TUGAS AKHIR – TM184835

DETEKSI KOROSI PADA MATERIAL BAJA MENGGUNAKAN KAMERA *DRONE* DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*

BARA ATMAJA

NRP 02111840000132

Dosen Pembimbing

M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.

NIP. 198204142010121001

TEKNIK MESIN

Departemen Teknik Mesin

Fakultas Teknologi Industri dan Rekayasa Sistem

Institut Teknologi Sepuluh Nopember

Surabaya

2022

TUGAS AKHIR – TM184835

DETEKSI KOROSI PADA MATERIAL BAJA MENGGUNAKAN KAMERA DRONE DENGAN METODE CONVOLUTIONAL NEURAL NETWORK

BARA ATMAJA

NRP 02111840000132

Dosen Pembimbing

M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.

NIP. 198204142010121001

TEKNIK MESIN

Departemen Teknik Mesin

Fakultas Teknologi Industri dan Rekayasa Sistem

Institut Teknologi Sepuluh Nopember

Surabaya

2022

FINAL PROJECT – TM184835

CORROSION DETECTION ON STEEL MATERIAL USING DRONE CAMERA WITH CONVOLUTIONAL NEURAL NETWORK METHOD

BARA ATMAJA

NRP 02111840000132

Advisor

M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.

NIP. 198204142010121001

MECHANICAL ENGINEERING

Department of Mechanical Engineering

Faculty of Industrial Technology and Systems Engineering

Institut Teknologi Sepuluh Nopember

Surabaya

2022

LEMBAR PENGESAHAN

DETEKSI KOROSI PADA MATERIAL BAJA MENGGUNAKAN KAMERA *DRONE* DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Teknik pada
Program Studi S-1 Teknik Mesin
Departemen Teknik Mesin
Fakultas Teknologi Industri dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember

Oleh : **BARA ATMAJA**
NRP. 02111840000132

Disetujui oleh Tim Penguji Tugas Akhir :

1. M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D. — Pembimbing
2. Dinny Harnany, S.T., M.Sc. — Penguji
3. Arif Wahjudi, S.T., M.T., Ph.D. — Penguji
4. Alief Wikarta, S.T., MSc.Eng., Ph.D. — Penguji

SURABAYA
Juli, 2022

APPROVAL SHEET

CORROSION DETECTION ON STEEL MATERIAL USING DRONE CAMERA WITH CONVOLUTIONAL NEURAL NETWORK METHOD

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree Bachelor of Engineering at
Undergraduate Study Program of Mechanical Engineering
Department of Mechanical Engineering
Faculty of Industrial Technology and Systems Engineering
Institut Teknologi Sepuluh Nopember

By : **BARA ATMAJA**
NRP. 02111840000132

Approved by Final Project Examiner Team :

1. M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D. — Advisor
2. Dinny Harnany, S.T., M.Sc. — Examiner
3. Arif Wahjudi, S.T., M.T., Ph.D. — Examiner
4. Alief Wikarta, S.T., MSc.Eng., Ph.D. — Examiner

SURABAYA
July, 2022

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

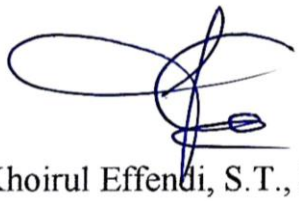
Nama mahasiswa / NRP	:	Bara Atmaja / 02111840000132
Departemen	:	Teknik Mesin
Dosen Pembimbing / NIP	:	M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D. / 198204142010121001

dengan ini menyatakan bahwa Tugas Akhir dengan judul "DETEKSI KOROSI PADA MATERIAL BAJA MENGGUNAKAN KAMERA *DRONE* DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 29 Juli 2022

Mengetahui
Dosen Pembimbing



M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.
NIP. 198204142010121001

Mahasiswa,

Bara Atmaja
NRP. 02111840000132

STATEMENT OF ORIGINALITY

The undersigned below:

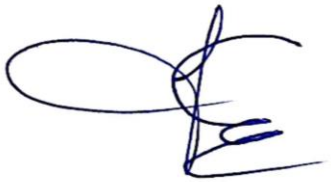
Name of student / NRP	:	Bara Atmaja / 02111840000132
Department	:	Mechanical Engineering
Advisor / NIP	:	M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D. / 198204142010121001

hereby declare that the Final Project with the title of "CORROSION DETECTION ON STEEL MATERIAL USING DRONE CAMERA WITH CONVOLUTIONAL NEURAL NETWORK METHOD" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, July 29th 2022

Acknowledged
Advisor



M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.
NIP. 198204142010121001

Student,

Bara Atmaja
NRP. 02111840000132

DETEKSI KOROSI PADA MATERIAL BAJA MENGGUNAKAN KAMERA *DRONE* DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*

Nama / NRP **: Bara Atmaja / 02111840000132**
Departemen **: Teknik Mesin FTIRS-ITS**
Dosen Pembimbing **: M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.**

ABSTRAK

Baja merupakan salah satu material logam yang paling banyak digunakan dalam dunia industri dan konstruksi. Kelebihan dalam penggunaan baja adalah gaya tarik dan tekan yang relatif tinggi dibanding material lain. Walaupun terdapat kekurangan dalam ketahanan terhadap korosi, material baja masih tetap digunakan pada banyak konstruksi di dunia. Korosi akan mempengaruhi daya tahan sebuah konstruksi dan dapat menyebabkan kerusakan yang menyebabkan kecelakaan. Hal tersebut dapat dicegah dengan melakukan inspeksi secara rutin. Selama ini inspeksi sebuah konstruksi dilakukan manual oleh tenaga manusia untuk mencapai area yang sulit diamati. Inspeksi secara manual ini dapat memakan waktu berjam-jam untuk mengidentifikasi masalah yang terjadi. Untuk meningkatkan efisiensi dan tingkat keamanan serta meminimalisir kesulitan dan risiko selama proses inspeksi, maka digunakanlah *drone* yang dilengkapi dengan kamera untuk pengambilan citra struktur konstruksi bermaterial baja yang terdapat korosi. Citra struktur material baja yang mengalami korosi kemudian akan diproses menggunakan *computer vision* yaitu *object detection* dengan metode *Convolutional Neural Network* (CNN). Pada penelitian ini dilakukan pendeteksian korosi pada Jembatan Petekan Surabaya dengan model pralatih SSD Mobilenet V1 yang tersedia secara *default* pada Tensorflow *Object Detection* API. Pendeteksian korosi dengan *drone* dilakukan pengujian dengan variasi jarak objek dengan *drone* sebesar 1 dan 2 m serta variasi kecepatan *drone* sebesar 0.6 m/s; 0.9 m/s; dan 1.3 m/s. Sedangkan untuk menentukan *setting* parameter CNN yang terbaik, dilakukan variasi nilai *batch size* dan *learning rate*. Penentuan nilai *batch size* dan *learning rate* sangat penting karena akan berdampak pada model serta hasil deteksi. Pada penelitian ini dilakukan variasi nilai *batch size* 4 dan 8 serta *learning rate* 0.001 dan 0.01. Hasil penelitian ini menunjukkan bahwa implementasi CNN dapat dilakukan untuk mendeteksi korosi pada material baja menggunakan *drone* dengan data set sebanyak 200 citra menggunakan variasi yang menghasilkan nilai akurasi paling tinggi didapat pada jarak 1 m dengan kecepatan 0.6 m/s serta dengan *setting* parameter *batch size* 8 dan *learning rate* 0.001 menggunakan iterasi 200000 *steps* dengan nilai akurasi sebesar 84.66% dan nilai total *loss* sebesar 1.673.

Kata kunci: baja, korosi, *drone*, *Convolutional Neural Network*, SSD Mobilenet V1

CORROSION DETECTION ON STEEL MATERIAL USING DRONE CAMERA WITH CONVOLUTIONAL NEURAL NETWORK METHOD

Student Name / NRP : Bara Atmaja / 02111840000132
Department : Mechanical Engineering FTIRS-ITS
Advisor : M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.

ABSTRACT

Steel is one of the most widely used metal materials in the world of industry and construction. The advantage in the use of steel is that the tensile and compressive forces are relatively high compared to other materials. Although there are shortcomings in corrosion resistance, steel materials are still used in many constructions in the world. Corrosion will affect the durability of a construction and can cause damage that causes accidents. This can be prevented by conducting regular inspections. During this time the inspection of a construction is carried out manually by manpower to reach areas that are difficult to observe. These manual inspections can take hours to identify the problem that occurs. To increase efficiency and safety levels and minimize difficulties and risks during the inspection process, drones equipped with cameras are used to capture images of steel-material construction structures that have corrosion. The image of the steel material structure that is subject to corrosion will then be processed using computer vision, namely object detection using the Convolutional Neural Network (CNN) method. In this study, corrosion detection was carried out on the Petekan Bridge Surabaya with a pretrained model of the Mobilenet V1 SSD which is available by default in the Tensorflow Object Detection API. Corrosion detection with drones was tested with variations in the distance of objects with drones of 1 m and 2 m and variations in drone speed of 0.6 m / s; 0.9 m / s; and 1.3 m / s. Meanwhile, to determine the best CNN parameter settings, variations were made in the form of batch size and learning rate values . Determining the value of the batch size and learning rate is very important because it will have an impact on the model as well as the detection results. In this study, variations in the values of batch sizes 4 and 8 were carried out as well as learning rate values of 0.001 and 0.01. The results of this study show that the implementation of CNN can be carried out to detect corrosion in steel materials using drones with a data set of 200 images using variation which produces the highest accuracy value found at a distance of 1 meter at a speed of 0.6 m / s and with a parameter setting batch size 8 and learning rate 0.001 using iteration 200000 steps with an accuracy value of 84.66% and a total loss value of 1,673.

Keywords: ***steel, corrosion, drone, Convolutional Neural Network, SSD Mobilenet V1***

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas segala nikmat dan karunia-Nya, sehingga Tugas Akhir berjudul "DETEKSI KOROSI PADA MATERIAL BAJA MENGGUNAKAN KAMERA *DRONE* DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*" ini dapat selesai sesuai dengan waktu yang telah ditentukan.

Pengerjaan Tugas Akhir ini menjadi sebuah sarana untuk penulis memperdalam ilmu yang telah didapatkan di Institut Teknologi Sepuluh Nopember (ITS) Surabaya, khususnya dalam disiplin ilmu Teknik Mesin. Terselesaikannya buku Tugas Akhir ini tidak terlepas dari bantuan dan dukungan semua pihak. Pada kesempatan kali ini penulis ingin mengucapkan terima kasih kepada:

1. Kedua orang tua, Bapak Bambang Sihna, S.E., dan Ibu Dra. Rr. Ratna Wiwara Nugrahaningsih, M.M., adik Hendra Kusuma, dan keluarga penulis yang senantiasa mendukung dan mendoakan penulis dalam perjalanan panjangnya.
2. M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D. selaku dosen pembimbing penulis yang telah meluangkan waktu berharganya demi memberikan ide, bimbingan, dan evaluasi yang sangat berdampak besar dalam penyelesaian Tugas Akhir ini.
3. Dinny Harnany, S.T., M.Sc., Arif Wahjudi, S.T., M.T., Ph.D., dan Alief Wikarta, S.T., M.Sc.Eng., Ph.D. selaku dosen penguji yang telah memberikan saran, masukan, dan arahan untuk menyempurnakan Tugas Akhir ini.
4. Alief Wikarta, S.T., M.Sc.Eng., Ph.D. selaku dosen wali yang telah membimbing penulis dalam menjalani proses perkuliahan di Departemen Teknik Mesin ITS.
5. Mohammad Anas Tawakkal selaku pemilik *drone* yang telah meminjamkan *drone* DJI Mavic Air 2.
6. Seluruh dosen Departemen Teknik Mesin FTIRS ITS.
7. Sobad Gayam yang telah menghambat penulisan Tugas Akhir.
8. Teman-teman di Laboratorium Rekayasa Produksi.
9. Teman-teman sarjana Departemen Teknik Mesin FTIRS ITS serta semua pihak yang telah banyak membantu dalam menyelesaikan Tugas Akhir ini yang tidak dapat penulis sebutkan satu-persatu.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak sekali kekurangan namun besar harapan penulis semoga Tugas Akhir ini dapat membantu serta memberikan pengetahuan yang berguna bagi para pembaca.. Dengan kerendahan hati, penulis memohon maaf sebesar-besarnya atas kekurangan tersebut dan sangat terbuka akan kritik serta saran.

