan 2
0	1	1
1	1	1
2	0.967	0.650
3	0.864	0.626
4	0.863	0.666
5	0.878	0.744
6	0.883	0.727
7	0.880	0.717
8	0.873	0.653
9	0.888	0.660
10	0.899	0.672

4.3.3 Pembahasan hasil penelitian

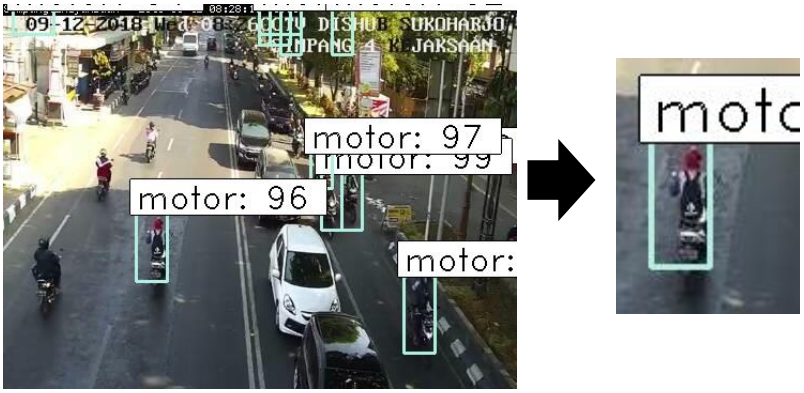
Pada pengujian pertama dan kedua, bisa dilihat pada gambar 4.11 bahwa terdapat perbedaan deteksi motor di kedua gambar test tersebut. Pada percobaan pertama terdeteksi 3 motor dan pada percobaan kedua terdeteksi 5 motor. Jika dilihat dengan mata penulis, objek yang dideteksi mendekati benar yaitu percobaan kedua. Itu disebabkan perbedaan jumlah data training pada percobaan kedua lebih banyak daripada percobaan pertama. Maka dari itu sistem akan bisa lebih mempelajari bentuk dan

jenis motor itu. Tetapi pada perhitungan presisi, percobaan pertama lebih baik dari percobaan kedua. Untuk percobaan pertama maupun percobaan tersebut memiliki hasil deteksi yang kurang dari percobaan kedua, tetapi hasil deteksi tersebut lebih akurat dibandingkan percobaan kedua.



Gambar 4.12 Kesalahan Sistem Mendeteksi Objek

Tetapi, percobaan kedua juga terdapat kesalahan pendeteksi sepeda motor yang bisa dilihat pada gambar 4.12. Di gambar tersebut, percobaan kedua mendeteksi motor yang berarah keatas. Pada bab 3 telah dijelaskan bahwa penulis hanya men-*training* motor yang arahnya kebawah. Tetapi pada percobaan kedua mendeteksi motor yang arahnya keatas. Tetapi, motor yang terdeteksi pada arah yang kebawah jauh lebih teliti dari percobaan pertama.



Gambar 4.13 Kesalahan Sistem Mendeteksi Objek

Jika dibandingkan pada percobaan pertama dan kedua, banyaknya data yang dipelajari sangat mempengaruhi pada fungsi loss dan hasil deteksi sepeda motor. Maupun nilai loss pada percobaan kedua jauh lebih besar dari percobaan pertama, percobaan kedua dapat lebih teliti dalam mendeteksi objek motor tetapi tidak menutup kemungkinan percobaan kedua tidak terdapat kesalahan. Bisa dilihat pada gambar 4.12 percobaan kedua menghasilkan kesalahan dalam mendeteksi objek. Maka dari itu, penulis beranggapan bahwa nilai loss tidak 100% dapat dipercaya untuk mendeteksi objek. Untuk analisis pendeteksian objek harus dilakukan lebih lanjut dengan melihat berapa banyak data yang dipelajari dan berapa banyak epoch yang dilakukan pada tahap training.

BAB V
PENUTUP

5.1 Kesimpulan

Cara mendeteksi motor dalam sebuah citra dapat menggunakan arsitektur *Faster R-CNN (Base VGG16)* yang telah berhasil diimplementasikan pada sistem untuk mendeteksi sepeda motor. Sistem pendeteksi motor menggunakan data yang diambil dari *CCTV Online* Dishub Kab. Sukoharjo yang memiliki resolusi 480 x 360 *pixel*. Tahapan dalam pembuatan sistem dimulai dengan pengambilan dan transformasi citra yaitu dengan mengambil objek motor pada video yang telah di-*screenshot*. Selanjutnya masuk ke tahap pra-proses, dimana tahap ini penulis hanya melakukan *resize* gambar menjadi 600 x 800 *pixel*. Setelah melakukan pra proses, tahap selanjutnya yaitu tahap *training* dan *testing* menggunakan *Faster R-CNN*. Pada proses *training* dengan menggunakan arsitektur *Faster R-CNN* yang digunakan dengan basis *VGG16*. Selanjutnya yaitu tahap pendeteksi / *testing* sepeda motor, pada tahap ini sistem hanya bisa mendeteksi suatu gambar, gambar tersebut akan diinput kedalam sistem.