Surabaya, 28 Juli 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Logam merupakan salah satu jenis material yang sering dimanfaatkan dalam bidang industri maupun konstruksi karena memiliki ketahanan yang baik terhadap lingkungan. Salah satu material logam yang banyak digunakan dalam dunia industri dan konstruksi adalah baja. Baja merupakan logam paduan dengan besi sebagai unsur dasar dan karbon sebagai paduan utamanya. Kelebihan dalam penggunaan baja adalah gaya tarik dan tekan yang relatif tinggi dibanding material lain, serta kemudahannya dalam proses produksi serta pemasangannya di segala lingkungan. Walaupun terdapat kekurangan dalam ketahanan terhadap korosi, material baja masih tetap digunakan pada banyak konstruksi di dunia (Aini, 2016).

Korosi adalah proses kerusakan material akibat reaksi kimia atau elektrokimia dengan lingkungannya yang dihadapi oleh setiap negara di dunia karena diperkirakan 1 – 5% dari Produk Nasional Bruto (PNB) habis akibat korosi. Pada tahun 1998 *The National Association of Corrosion Engineers* (NACE) mencatat bahwa total biaya yang dikeluarkan Amerika Serikat setiap tahun akibat korosi mencapai $276 miliar atau sebesar 3.1% dari PNB Amerika Serikat (Verma, 2017). Korosi menghasilkan degradasi yang akan mempengaruhi daya tahan pada sebuah konstruksi serta sistem industri, sehingga dapat menyebabkan kecelakaan yang dapat mengakibatkan kerugian ekonomi yang besar serta hilangnya nyawa.

Kasus kecelakaan akibat korosi pernah terjadi di Jembatan Morandi, Genoa, Italia. Pada tanggal 14 Agustus 2018, sepanjang 100 meter bagian dari jembatan layang di Genoa ambruk seperti terlihat pada Gambar 1.1, sehingga mengakibatkan 39 orang meninggal dunia. Para ahli teknik mengungkapkan bahwa terdapat korosi pada kabel-kabel logam jembatan tersebut, yang mengurangi kekuatan jembatan sebesar 20% (Tempo, 2018).

Gambar 1.1 Jembatan Morandi yang Runtuh di Kota Genoa, Italia
(Tempo, 2018)

Tanpa perawatan yang baik, daya tahan sebuah konstruksi yang menggunakan material baja dapat berpotensi menurun. Oleh karena itu diperlukan proses inspeksi yang dilakukan secara rutin sebagai proses pemeliharaan untuk meyakinkan bahwa konstruksi masih aman dan berfungsi dengan baik. Selama ini inspeksi sebuah konstruksi dilakukan manual oleh tenaga manusia menggunakan alat bantu yang membantu pekerja tersebut untuk mencapai aera yang sulit diamati seperti pada bagian bawah jembatan atau *stack* yang terdapat pada sebuah pembangkit listrik. Selain memakan waktu berjam-jam untuk mengidentifikasi masalah yang terjadi, inspeksi manual juga dapat menimbulkan risiko kecelakaan. Risiko kecelakaan yang mungkin terjadi misalnya terjatuh, tertimpa benda jatuh, terganggunya pernafasan akibat debu,

terkena arus listrik, temperatur lingkungan yang ekstrim, serta suara bising yang dapat menurunkan daya dengar (Akbari, 2021).

Risiko kecelakaan serta keterbatasan kemampuan manusia, seperti menjangkau seluruh area, lama waktu yang dibutuhkan, dan banyaknya tenaga kerja yang dikerahkan merupakan salah satu penyebab digunakannya *drone*. *Drone* merupakan salah satu jenis pesawat tanpa awak atau *Unmanned Aerial Vehicle* (UAV) yang dikendalikan dari jarak jauh oleh *flight controller*. *Drone* digunakan untuk meningkatkan efisiensi dan tingkat keamanan serta meminimalisir kesulitan dan risiko selama proses inspeksi. Oleh karena itu *drone* dilengkapi dengan kamera untuk pengambilan citra struktur konstruksi yang kemudian rekaman tersebut akan diproses menggunakan *computer vision*. *Computer vision* sendiri merupakan proses pengolahan citra oleh komputer untuk membaca informasi dari citra yang ditangkap melalui kamera. Terdapat beberapa penerapan pada *computer vision*, salah satunya adalah *object detection*. *Object detection* terinspirasi dari kemampuan manusia untuk melihat dan memahami objek yang terlihat oleh indra pengelihatannya. Salah satu metode yang banyak digunakan dalam *object detection* adalah *Convolutional Neural Network* (CNN).

Convolutional Neural Network (CNN) termasuk bagian dari *deep learning* yang berkembang saat ini karena kedalaman jaringannya. CNN merupakan salah satu metode yang banyak digunakan dalam melakukan ekstraksi citra yang kemudian menghasilkan informasi yang selanjutnya dilakukan pemrosesan infomasi, salah satunya adalah *object detection*. CNN dipilih karena banyak digunakan pada banyak penelitian karena tingkat akurasinya yang relatif tinggi dan memiliki hasil yang signifikan dalam proses pengolahan citra (Dewi, 2018).

Uraian di atas mendasari penelitian ini dalam menerapkan *Convolutional Neural Network* (CNN) dalam pembuatan model untuk mendeteksi korosi pada material baja dengan bantuan *drone* yang mana kedepannya pendeteksian korosi dapat dikembangkan dan diterapkan untuk meningkatkan efektifitas serta kinerja dalam melakukan suatu inspeksi.

1.2 Rumusan Masalah Penelitian

Berdasarkan latar belakang yang telah dijelaskan, maka perumusan masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimana implementasi metode *Convolutional Neural Network* (CNN) untuk mendeteksi lokasi korosi pada material baja dengan *drone*?
2. Berapa variasi jarak dan kecepatan *drone* sehingga menghasilkan nilai akurasi terbaik dalam pendeteksian korosi pada material baja?
3. Bagaimana menentukan *setting* parameter *Convolutional Neural Network* (CNN) yang paling baik, sehingga dapat mendeteksi korosi dengan nilai akurasi yang tinggi?

1.3 Tujuan Penelitian

Berdasarkan perumusan masalah yang sudah ditetapkan, maka penelitian ini dilakukan dengan tujuan sebagai berikut:

1. Mengetahui implementasi metode *Convolutional Neural Network* (CNN) untuk mendeteksi lokasi korosi pada material baja dengan *drone*.
2. Mengetahui variasi jarak dan kecepatan *drone* sehingga menghasilkan nilai akurasi terbaik dalam pendeteksian korosi pada material baja.
3. Mengetahui *setting* parameter *Convolutional Neural Network* (CNN) yang paling baik, sehingga dapat mendeteksi korosi dengan nilai akurasi yang tinggi.

1.4 Batasan Masalah Penelitian

Agar dalam proses penyelesaian penelitian ini dapat berjalan dengan fokus dan terarah, maka diberlakukan batasan masalah sebagai berikut:

1. Jenis korosi yang dideteksi adalah *uniform corrosion*.
2. Lokasi pengambilan data *input* video adalah Jembatan Petekan Surabaya dengan waktu pengambilan dilakukan pada pukul 09.00 – 10.00 WIB.
3. Struktur jembatan akan direkam menggunakan *drone* DJI Mavic Air 2.
4. Variasi yang digunakan pada saat pengambilan *input* video berupa variasi jarak *drone* dengan objek 1 dan 2 m serta kecepatan *drone* 0.6 m/s; 0.9 m/s; dan 1.3 m/s.
5. Data yang digunakan sebagai *training* adalah citra korosi yang diunduh dari Google Image dan dokumen pribadi yang didapat di lingkungan Institut Teknologi Sepuluh Nopember (ITS).
6. Digunakan model pralatih SSD Mobilenet V1 sebagai arsitektur *Convolutional Neural Network* (CNN).
7. Bahasa pemrograman yang digunakan adalah *python* 3.7.
8. Variasi *setting hyperparameter* yang digunakan adalah *batch size* 4 dan 8 serta *learning rate* 0.001 dan 0.01.

1.5 Manfaat Penelitian

Adapun manfaat yang dapat diperoleh dari penelitian ini adalah memberikan pengetahuan mengenai implementasi *deep learning* menggunakan *Convolutional Neutral Network* (CNN) dengan bantuan *drone* untuk melakukan inspeksi pada material baja yang mengalami korosi yang mana sebelumnya dilakukan secara manual. Lokasi yang sulit untuk dideteksi memungkinkan tingkat efektivitas dan kinerja penggunaan *drone* lebih baik jika dibandingkan dengan inspeksi secara manual.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian yang berjudul "*Corrosion Detection with Computer Vision and Deep Learning*" yang ditulis oleh Achilleas Matthaiou mencari solusi untuk proses deteksi korosi otomatis yang berfokus pada atribut visual korosi. Metode *artificial intelligence* digunakan untuk mengekstrak informasi dari citra. Permukaan yang terkorosi memiliki dua atribut warna dan tekstur yang diidentifikasi secara visual. Untuk mendeteksi korosi berdasarkan warna, algoritma pelacakan warna dibuat dan diuji menggunakan citra dari kompartemen kapal yang berbeda. Untuk mendeteksi korosi berdasarkan tekstur, algoritma *deep learning* digunakan, dan dua pendekatan diuji. Pendekatan pertama adalah model klasifikasi biner yang dilatih menggunakan arsitektur *Convolutional Neural Network* (CNN) dengan menggunakan *transfer learning*. Model ini juga digunakan oleh algoritma *sliding* untuk memungkinkan deteksi dan lokalisasi pada pelat besar yang terkorosi. Pendekatan kedua memperlakukan deteksi korosi sebagai masalah deteksi objek. *Single Shot Detector* (SSD) dilatih menggunakan *transfer learning* untuk mendeteksi korosi pada citra dunia nyata. Untuk mendukung pelatihan dan pengujian semua model, dua data set dibuat. Data set pertama terdiri dari citra logam terkorosi di lingkungan laboratorium, sedangkan dataset kedua dari citra dunia nyata dari kompartemen terkorosi dari inspeksi kapal curah. Penelitian ini menemukan bahwa ketiga metode tersebut mampu melakukan deteksi korosi dengan pendekatan *deep learning* yang menghasilkan hasil yang lebih baik. Tingkat akurasi dari penelitian ini mencapai 85% (Matthaiou, 2021).

Penelitian yang berjudul "*Corrosion Identification of Fittings Based on Computer Vision*" ini bertujuan untuk mengidentifikasi korosi menggunakan metode Faster-RCNN. Lingkungan korosi logam sangat kompleks, dan bagian korosi serta bentuknya sangat berbeda, membuat korosi sulit dideteksi. Karena *drone* secara bertahap diterapkan pada inspeksi jalur, *computer vision* dapat digunakan untuk mengidentifikasi korosi logam. Bertujuan pada masalah yang ada dalam deteksi korosi saat ini, penelitian ini mengusulkan algoritma deteksi korosi berdasarkan model deteksi target Faster-RCNN dan fitur warna korosi HSI, yang digunakan untuk memecahkan masalah penerapan yang buruk dan inefisiensi pemrosesan citra digital dan fitur tidak dapat diekstraksi secara akurat saat menggunakan metode *deep learning* dan masalah lainnya. Pertama, citra korosi dikonversi dari model warna RGB ke model warna HSI, dan kemudian setiap piksel dari ruang HSI dilintasi. Kemudian, pelabelan secara manual ke dalam set pelatihan melalui alat anotasi *open source* LabelImg. Kumpulan data berlabel memfasilitasi ekstraksi fitur oleh *Convolutional Neural Network* karena hanya area berkorosi yang ada. Deteksi korosi dan lokalisasi dilakukan pada set pelatihan yang disiapkan menggunakan model deteksi target Faster-RCNN. Hasilnya menunjukkan bahwa Faster-RCNN memiliki efek pengenalan yang baik untuk beberapa kondisi korosi umum. Selain itu, metode menggabungkan algoritma *deep learning* dengan fitur warna HSI mencapai tingkat tinggi dalam menentukan tingkat kebenaran dan penarikan korosi, dan tingkat pengenalan kebocoran juga memenuhi persyaratan praktis. Hasilnya menunjukkan bahwa Faster-RCNN memiliki efek pengenalan yang baik untuk beberapa kondisi korosi umum, dengan tingkat keakuratannya mencapai 76,84% (Tian & Zhang, 2020).

Penelitian yang berjudul "Pengaplikasian *Convolutional Neural Network* sebagai Metode Pendeteksi Korosi pada Struktur Kapal" yang ditulis oleh Nurdiansyah, dibuat dengan tujuan untuk mengurangi risiko terhadap keselamatan *surveyor* dan meningkatkan objektivitas dalam pelaksanaan inspeksi. Pada penelitian ini, dirancang sistem berbasis *machine learning* untuk membantu proses inspeksi sebuah struktur kapal. Pada penelitian ini digunakan metode pendeteksian objek dengan menggunakan model pralatih SSD MobileNet V1 untuk mendeteksi

3 tipe korosi yaitu korosi seragam (*uniform corrosion*), korosi sumuran (*pitting corrosion*), dan korosi tepi (*edge corrosion*). *Software* kemudian diintegrasikan dengan Raspberry Pi sehingga proses inspeksi dapat dilakukan dengan mobilitas yang tinggi. Dengan data set sebesar 138 citra yang didapat dari internet dan dokumen pribadi pada plat kapal, penelitian ini mendapatkan tingkat akurasi dari pendeteksian korosi sebesar 63% dengan nilai total *loss* sebesar 0.440 (Nurdiansyah, 2021).

2.2 Baja

Baja pada dasarnya adalah paduan antara besi dan karbon dengan kandungan karbon maksimum 1.7%. Selain terdiri dari besi dan karbon, baja juga mengandung unsur-unsur lain yang berasal dari pengotor bijih besi (misalnya belerang dan fosfor), biasanya kadarnya harus dijaga serendah mungkin. Sebagian lagi unsur yang digunakan pada proses produksi besi/baja (misalnya silikon dan mangan). Selain itu, biasanya beberapa unsur-unsur paduan ditambahkan untuk mencapai sifat baja tertentu, sehingga jenis baja akan beragam. Berdasarkan komposisinya, baja dapat dikelompokkan menjadi beberapa kelompok, yaitu baja karbon rendah, baja karbon menengah, dan baja karbon tinggi (Zakharov, 1962).

Baja karbon rendah (*low carbon steel*) mengandung karbon antara 0.10% – 0.30%. Baja karbon rendah biasanya digunakan sebagai baja konstruksi umum untuk baja profil rangka sebuah bangunan, baja tulangan beton, rangka kendaraan, mur, baut, dan lain-lain. Baja karbon menengah (*medium carbon steel*) yang mempunyai kandungan karbon sebesar 0.30% – 0.70%. Baja karbon menengah banyak digunakan untuk konstruksi mesin, seperti poros, *crankshaft*, batang piston, *gear*, pegas, dll. Baja karbon tinggi (*high carbon steel*) mempunyai kadar karbon di atas 0.70%. Baja ini bersifat lebih kuat dan keras, tetapi keuletan dan ketangguhannya rendah. Baja jenis ini digunakan untuk konstruksi mesin yang membutuhkan kekuatan dan ketangguhan yang tinggi, misalnya untuk gunting, mata bor, dll (Zakharov, 1962).

2.3 Korosi

Korosi merupakan serangan yang destruktif dan tidak disengaja pada suatu logam, yang mana adalah reaksi elektrokimia yang biasanya dimulai dari permukaan. Untuk material logam, proses korosi biasanya bersifat elektrokimia, yaitu reaksi kimia di mana elektron ditransfer dari satu jenis bahan kimia ke bahan kimia yang lain. Variabel dalam lingkungan korosi, yang meliputi kecepatan fluida, suhu, dan komposisi, dapat memiliki pengaruh yang menentukan pada sifat korosi dari bahan yang bersentuhan dengannya. Dalam kebanyakan kasus, peningkatan kecepatan fluida meningkatkan laju korosi karena efek erosif. Beberapa metode pencegahan korosi meliputi pelapisan (*coating*), perlakuan lingkungan, pemilihan material, desain berlebih, dan proteksi katodik (William D Callister & Rethwisch, 2019).

Klasifikasi korosi dapat ditinjau dari caranya terbentuk dan dapat dikenali dengan pengamatan visual. Dalam banyak kasus, pengamatan visual sudah cukup, tetapi dalam beberapa kasus tertentu diperlukan pengamatan dengan alat bantu. Jenis korosi yang dapat diidentifikasi dengan inspeksi visual meliputi *uniform corrosion* (korosi seragam), *pitting corrosion* (korosi sumuran), *crevice corrosion* (korosi celah), dan *galvanic corrosion* (korosi galvanis). Jenis korosi ini ditetapkan berdasarkan perbedaan dari karakteristik dan penyebabnya (Fontana, 1986).

2.3.1 *Uniform Corrosion*

Uniform corrosion (korosi seragam) adalah bentuk korosi paling umum yang terjadi secara bersama-sama pada seluruh permukaan logam akibat reaksi kimia atau elektrokimia, sehingga logam yang terkorosi seragam akan mengalami pengurangan ukuran yang relatif besar per satuan waktu. Korosi seragam atau korosi keseluruhan secara umum mewakili penghancuran logam terbesar, kerugian langsung akibat korosi seragam berupa hilangnya

material konstruksi, keselamatan kerja, dan pencemaran lingkungan akibat produk korosi berupa senyawa-senyawa yang mencemari lingkungan. Pada saat yang sama, kerugian tidak langsung meliputi penurunan kapasitas dan peningkatan biaya *maintenance*. Contoh dari *Uniform corrosion* (korosi seragam) dapat dilihat pada Gambar 2.1 (Fontana, 1986).

Gambar 2.1 *Uniform Corrosion*
(Nurdiansyah, 2021)

2.4 UAV (*Unmanned Aerial Vehicle*)

UAV adalah pesawat yang diterbangkan tanpa awak. Pesawat ini dibagi menjadi beberapa tipe berdasarkan pengendalinya yaitu, *remotely controlled*, *semi-autonomous*, *autonomous*, dan kombinasinya. Teknologi untuk *aerial surveying* atau fotogrametri adalah penggunaan pesawat tanpa awak ini sebagai alat pengukurnya. UAV saat ini umumnya sudah dilengkapi *tracking* untuk posisi dan orientasi sensor dalam sistem koordinat lokal maupun global (Eisenbeiss, 2009).

Keuntungan menggunakan UAV dibandingkan dengan pesawat berawak adalah dalam situasi kritis tidak melukai orang atau merusak area yang sedang dikerjakan. Pemetaan menggunakan UAV dapat digunakan dalam area-area beresiko seperti pegunungan, gunung berapi, banjir, dan area lainnya yang sulit dijangkau oleh manusia. Akan tetapi kelemahan dari penggunaan UAV ini adalah bergantung pada cuaca dan pencahayaan pada saat terbang. UAV juga tidak dapat melakukan pengambilan data dengan ketinggian yang sangat tinggi (Eisenbeiss, 2009).

2.4.1 Klasifikasi UAV

UAV dapat diklasifikasikan menurut karakteristik utama pesawat, yaitu dengan tenaga atau tanpa tenaga, lebih ringan atau lebih berat dari udara, dan sayap tetap atau *rotary*. menunjukkan klasifikasi dari UAV (Eisenbeiss, 2009).

Tabel 2.1 Klasifikasi UAV Sesuai Tenaga
(Eisenbeiss, 2009)

<table>
<tr><th rowspan="2">UAV</th><th rowspan="2">Lighter than Air</th><th colspan="3">Heavier than Air</th></tr>
<tr><th>Flexible Wing</th><th>Fixed Wing</th><th>Rotary Wing</th></tr>
<tr><td rowspan="3">Powered</td><td rowspan="3">Balloon</td><td>Hang Glider</td><td rowspan="3">Gliders</td><td rowspan="3">Rotor-Kite</td></tr>
<tr><td>Paraglider</td></tr>
<tr><td>Kites</td></tr>
<tr><td rowspan="4">Unpowered</td><td rowspan="4">Airship</td><td rowspan="4">Paraglider</td><td rowspan="2">Propeller</td><td>Single Rotors</td></tr>
<tr><td>Coaxial</td></tr>
<tr><td rowspan="2">JetEngines</td><td>Quadrotors</td></tr>
<tr><td>MultiRotors</td></tr>
</table>

UAV *rotary wing*, yang biasa disebut sebagai *Vertical Takeoff and Landing Vehicles* (VTOL), dapat diklasifikasikan menjadi, *single-*, *double-*, *four-* dan *multi* sistem rotor. Rotor yaitu penggerak, penggerak tunggal dengan satu baling–baling dan satu di ekor pesawat. Penggerak utama digunakan untuk mengangkat dan mendorong dan di bagian ekor digunakan untuk menstabilkan gerakan yaw (gerakan putar terhadap sumbu vertikal) dan torsi. Penggerak ganda berbeda dari penggerak tunggal karena meningkatkan beban dan dapat dioperasikan pada ketinggian yang tinggi dengan tenaga mesin. Penggerak ganda juga mudah dikendalikan dan kebisingannya rendah. Sistem pengendali empat rotor mempunyai kapasitas berat rendah sehingga menggunakan sensor yang ringan dengan sistem *low-cost*. Dengan ukurannya yang kecil dan kemampuan maneuver yang baik, sistem ini dapat terbang baik di dalam maupun di luar ruangan (Eisenbeiss, 2009).

2.4.2 *Drone* Sistem *Low-Cost*

Drone low-cost merupakan alternatif untuk UAV *rotary* dan *fixed wing*. Beberapa UAV menggunakan sensor yang berbeda, bahkan dapat diganti sesuai kebutuhannya. Penglihatan dari UAV dapat dilihat oleh *flight controller* melalui *smartphone* dan laptop. Pengambilan data dilakukan melalui remot kontrol dan lokasinya dapat dipantau menggunakan sistem GPS *low-cost*. Sistem ini terbatas untuk terbang dengan kecepatan angin kurang dari 6 m/s dengan ketinggian maksimum 4500 m dan jarak operator dengan UAV hingga 5 km (Eisenbeiss, 2009).

2.4.3 DJI Mavic Air 2

DJI merupakai perusahaan yang berfokus pada piranti multimedia seperti UAV *drone, action cam,* dan *stabilizer*. Mavic Air 2 merupakan jenis *drone* yang di desain agar lebih kuat dan praktis untuk dibawa dan untuk digunakan. Mavic Air 2 adalah *drone* yang menampilkan peningkatan kemampuan kamera, transmisi, dan kinerja penerbangan dibandingkan dengan versi sebelumnya. Mavic Air 2 dilengkapi pengendalian penerbangan, sistem penelihatan, peninderaan inframerah, dan baterai penerbangan cerdas. Mavic Air 2 seperti pada Gambar 2.2 memiliki tiga mode penerbangan, yaitu *normal*, *sport*, dan *tripod*. Data penerbangan secara otomatis disimpan ke perekam data internal *drone* yang dapat diakses melalui DJI Assistant 2 untuk Mavic (DJI, 2020).

Gambar 2.2 *Drone* DJI Mavic Air 2
(DJI, 2020)

2.5 Kamera

Kamera adalah alat yang sangat berpengaruh dalam bidang *aerial surveying* atau fotogrametri karena kamera merupakan alat utama untuk menangkap citra atau gambar. Alat fotografi dapat diklasifikasikan berdasarkan cara gambar tersebut dihasilkan. Kebutuhan akan kamera udara berbeda dengan kamera saku digital. Kamera yang dibutuhkan untuk

pengambilan gambar dalam udara adalah kamera dengan lensa dengan kualitas geometris yang tinggi. Kamera udara harus mampu mengambil citra dengan objek yang bergerak. Kamera udara juga harus memiliki kecepatan waktu pengambilan citra, lensa cepat, dan *shutter* yang cepat. Kamera diklasifikasikan menjadi dua kategori umum yaitu kamera metrik dan non metrik (Akbar, 2018).