Pada dua kali percobaan, percobaan pertama dapat hasil presisi yang lebih baik dari percobaan kedua, maka dari itu percobaan pertama bisa dikatakan objek yang dideteksi lebih akurat dibandingkan percobaan kedua dengan nilai presisi rata-rata 0.88 pada percobaan pertama.

5.2 Saran

Berikut saran penulis untuk penelitian selanjutnya dalam sistem pendeteksi motor:

1. Penulis berharap sistem ini dapat dilakukan pada arsitektur lain. Beberapa tulisan jurnal menyebutkan terdapat arsitektur yang lebih baik dengan layer yang lebih banyak dari arsitektur *Faster R-CNN*. Penulis berharap dapat dilakukan pada arsitektur seperti *YOLO V2*.
2. Menambahkan jumlah data yang digunakan pada tahap *training*. Dapat dilihat pada gambar 4.11, bahwa percobaan kedua dengan memiliki

lebih banyak data, kemungkinan sistem mendeteksi akan semakin akurat.

3. Setelah mendeteksi motor, penulis berharap sistem ini berkembang lebih lanjut misal terdapat pelanggaran didalamnya, misal pelanggaran orang yang tidak memakai helm.

DAFTAR PUSTAKA

Han,Chin-Chuan., dkk, (2001), *Personal Authentication using Palm-print Features*, Department of Computer Science and Information Engineering, Pergamon.

Zhu, Wentao, dkk, (2012), *Vehicle Detection in Driving Simulation Using Extreme Learning Machine*. Intitute of Computing Technology, China.

Bianco, Simone,. Marco, B., Davide, M., Raimondo, S., (2016). *Deep Learning for Logo Recognition*. University Degli Study. Milano.

Jiang, Haomiao, Qiyuan, T., Joyce, F., Brian, W. (2016). *Learning the Image Processing Pipeline*. Department of Electrical Engineering. Stanford University.

Han, Sue, dkk. (2016). *How Deep Learning Extracts and Learns Leaf Features for Plant Classification*. Centre of Image and Signal Processing. Malaysia.

Qi, Tangquan, Xu, Y., Quan, Y., Wang, Y., Ling, H. (2016). *Image-Based Action Recognition Using Hint-Enhanced Deep Neural Networks*. The School of Computer Science and Engineering. China

Young, Ian T., Jan, G., Lucas, V. (2007). *Fundamentals of Image Processing*. Delft University of Technology.

Liaw, Andy, Matthew, W. (2002). *Classification and Regression by randomForest*. Merck Research Laboratories.

Suzuki, Kenji, Luping, Z., Qian, W. (2016). *Machine Learning in Medical Imaging*. Medical Imaging Research Center. USA

Lee, Hyo-Haen, Kwang, H. (2015). *Automatic Recognition of Flower Species in the Naural Environment*. College of Information and Communication Engineering. South Korea

Kapugama, K., Lorensuhewa, Kalyani. (2016). *Enhancing Wikipedia Search Results Using Text Mining*. University of Ruhuna. Sri Lanka

Rose, Kenneth. (1998). *Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems*. Member IEEE

Agarwa, Pankay K. (2003). *Lecture 18: Clustering and Classification*. Algorithms in Computational Biology.

Lee, Honglak, Roger, G., Rajesh, R., Andrew, Y. (2009). *Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representasions*.

Stanford University. Stanford

Zhan, Shu, Qin, T., Xiao, Li. (2015). *Face Detection Using Representation Learning*. Hefei University of Technology. China.

Wang, Yi, Zhiming, L., Pierre, J. (2016). Interactive Deep Learning Method for Segmenting Moving Objects. University of Sherbrooke. Canada.

Rohrer, Brandon. (2016). *Deep Learning Demystified*. ODSC Boston Sommerville, I. (2011). *Software Engineering.* Addison-Wesley.

Roghganger, F., Warrender, C., Speed, A., dkk. (2011). *Connecting Cognitive and Neural Models*, BICA

Kriesel, David. (2005). *A Brief Introduction to Neural Networks*. University of Bonn. Germany

R. Rojas. (1996). *Neural Networks*. Springer. Berlin

Riza, L. S. (2015). *Data Science and Big Data Processing in R: Representations and Software.* Granada: Universidad de Granada.

Arthur, S. L. (1959). *Some Studies in Machine Learning Using the Game of Checkers.* IBM Journal of Research and Development.

Muhammad, A., Irawan, I. M., & Matu, E. (2015). *Study Comparison Of Svm-, K- Nn-And Backpropagation-Based Classifier*. Journal of Computer Science and Information , 1.

Johnson, R., & Wichern, D. (1982). *Applied multivariate statistical analysis.* New Jersey: Englewood Cliffs.

Deng, Zhipeng, Hao Sun, Shilin Zhou, Juanping Zhao, Lin Lei, Huanxin Zou. (2018). *Multi-scale Object Detection in Remote Sensing Imagery with Convolutional Neural Network.* Changsha: College of Electronic Science.

Song, Zhenzhen., dkk, (2019), Kiwifruit detection in field images using Faster R-CNN with VGG16, Northwest A&F University, China.

Chahyati, Dina, Mohamad Ivan Fanany, Aniati Murni Arymurthy. (2017). Tracking People by Detection Using CNN Features. Universitas Indonesia, Indonesia.