2.5.1 Kamera Metrik

Kamera metrik adalah kamera yang dibuat khusus untuk tujuan fotogrametrik. Kamera metrik yang biasa digunakan memiliki ukuran format 23 x 23 cm. Kamera metrik seperti pada Gambar 2.3 didesain stabil dan sebelum dipakai dikalibrasi secara menyeluruh. Kalibrasi kamera adalah proses untuk menentukan parameter *input*, seperti panjang fokus dan koordinat titik foto (Akbar, 2018).

Gambar 2.3 Kamera PhaseOne iXU-RS1000
(Akbar, 2018)

2.5.2 Kamera Nonmetrik

Gambar 2.4 Kamera Mavic Air 2
(DJI, 2020)

Kamera nometrik dibuat untuk mengambil gambar di mana kualitasnya citra lebih diutamakan daripada kualitas geometriknya. Kamera nonmetrik seperti pada Gambar 2.4 memiliki beberapa keterbatasan pada ketidakstabilan geometrik dan ukuran film (Akbar, 2018).

1. **Ketidakstabilan Geometrik**

   Ketidakstabilan geometrik merupakan salah satu masalah paling besar dalam penggunaan kamera nonmetrik. Lensa yang tidak sempurna merupakan salah satu kelemahan dari kamera nonmeterik, yang menyebabkan pengambilan citra udara dapat mengalami kesalahan. Untuk mengatasi hal tersebut diperlukan kalibrasi dengan teknik tertentu sehingga parameter internal dapat diketahui dan dapat digunakan dengan baik.

2. **Ukuran Film**

Terbatasnya ukuran film atau sensor merupakan salah satu keterbatasan dalam penggunaan kamera nonmetrik. Penggunaan kamera nonmetrik membutuhkan lebih banyak foto dibandingkan dengan kamera metrik jika diambil dengan skala dan ukuran yang sama. Pengambilan citra udara dengan jarak yang sangat jauh juga membutuhkan kualitas citra yang bagus dengan ukuran asli yang besar.

2.6 Sensor

Saat mengandalkan kamera digital untuk merekam gambar harus memperhatikan sensor digital yang digunakan. Terdapat dua jenis sensor digital yang sering digunakan, yaitu CMOS dan CCD. Sensor CCD (*Charged Coupled Device*) adalah sensor bertipe analog yang sudah lama digunakan sebagai sensor kamera digital. Prinsip kerja dari sensor CCD sangat sederhana, yaitu dengan cara mengubah intensitas cahaya menjadi nilai tegangan. Kemudian akan diproses menjadi data digital oleh *Analog to Digital Converter* (ADC) di kamera digital (Akbar, 2018).

Sensor CMOS (*Complimentary Metal Oxide Semiconductor*) adalah sensor berteknologi modern dengan transistor di setiap pikselnya. Sensor CMOS dirancang dengan konsep *chip digital* sehingga *output* dari sensor ini sudah berbentuk data digital. Sehingga, kamera dengan sensor CMOS tidak lagi membutuhkan rangkaian ADC, karena *output* dari sensor CMOS dapat secara langsung masuk ke prosesor kamera. Sensor CMOS banyak digunakan pada kamera *handphone* meskipun dengan kualitas gambarnya yang tidak bagus karena mekanisme kerja sensor CMOS yang sangat sederhana (Akbar, 2018).

2.7 *Computer Vision*

Computer vision adalah pembelajaran untuk menganalisis citra maupun video untuk mendapatkan hasil sebagaimana yang dapat dilakukan manusia. Sebenarnya, *computer vision* mencoba menirukan dan menganalisis sistem pengelihatan manusia. Manusia melihat sebuah objek dengan indra penglihatannya, kemudian citra objek diteruskan ke otak untuk diinterpretasikan, sehingga manusia dapat memahami objek yang tampak di pandangannya. Hasil interpretasi ini dapat digunakan untuk mengambil sebuah keputusan berdasarkan apa yang dilihat. Secara umum, *computer vision* merupakan sebuah teknologi yang digunakan oleh komputer untuk dapat melihat (Mashita, 2020).

Computer vision adalah kombinasi antara *image processing* dan *pattern recognition*. *Computer vision* adalah konstruksi deskripsi objek fisik yang eksplisit dan jelas dari sebuah citra. *Output* dari *computer vision* adalah interpretasi beberapa pengukuran kuantitatif struktur dalam adegan tiga dimensi (Mashita, 2020).

2.7.1 *Image Processing*

Image processing adalah metode untuk mengubah citra menjadi bentuk digital dan melakukan beberapa operasi pemrosesan untuk memperolah informasi yang penting dari citra tersebut. *Input* pada proses ini adalah citra dan *output* prosesnya merupakan citra atau fitur yang terkait dengan citra itu.

2.7.2 *Pattern Recognition*

Klasifikasi atau penggambaran suatu citra dilakukan dengan proses mengukur ciri atau sifat utama dari suatu objek. Bagian terpenting dari teknik *pattern recognition* adalah bagaimana memperoleh informasi atau fitur penting yang terdapat pada citra. Contoh dari *pattern recognition* adalah sidik jari, raut wajah, gelombang suara, tulisan tangan,dll.

2.8 *Artificial Intelligence*

Menurut orang pertama yang mengusulkan nama *Artificial Intelligence* (AI), John McCarthy mengatakan bahwa tujuan AI adalah untuk mengembangkan mesin yang berperilaku seolah-olah mereka cerdas. Tetapi definisinya dinilai terlalu luas, sehingga saat ini AI didefinisikan sebagai metode yang menggunakan pemikiran manusia sehingga mesin dapat melakukan aktivitas yang dapat bersaing dengan manusia.

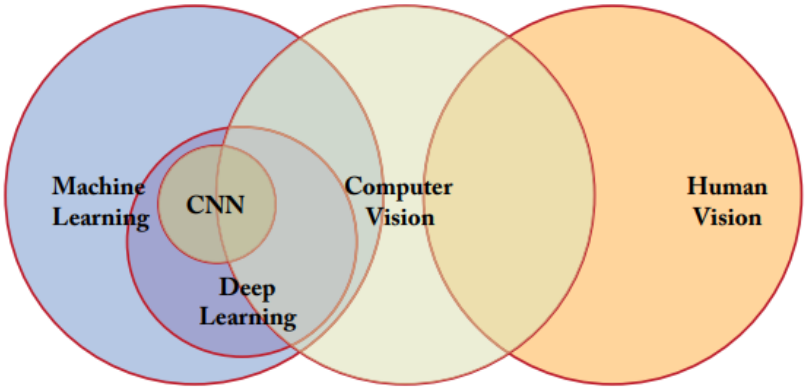
Meski pada level yang lebih rendah, AI kini menjadi ilmu yang mencakup banyak ilmu di dalamnya. Ilmu yang umum digunakan antara lain adalah pemrosesan bahasa natural atau *Natural Language Processing* (NLP), pengenalan ucapan atau *speech recognition*, pengelihatan komputer atau *computer vision*, dll. Di dalam AI sendiri, terdapat 2 metode yang dapat digunakan yaitu *machine learning* dan *deep learning* (Nurdiansyah, 2021).

2.9 *Machine Learning*

Machine learning (pembelajaran mesin) adalah pendekatan yang banyak digunakan untuk meniru bagaimana perilaku manusia untuk menyelesaikan suatu masalah dengan sendirinya. *Machine learning* memiliki dua aplikasi utama, yaitu klasifikasi dan prediksi dengan ciri khasnya berupa proses pembelajaran atau pelatihan (*training*). Oleh karena itu dibutuhkan data untuk *machine learning* mempelajarinya. Klasifikasi dalam *machine learning* digunakan untuk memilah (mengklasifikasikan) objek berdasarkan karakteristik tertentu, sedangkan prediksi atau regresi digunakan untuk menerka *output* dari suatu *input* berdasarkan data yang sudah dilakukan *training* sebelumnya (Mashita, 2020).

2.10 *Deep Learning*

Deep learning adalah cabang *machine learning* yang terinspirasi dari korteks manusia dengan menerapkan *artificial neural network* yang mempunyai banyak *hidden layer*. Karena kesuksesan besar dalam mempelajari *deep neural network*, *deep learning* berkembang lebih canggih sehingga dapat mendeteksi, mengklasifikasi, mengenali, dan verifikasi objek pada sebuah citra. Gambar 2.5 menunjukkan hubungan dari pengelihatan manusia, computer vision, machine learning, deep learning, dan CNN. *Deep learning* memiliki fitur untuk mengekstrak pola yang diperoleh dari data yang membantu model untuk membedakan antar kelas, sehingga fitur ini juga berperan untuk membantu mencapai prediksi dan akurasi yang baik (Mashita, 2020).



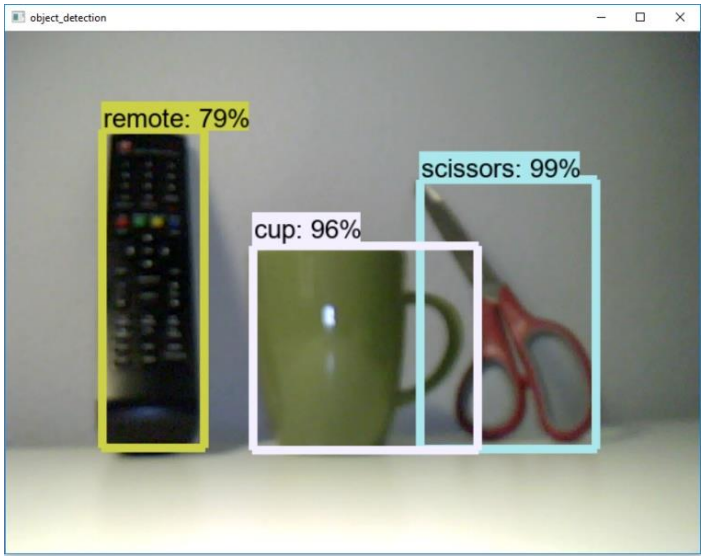
Gambar 2.5 Hubungan Antara Pengelihatan Manusia, *Computer Vision*, *Machine Learning*, *Deep Learning*, dan CNN
(Mashita, 2018)

2.11 *Transfer Learning*

Transfer learning dalam *computer vision* adalah metode yang memungkinkan kita untuk membangun model yang akurat secara efisien dan cepat. Biasanya, alih-alih model yang akan dilatih dari awal, *transfer learning* mewarisi potongan arsitektur dan parameternya dari model yang telah dilatih sebelumnya. Model-model ini dilatih pada kumpulan data untuk memecahkan beberapa masalah yang membutuhkan berbagai pola untuk dideteksi. Dengan cara ini model baru memanfaatkan pelatihan sebelumnya dan menghindari pembelajaran pola yang sudah diketahui. Daya komputasi dan waktu yang dibutuhkan untuk melatih model yang mengeksploitasi teknik ini adalah minimum karena hanya beberapa parameter yang perlu disetel. Ketika *transfer learning* digunakan, pengklasifikasi model yang telah dilatih sebelumnya dihapus dan basis konvolusi dipasang ke model baru. Salah satu strateginya adalah melatih semua parameter model baik pada basis konvolusi maupun pengklasifikasi. Dengan melakukan ini, hanya arsitektur asli basis yang digunakan kembali, dan model akan memerlukan kumpulan data besar untuk mempelajari tugas yang diperlukan (Matthaiou, 2021).

2.12 *Object Detection*

Object detection atau deteksi objek merupakan suatu teknik dalam *computer vision* yang berfungsi untuk mengetahui keberadaan dan letak suatu objek pada citra. Deteksi objek dapat dibagi menjadi *hard detection* dan *soft detection*. *Hard detection* adalah pendeteksian keberadaan dan lokasi dari objek sedangkan *soft detection* adalah pendeteksian adanya sebuah objek atau tidak. *Object detection* biasanya dilakukan dengan mencari setiap bagian citra untuk melokalisasi atau melabeli objek target, yang sifat geometris atau warnanya cocok dengan objek dalam data pembelajaran (*training*). *Object detection* dapat dinyatakan baik atau berhasil jika kemiripan antara model dan objek sesungguhnya cukup tinggi. Hasil deteksi objek pada beberapa objek divisualisasikan pada Gambar 2.6 (Jalled & Voronkov, 2016).

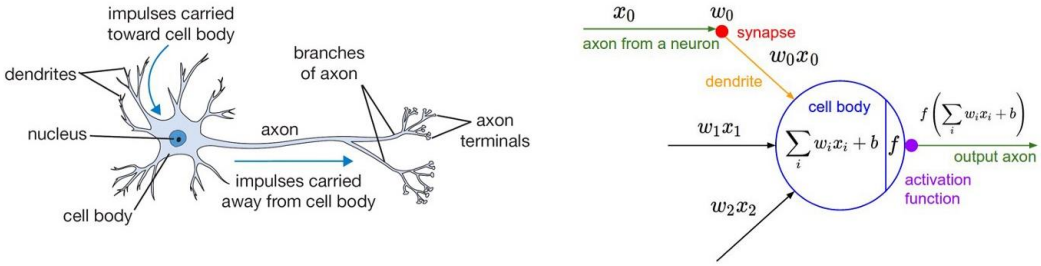


Gambar 2.6 Visualisasi *Object Detection*
(Mashita, 2019)

2.13 *Neural Network*

Neural network (jaringan saraf) adalah model yang terinspirasi oleh aktivitas neuron di dalam otak manusia. Setiap neuron di otak manusia saling berhubungan dan informasi mengalir dari setiap neuron. Gambar 2.7 berikut merupakan ilustrasi dari *neuron* dengan model matematisnya. Setiap neuron mengambil *input* dan melakukan operasi dot dengan (*weight*)

bobot, menambahkannya (*weighted sum*) dan menambahkan *bias*. Hasil dari operasi ini akan digunakan sebagai parameter dari fungsi aktivasi yang akan menjadi keluaran dari neuron (Sena, 2017).



Gambar 2.7 Ilustrasi *Neuron* dengan Model Matematis
(Sena, 2017)

2.14 *Convolutional Neural Network*

Convolutional Neural Network (CNN) muncul dari penelitian korteks visualisasi otak, dan telah digunakan dalam pengenalan citra sejak 1980. Dengan jumlah data pelatihan yang tersedia, CNN telah berhasil mencapai kinerja manusia super dalam sejumlah tugas visual yang kompleks. Mereka mendukung layanan pencarian citra, mobil *self-driving*, dan sistem klasifikasi video. CNN tidak terbatas pada persepsi visual tetapi juga berhasil dalam banyak tugas lain, seperti pengenalan suara atau *natural language processing* (NLP) (Géron, 2019).

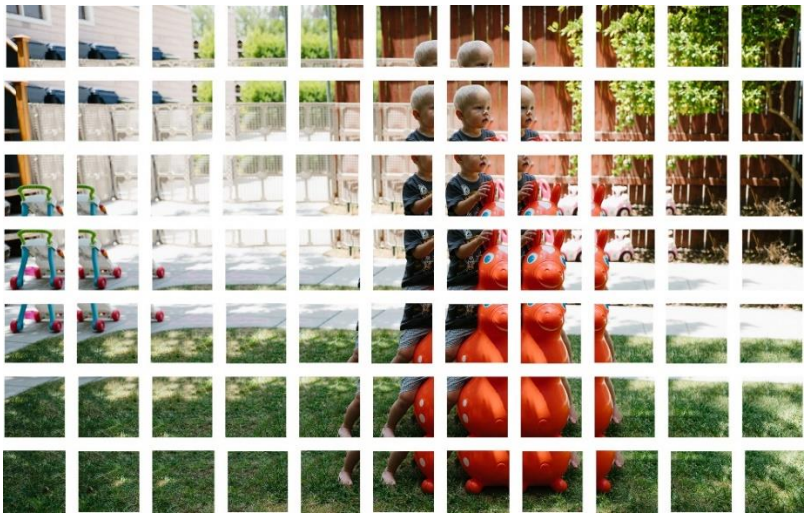
CNN adalah sebuah teknik yang terinspirasi dari cara manusia menghasilkan persepsi visual. Secara umum CNN tidak jauh berbeda dengan *neural network* lainnya. CNN terdiri dari *neuron* yang memiliki *weight* (bobot), *bias*, dan fungsi aktivasi. Seperti namanya, CNN menggunakan proses konvolusi (*convolution*) dengan menggeser *filter* (*kernel*) pada sebuah gambar atau citra *input*, kemudian komputer akan menerima dan mengolah informasi baru dari hasil perkalian antara *input* citra dengan *filter* yang digunakan (Geitgey, 2016).

2.14.1 Cara Kerja CNN

Berikut adalah cara kerja CNN, langkah demi langkah:

1. **Memecah Citra Menjadi Lebih Kecil yang Tumpang Tindih**

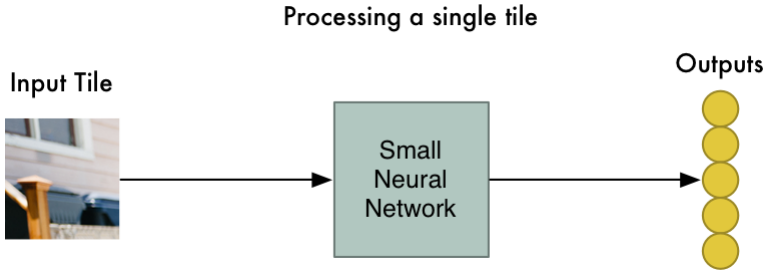
   Citra anak kecil menaiki kuda mainan yang dipecah menjadi lebih kecil dapat diilustrasikan pada Gambar 2.8. Dengan melakukan ini, citra asli diubah menjadi 77 citra yang lebih kecil dengan konvolusi sama.



Gambar 2.8 Citra Kecil dengan Konvolusi Sama
(Geitgey, 2016)

2. **Memasukkan Setiap Citra Kecil ke *Small Neural Network***

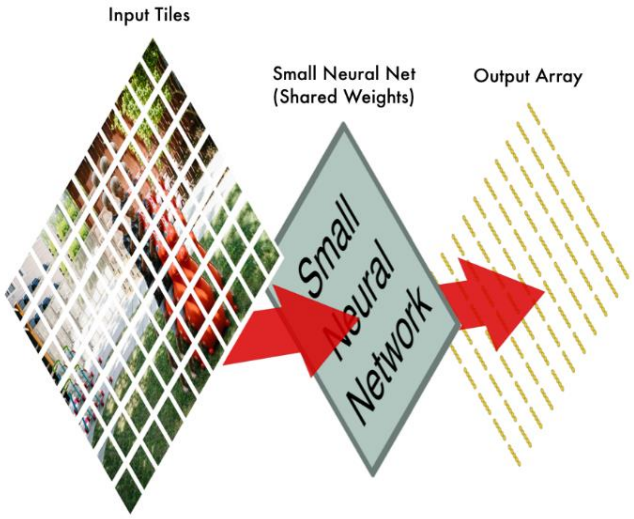
Setiap citra kecil hasil konvolusi digunakan sebagai *input* untuk menghasilkan sebuah representasi fitur. Ini memberi CNN kemampuan untuk mengidentifikasikan objek, terlepas di mana lokasi objek dalam citra. Proses ini dilakukan untuk semua bagian dari setiap citra kecilnya. Jika ada sesuatu yang menarik pada sebuah citra, maka bagian tersebut akan ditandai sebagai *object of interest*. Proses *small neural network* diilustrasikan pada Gambar 2.9



Gambar 2.9 Ilustrasi Proses *Small Neural Network*
(Geitgey, 2016)

3. **Menyimpan Hasil dari Setiap Citra Kecil ke Dalam *Array* Baru**

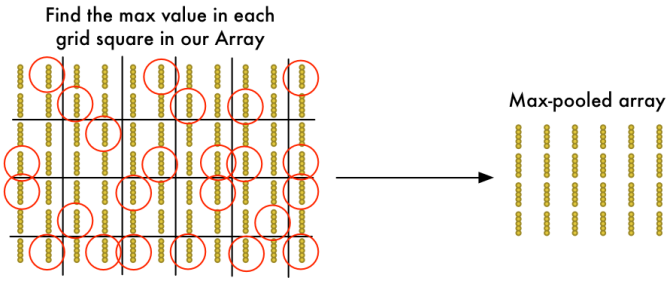
Agar tidak kehilangan susuan citra yang lebih kecil, dilakukan penyimpanan hasil dari pemrosesan menjadi *grid* (kisi-kisi) dalam susunan yang sama seperti citra asilnya, yang diilustrasikan pada Gambar 2.10. Dengan kata lain, telah dimulai dengan citra yang berukuran besar dan diakhiri dengan *array* yang sedikit lebih kecil yang merekam bagian mana dari citra asli yang paling tampak menarik (*object of interest*).



Gambar 2.10 Ilustrasi Merekam *Object Of Interest*
(Geitgey, 2016)

4. ***Downsampling***

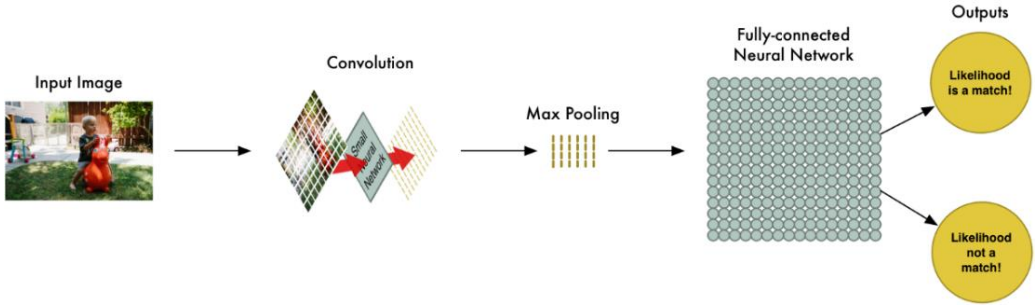
Downsampling berfungsi untuk mereduksi *array* yang masih terlalu besar yang penggunaannya dinamakan *max pooling* atau mengambil nilai piksel terbesar di setiap *array*, seperti yang diilustrasikan pada Gambar 2.11. Dengan cara ini, bahkan jika mengurangi informasi *input*, bagian terpenting akan tetap diambil.



Gambar 2.11 Ilustrasi *Max Pooling*
(Geitgey, 2016)

5. Membuat Prediksi

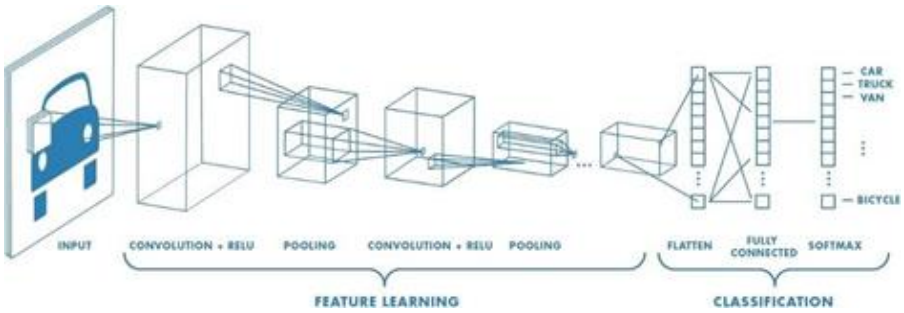
Dengan mengonversi citra berukuran besar menjadi *array* (sekelompok angka) yang agak kecil, langkah selanjutnya adalah menggunakan *array* kecil tersebut sebagai *input* ke dalam *neural network* lain. *Neural network* yang paling terakhir (*fully connected layer*) akan memutuskan apakah citra tersebut sesuai atau tidak. Secara umum, langkah kerja CNN tampak seperti Gambar 2.12.



Gambar 2.12 Langkah Kerja CNN
(Geitgey, 2016)

2.14.2 Arsitektur CNN

Arsitektur CNN terbilang mirip dengan pola koneksi sel saraf (*neuron*) dalam otak manusia. Arsitektur CNN seperti yang diilustrasikan pada Gambar 2.13 terbagi atas *feature extraction layer* dan *fully-connected layer* (Sena, 2017).



Gambar 2.13 Ilustrasi Arsitektur CNN
(Lina, 2019)

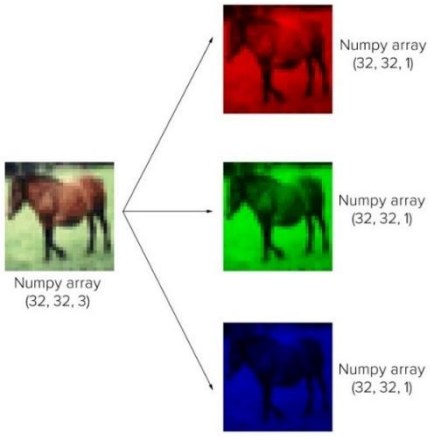
a. *Feature Extraction Layer*

Proses yang terjadi adalah "*encode*" suatu citra menjadi fitur yang merupakan representasi digital dari citra tersebut (*feature extraction*). *Feature Extraction Layer* terdiri atas dua bagian utama, yaitu *convolutional layer* dan *pooling layer*. *Convolution layer* bekerja berdasarkan prinsip *sliding window*. *Pooling layer* digunakan untuk meringkas informasi yang dihasilkan oleh *convolution layer* (Lina, 2019).

b. *Fully-Connected Layer*

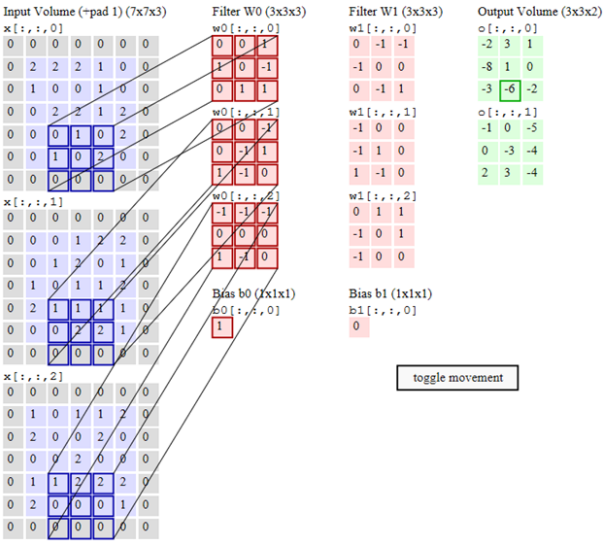
Fully-connected layer adalah lapisan penghubung *neuron* dari lapisan sebelumnya dengan lapisan berikutnya, seperti dengan *neural network* biasa. Aktivitas dari lapisan sebelumnya harus diubah menjadi vektor satu dimensi (*flatten*) terlebih dahulu sebelum terhubung ke semua *neuron*. *Fully-connected layer* umumnya digunakan pada metode *multilayer perceptron* (MLP) yang didgunakan untuk mengklasifikasikan suatu data (Lina, 2019).

1. *Convolutional Layer*



Gambar 2.14 *Image* RGB
(Lina, 2019)

Gambar 2.14 menunjukkan citra RGB (*Red*, *Green*, *Blue*) dengan ukuran 32 x 32 piksel. Namun sebenarnya adalah *array* multi dimensi dengan ukuran 32 x 32 x 3 piksel (3 jumlah channel). *Convolutional layer* terdiri atas *neuron* yang tersusun membentuk sebuah *filter* dengan panjang dan tinggi (*pixels*). Sebagai contoh, lapisan pertama pada *feature extraction layer* adalah *convolutional layer* dengan ukuran 7 x 7 x 3. Panjang 7 piksel dan tinggi 7 piksel dengan *channel* 3. *Filter* ini akan digeser keseluruhan bagian *input* citra dengan operasi "*dot*" untuk menghasilkan *output* atau berupa *feature map* seperti yang diilustrasikan pada Gambar 2.15.



Gambar 2.15 Tampilan *Feature Map*
(Lina, 2019)

2. *Stride*

Stride merupakan parameter yang menentukan seberapa banyak *filter* yang digeser. Jika nilai *stride* adalah satu, maka *filter* akan bergeser sebanyak satu piksel secara horizontal kemudian vertikal. Semakin kecil nilai *stride*, semakin detail informasi yang didapatkan, akan tetapi membutuhkan lebih banyak perhitungan sehingga menyebabkan lamanya waktu dalam proses pelatihan (*training*).

3. *Padding*

Padding merupakan parameter untuk menentukan jumlah piksel yang akan ditambahkan ke setiap sisi luar dari citra *input*. Hal ini digunakan untuk membuat dimensi *output* dari *convolutional layer* tetap sama dengan *input*, atau setidaknya tidak berkurang signifikan.

4. Fungsi Aktivasi

Fungsi aktivasi adalah sebuah fungsi yang mengambil *input* dari sebuah *neuron* dan menghasilkan *output* yang kemudian diteruskan ke *neuron* lain. Fungsi aktivasi menambahkan properti non-linear pada algoritma, Properti non-linear ini sangat penting agar algoritma dapat menyelesaikan masalah yang kompleks (Ng, 2017).

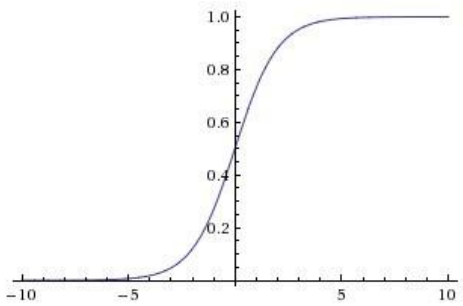
Ada beberapa fungsi aktivasi yang biasa digunakan dalam CNN, di antaranya adalah:

a. Sigmoid

Sigmoid memiliki bentuk formula seperti pada persamaan 2.1 berikut:

$$\sigma(x) = \frac{1}{(1 + e^{-x})} \tag{2.1}$$

Sigmoid memiliki *range* antara nilai 0 – 1 dengan dengan kurva seperti huruf S. Fungsi ini mudah dipahami dan diterapkan, tetapi sudah jarang digunakan karena kurangnya gradien data, titik tengah kurva dari fungsi ini bukan nol, gradien tersaturasi, dan konvergensi yang lambat. Grafik kurva sigmoid dapat dilihat pada Gambar 2.16.



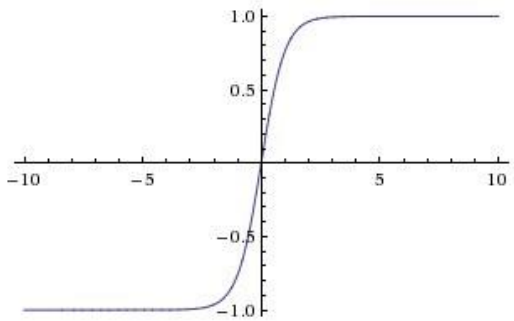
Gambar 2.16 Grafik Kurva Sigmoid

b. Tanh (*Hyperbolic Tangent*)

Tanh memiliki bentuk formula seperti pada persamaan 2.2 berikut:

$$\sigma(x) = \tanh(x) \tag{2.2}$$

Fungsi tanh merupakan fungsi aktivasi yang menyelesaikan masalah sigmoid. Karena titik tengah fungsi tanh adalah 0, ini memecahkan salah satu masalah fungsi sigmoid, yaitu masalah optimasi fungsi aktivasi. Fungsi aktivasi tanh lebih mudah dioptimasi, tetapi masalah gradien tersaturasi dan *vanishing gradient* masih tetap ada. Oleh karena itu, fungsi tanh jarang digunakan. Grafik kurva tanh dapat dilihat pada Gambar 2.17.



Gambar 2.17 Grafik Kurva Tanh

c. ReLU

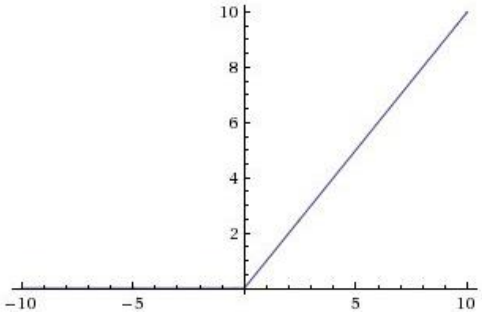
ReLU merupakan fungsi aktivasi yang sangat populer dalam beberapa tahun terakhir. ReLU memiliki bentuk formula seperti pada persamaan 2.3 berikut:

$$\sigma(x) = \max(0, x) \tag{2.3}$$

Dengan fungsi yang sangat sederhana, ReLU mempunyai kinerja yang lebih tinggi dibandingkan fungsi aktivasi lainnya. Selain itu, ada beberapa alasan lain mengapa fungsi ReLU lebih dipilih saat membangun *neural network*:

- Konvergensi data berkecepatan tinggi
- Pergerakan data yang stabil, tidak mengalami permasalahan gradien hilang
- *Cost-effective*
- Mudah dioptimasi

Namun, fungsi ReLU memiliki kelemahan yaitu ketika banyak *input* negatif, semua *output* akan diberi nilai 0, sehingga *neuron* tidak mengeluarkan informasi dan mati. Grafik kurva ReLU dapat dilihat pada Gambar 2.18.



Gambar 2.18 Grafik kurva ReLU

d. Leaky ReLU

Leaky ReLU digunakan untuk menyelesaikan permasalahan *neuron* mati dari ReLU. Leaky ReLU memiliki bentuk formula seperti pada persamaan 2.4 berikut:

$$\sigma(x) = \max\left(\frac{x}{100}, x\right) \tag{2.4}$$

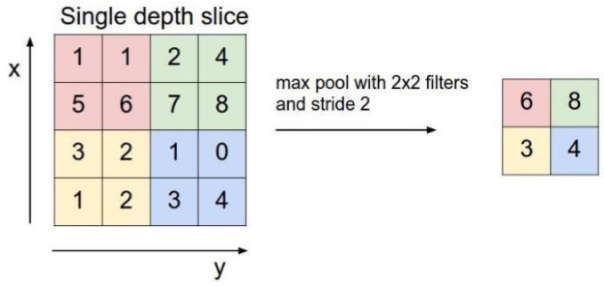
Fungsi Leaky ReLU memungkinkan ReLU untuk mengeluarkan nilai negatifnya. Nilai negatif tersebut kemudian dibagi dengan beberapa konstanta tertentu sehingga nilainya menjadi kecil.

5. *Pooling Layer*

Pooling layer merupakan proses setelah *convolutional layer*. Seperti halnya *convolutional layer*, *pooling layer* terdiri dari *filter* dengan *stride* yang akan bergeser di

seluruh area *feature map* hasil konvolusi. *Pooling* yang paling umum digunakan adalah *Max Pooling* dan *Average Pooling*. Tujuan dari *pooling* adalah mengurangi ukuran dari *feature map* (*downsampling*), sehingga mengurangi jumlah parameter, dan mempercepat perhitungan.

Pada Gambar 2.19 digunakan *Max Pooling* 2 x 2 dengan stride 2, setiap *filter* digeser, dengan ukuran *input* 4 x 4, operasi mengambil nilai maksimum untuk masing-masing dari empat angka dalam *input* dan menghasilkan ukuran *output* baru 2 x 2. Sedangkan *Average Pooling* akan memilih nilai rata-ratanya.



Gambar 2.19 Contoh *Max Pooling*
(Lina, 2019)

6. *Dropout Regularization*

Dropout merupakan teknik untuk mengondisikan *neural network* dimana beberapa *neuron* akan dipilih secara *random* dan tidak digunakan selama proses *training* agar sebuah *neural network* tidak mengalami *overfitting*. *Neuron* akan dibuang secara *random*.

7. Softmax *Classifier*

Fungsi aktivasi Softmax sering digunakan di layer yang terakhir dari *neural network* untuk menghitung dan mencari probabilitas kelas target. Semakin besar nilai Softmax, semakin tinggi kemungkinan bahwa suatu data adalah bagian dari kelas tersebut. Fungsi Softmax memiliki bentuk formula seperti pada persamaan 2.5 berikut:

$$\sigma(Xi) = \frac{e^{Xi}}{\sum_{j=0}^{k} e^{Xj}} \tag{2.5}$$

8. *Loss Function*

Loss function merupakan fungsi yang menggambarkan nilai kerugian relatif terhadap semua kemungkinan yang dihasilkan oleh sebuah model. *Loss function* berfungsi ketika model pembelajaran memberikan kesalahan yang perlu diperhatikan. *Loss function* yang baik adalah yang menghasilkan nilai *error* yang paling rendah.

9. *Learning Rate*

Learning rate adalah *hyperparameter neural network* untuk menghitung nilai koreksi *weight* (bobot) saat proses pelatihan (*training*). Nilai *learning rate* mempengaruhi lama waktu proses *training*. Semakin kecil *learning rate* mengakibatkan laju perubahan pada *weight* menjadi pelan. Semakin besar *learning rate* mengakibatkan laju perubahan *weight* menjadi besar sehingga sulit menemukan solusi.

10. *Batch Size*

Batch size merupakan total sampel data yang disebar ke *neural network*. Sebagai contoh jika terdapat 200 data set dengan nilai *batch size* 5, maka akan digunakan 5 data pertama dari 200 data lalu dilakukan *training* sampai selesai kemudian digunakan 5 data kedua dari 200 data, dan seterusnya sampai 5 sampel data ke 40.

2.15 *Confusion Matrix*

Confusion matrix merupakan tabel yang berisikan informasi hasil klasifikasi atau deteksi yang berupa nilai aktual dan prediksi. *Confusion matrix* adalah metode untuk menganalisis seberapa bagus kinerja sebuah model dalam mengidentifikasi data maupun objek (Mashita, 2020).

Tabel 2.2 Tabel *Confusion Matrix*

		Nilai Aktual	
		TRUE	FALSE
Nilai Prediksi	TRUE	TP (*True Positive*) *Correct Result*	FP (*False Positive*) *Unexpected Result*
	FALSE	FN (*False Negative*) *Missing Result*	TN (*True Negative*) *Correct Absence of Result*

Dengan keterangan sebagai berikut:

1. TP (*True Positive*): *output* dimana model dengan benar memprediksi kelas positif. Contohnya, terdapat objek lampu lalu lintas, maka model memprediksi bahwa objek tersebut adalah lampu lalu lintas.
2. TN (*True Negative*): *output* dimana model dengan benar memprediksi kelas negatif. Contohnya, tidak terdapat objek lampu lalu lintas, maka model memprediksi bahwa objek tersebut bukan lampu lalu lintas.
3. FP (*False Positive*): *output* dimana model tidak dapat memprediksi kelas positif. Contohnya, tidak terdapat objek lampu lalu lintas, tetapi model memprediksi objek lain sebagai lampu lalu lintas.
4. FN (*False Negative*): *output* dimana model tidak dapat memprediksi kelas negatif. Contohnya, terdapat objek lampu lalu lintas, tetapi model tidak memprediksi bahwa objek tersebut adalah lampu lalu lintas.

Gambar 2.20 berikut merupakan ilustrasi dari *cofusion matrix*.

Gambar 2.20 Ilustrasi dari *Confusion Matrix*
(Janahiraman, 2019)

Hasil dari Tabel 2.2 dapat digunakan untuk menghitung nilai akurasi sebuah model. Akurasi merepresentasikan seberapa akurat sistem atau model dapat mengklasifikasi atau mendeteksi data dengan benar dan akurat. Tujuannya adalah agar akurasi dapat ditunjukkan kebenarannya. Akurasi dapat didefinisikan sebagai tingkat kemiripan antara nilai aktual dengan prediksi. Nilai akurasi dapat dihitung dengan persamaan 2.6 (Mashita, 2020).

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \; x \; 100\% \tag{2.6}$$

2.16 Python

Python adalah salah satu bahasa pemrograman interpretatif multifungsi yang dikembangkan khusus untuk membuat *source code* yang mudah dipahami dan dapat dipasang di berbagai platform. Python juga memiliki pustaka lengkap yang memungkinkan pemrogram untuk membuat aplikasi yang kompleks. *Data science* dan *Internet of Things* adalah salah satu yang berhubungan erat dengan bahasa pemrograman Python (Dewi, 2018).

2.17 Tensorflow *Object Detection* API

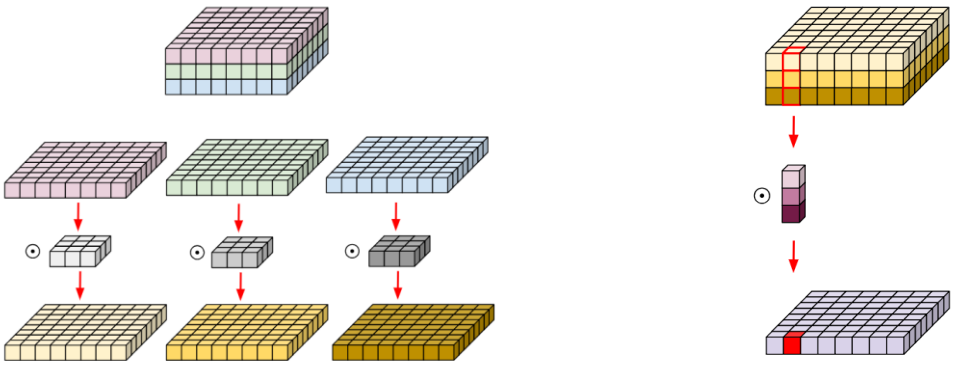
Tensorflow *Object Detection* API merupakan kerangka kerja (*framework*) *open source* yang diluncurkan oleh Google yang dapat digunakan untuk membuat program deteksi objek. Tensorflow *Object Detection* API menyediakan model pralatih seperti SSD Mobilenet V1 dan V2, Faster R-CNN, R-FCN, dll. Model tersebut dapat dilakukan untuk pendeteksian objek dengan menghasilkan akurasi dan *bounding box* untuk lokasi setiap objek yang akan dideteksi.

2.18 MobileNet

MobileNet adalah model arsitektur *Convolution Neural Network* (CNN) yang secara eksplisit berfokus pada klasifikasi citra. Alih-alih menggunakan *convolution layer* standar, MobileNet menggunakan *depth wise separable convolution layers*. Yang membuat model ini menonjol adalah bahwa arsitekturnya mengurangi biaya komputasi dan daya komputasi yang sangat rendah diperlukan untuk menerapkan *transfer learning* (Sanjana, 2020).

2.18.1 MobileNet V1

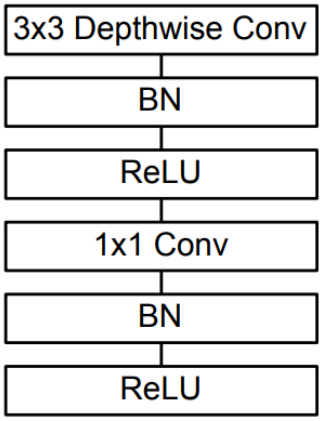
MobileNet V1 adalah adaptasi dari model MobileNet. Pada MobileNet V1, kotak konvolusi pada gambar yang diberikan yang terdiri dari konvolusi *depthwise* dan *pointwise*. Tujuan dari *depthwise separable convolution* ini untuk mereduksi komputasi dan ukuran dari model serta mengurangi jumlah parameter lebih dari 7 kali lipat dari penggunaan konvolusi standar. *Depthwise separable convolution* diilustrasikan pada Gambar 2.21 (Sanjana, 2020).



(a) (b)

Gambar 2.21 (a) *Depthwise Convolution* (b) *Pointwise Convolution* (Howard, 2017)

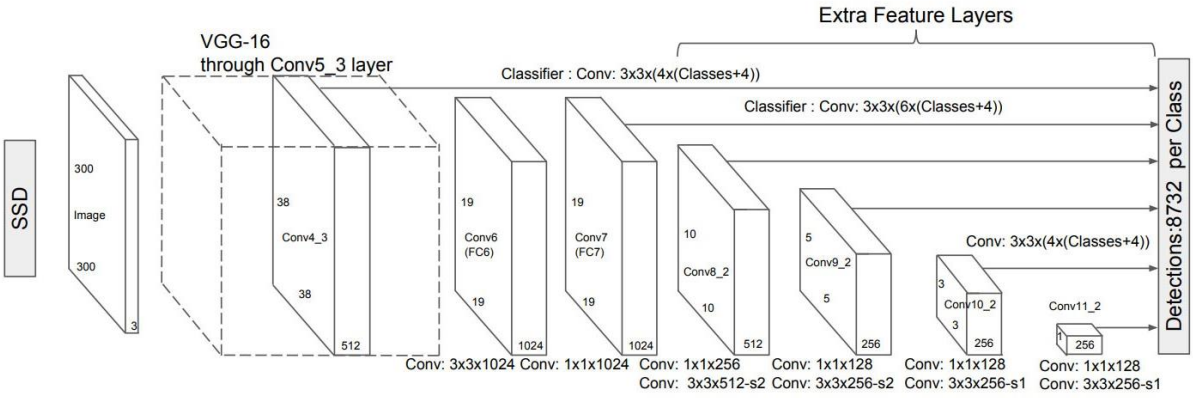
Gambar 2.22 menunjukan gambar blok layer yang tersusun dari *depthwise convolution* dan *pointwise convolution*, masing-masing layer tersebut diikuti oleh *batch normalization* dan fungsi aktivasi ReLU. Blok layer tersebut kemudian disusun secara berulang membentuk arsitektur MobileNet V1. Berdasarkan pustaka keras, MobileNetV1 memiliki jumlah *layer* total sebanyak 87 *layer* yang terbagi ke dalam 13 blok (Sanjana, 2020).



Gambar 2.22 *Depthwise Separable Convolution*
(Howard, 2017)

2.19 *Single Shot Multibox Detector* (SSD)

Single Shot MultiBox Detector (SSD) adalah sebuah metode pendeteksian objek dalam sebuah citra menggunakan *single deep neural network*. SSD terdiri dari dua bagian penting, yang pertama adalah *base network* yang umum digunakan untuk klasifikasi citra, yaitu VGG-16. Bagian kedua SSD menggunakan matriks yang dihasilkan oleh *base network* yang terdiri dari lapisan *neural network* yang melakukan proses konvolusi untuk mendeteksi dalam *citra input*. Lapisan pada bagian kedua dapat dikategorikan berdasarkan fungsinya, yaitu untuk mengekstraksi informasi dengan tingkat kompleksitas yang lebih tinggi dan untuk melakukan prediksi lokasi dan kategori objek berdasarkan matriks fitur yang diberikan sebagai *input*. SSD bekerja pada berbagai skala yang berbeda, sehingga mampu mendeteksi objek dengan berbagai ukuran/skala yang berbeda pada gambar. Arsitektur dari SSD yang diimplementasikan dengan VGG-16 sebagai *base network* dapat dilihat pada Gambar 2.23 (Sanjana, 2020).



Gambar 2.23 Arsitektur SSD
(Wei Liu, 2016)

2.20 SSD MobileNet V1

SSD MobileNet adalah sebuah model pendeteksian objek yang terdiri dari SSD sebagai basis model dan MobileNet sebagai *network model*. SSD akan mengatur pendeteksian objek dengan membuat bonding box dan MobileNet yang akan bekerja sebagai *feature extraction layer*. Ketika MobileNet V1 digunakan bersama dengan SSD, beberapa lapisan terakhir seperti FC, *max pool*, dan Softmax dihilangkan. Jadi, *output* dari lapisan konvolusi akhir di MobileNet digunakan, bersama dengan konvolusinya beberapa kali lagi untuk mendapatkan setumpuk *feature map*. Ini kemudian digunakan sebagai *input* untuk kepala deteksinya. Tabel 2.3 berikut merupakan *body architecture* dari MobileNet V1 (Sanjana, 2020).

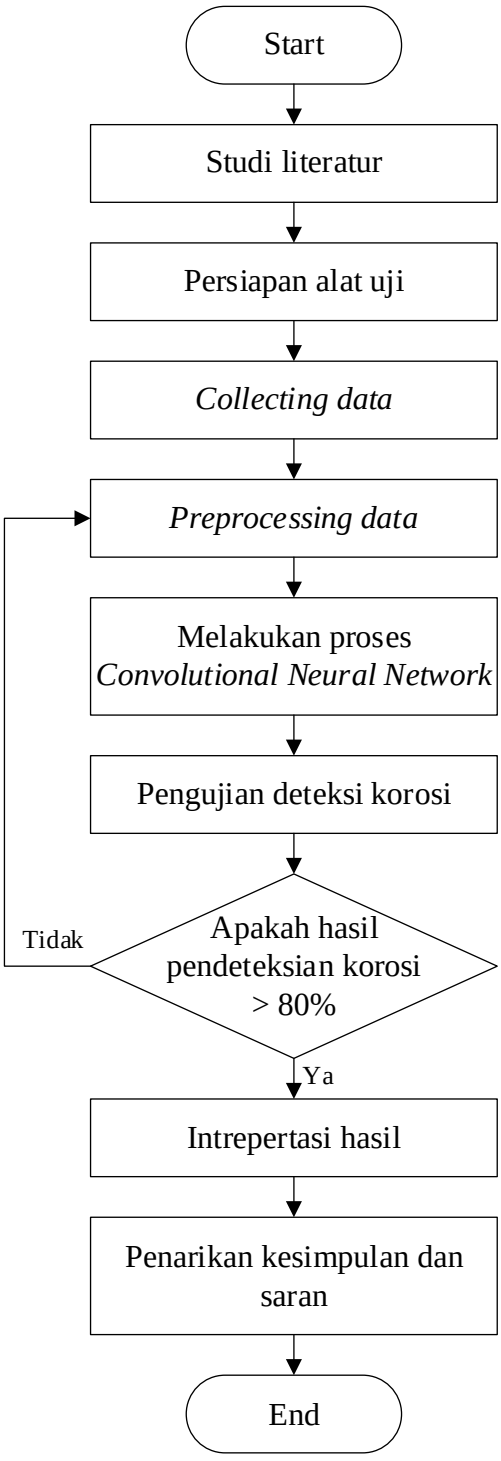
Tabel 2.3 *Body Architecture* MobileNet
(Howard, 2017)

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 ×1 × 1024
Softmax / s1	Classifier	1 ×1 × 1000

BAB III
METODOLOGI PENELITIAN

3.1 Diagram Alir Penelitian

Kegiatan yang akan dilakukan dalam penelitian ini digambarkan dalam bentuk diagram alir penelitian pada Gambar 3.1 berikut:



Gambar 3.1 Diagram Alir Tahapan Penelitian

3.2 Studi Literatur

Langkah pertama dalam penelitian ini adalah studi literatur. Studi literatur bertujuan untuk mendalami landasan teori yang akan digunakan dalam penelitian ini. Studi literatur dimulai dengan mencari jurnal penelitian sebelumnya yang membahas topik serupa, yaitu mengenai *deep learning, object recognition,* dan *Convolutional Neural Network* (CNN). Hasil yang diperoleh dikumpulkan dan digunakan sebagai referensi dalam penulisan serta perancangan eksperimen yang dilakukan.

3.3 Persiapan Alat Uji

Alat yang akan digunakan pada penelitian ini adalah Laptop Acer Aspire A315-41, *drone* DJI Mavic Air 2 seperti pada Gambar 2.2, dan software-software yang diperlukan dengan spesifikasi seperti yang terlihat pada Tabel 3.1, Tabel 3.2, Tabel 3.3, Tabel 3.4:

Tabel 3.1 Spesifikasi Laptop Acer Aspire A315-41

No.	Komponen	Spesifikasi
1.	*Processor*	AMD Ryzen 5 2500U *with* Radeon Vega Mobile Gfx (8 CPUs), ~2.0GHz
2.	RAM	8GB DDR4
3.	*System Type*	Windows 10 Home Single Language 64-bit
4.	GPU	AMD Radeon™ Vega 3

Tabel 3.2 Spesifikasi *Drone* DJI Mavic Air 2

No.	Komponen	Spesifikasi
1.	Bobot lepas landas	570 g
2.	Dimensi (PxLxT)	183 x 253 x 77 mm
3.	Kecepatan naik maks	4 m/s (Mode Sport dan Normal)
4.	Kecepatan turun maks	3 m/s (Mode Sport dan Normal)
5.	Kecepatan maks	19 m/s (Mode Sport) 12 m/s (Mode Normal) 5 m/s (Mode Tripod)
6.	Waktu penerbangan maks	34 Menit
7.	Jarak penerbangan maks	18.5 km

Tabel 3.3 Spesifikasi Kamera *Drone* DJI Mavic Air 2

No.	Komponen	Spesifikasi
1.	Sensor	CMOS ½ Piksel efektif: 12/48 MP
2.	Lensa	FOV: 84° 35 mm format setara: 24 mm Bukaan: f/2.8 Rentang pemotretan: 1 m hingga ∞
3.	ISO	Video: 100 – 6400
4.	Resolusi video	4K Ultra HD: 3840×2160 24/25/30/48/50/60 fps 2.7K: 2688×1512 24/25/30/48/50/60 fps FHD: 1920×1080 24/25/30/48/50/60/120/240 fps
5.	Bitrate video maks	120 Mbps

Tabel 3.4 *Tools* yang Digunakan

No.	*Tools*	Deskripsi
1.	Python 3.7	Bahasa pemrograman.
2.	Anaconda	Aplikasi untuk membuat *virtual environments* dan menginstall paket yang diperlukan untuk proses *deep learning*.
3.	Pycharm	IDE untuk penulisan dan pengujian kode.
4.	Tensorflow V1.15	Pustaka untuk keperluan *deep learning.*
5.	Keras	Pustaka yang digunakan untuk menyusun tensor-tensor dari Tensorflow.
6.	LabelImg	Pustaka untuk digunakan untuk melabeli kelas data.
7.	OpenCV	Pustaka yang mengolah gambar.

3.4 *Collecting* Data

Collecting data merupakan tahap pengumpulan gambar sebagai informasi *input*. Data-data tersebut berupa data set yang diperlukan dalam proses *training* dan *testing*, yaitu berupa gambar yang diunduh dari Google Images dan data langsung dari logam yang mengalami korosi di lingkungan Institut Teknologi Sepuluh Nopember (ITS). Data yang digunakan dalam penelitian ini sebanyak 200 citra korosi. Data tersebut dibagi menjadi dua kelompok yaitu *training* dan *testing*. Pembagian dari data tersebut adalah sebanyak 85% *training* dan 15% *testing* yang artinya sebanyak 170 citra digunakan untuk data *training* dan 30 citra untuk data *testing*.

3.5 *Preprocessing* Data

Tahap *preprocessing* data dibagi menjadi dua bagian, yaitu augmentasi data dan pemberian label data. Kedua hal ini diperlukan untuk membantu mempermudah program dalam proses *training* dan *testing*. Augmentasi data diperlukan untuk meningkatkan jumlah citra dengan melakukan berbagai transformasi pada citra. Sedangkan pemberian label data bertujuan agar proses *training* dapat dengan mudah mengenali objek dalam citra diantara objek lainnya. Preprocessing data digambarkan dalam bentuk diagram alir pada Gambar 3.1.

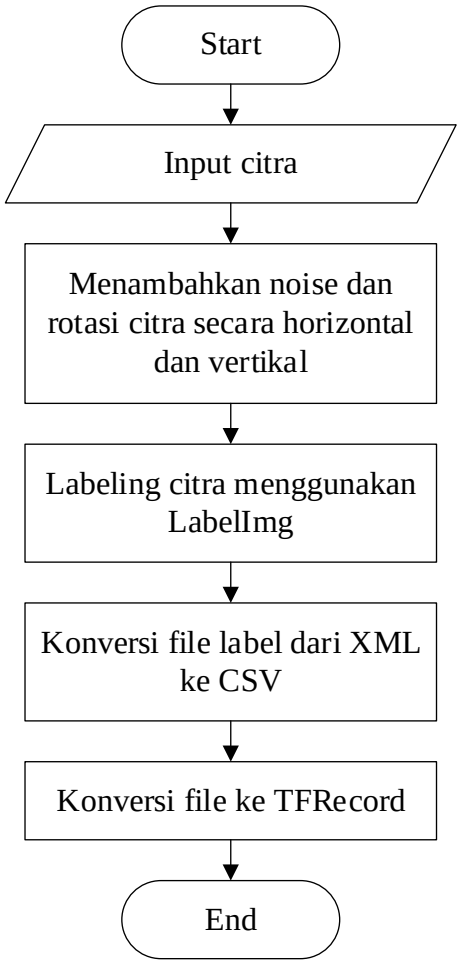
3.5.1 Augmentasi Data

Pada citra yang telah dikumpulkan dilakukan augmentasi berupa transformasi citra seperti *random noise*, *horizontal flip*, dan *vertical flip*. Dengan melakukan augmentasi data, satu citra dapat menghasilkan banyak citra dengan transformasi yang berbeda. Perubahan ukuran adalah proses yang umum dilakukan saat augmentasi data, di mana ukuran citra dengan piksel tertentu akan diubah menjadi piksel dengan ukuran yang lebih kecil. Hal ini dilakukan untuk meringankan beban dari *training* yang akan dilakukan. Meskipun selama proses *training* dengan besarnya dimensi citra dapat meningkatkan akurasi dan meningkatkan kemampuan program dalam melakukan deteksi tetapi besarnya dimensi citra akan membutuhkan kemampuan dan memori yang besar oleh GPU (*Graphic Processing Unit*).

3.5.2 Pelabelan Citra

Tahap selanjutnya dari *preprocessing* data adalah proses pemberian label berbentuk persegi pada bagian tertertu pada citra sehingga membentuk *Region of Interest* (ROI) menggunakan *software* LabelImg. *File* citra yang sudah diberi label akan disimpan dalam format XML. *File* XML yang dihasilkan berisi informasi tertentu, di antaranya adalah nama, resolusi gambar, dan akurasi pada setiap *bounding box*. *File* XML yang diperoleh setelah

melakukan pelabelan citra perlu di konversi ke *file* berekstensi CSV sehingga dapat di konversi TFRecord atau *Tensorflow Record* yang merupakan format pada penyimpanan Tensorflow. Generate TFRecord digunakan untuk merekam proses *training* data.



Gambar 3.2 Diagram Alir *Preprocessing* Data

3.6 *Training Data*

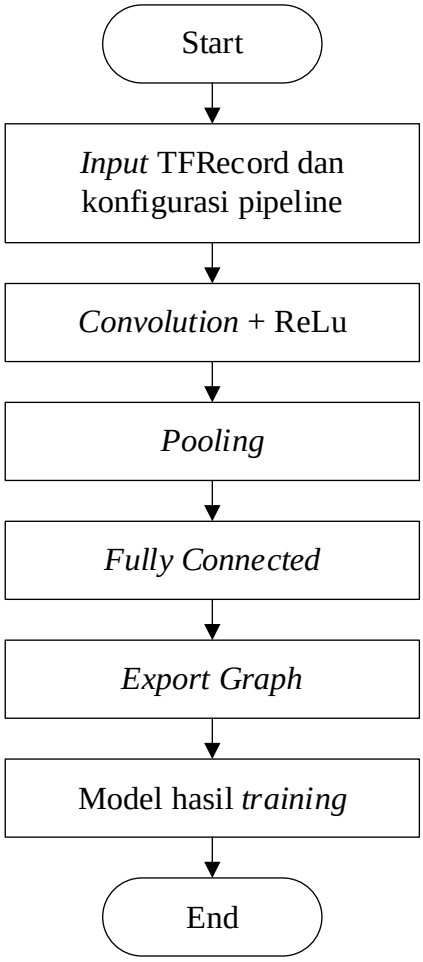
Untuk mendeteksi objek pada citra dengan metode *Convolutional Neural Network*, diperlukan proses *training* terlebih dahulu. Tujuan dari *training* adalah menemukan fitur atau ciri dari setiap citra kemudian dari fitur tersebut mengaktifkan *neuron-neuron* pada saat klasifikasi. Proses *training* dapat digambarkan dalam bentuk diagram alir pada Gambar 3.3. Pada penelitian ini tahapan yang akan dilakukan adalah sebagai berikut:

1. Data hasil pelabelan citra yang berupa TFRecord akan dijadikan *input* pada proses *training*.
2. Melakukan konfigurasi pipeline yang berguna untuk mengatur Convolutional Neural Network (CNN) yang digunakan, mengatur jumlah objek yang akan dideteksi, dan mengatur lokasi training serta pelabelan citra. Selain itu juga dapat mengatur hyperparameter CNN misalnya learning rate, batch size, dan steps. Pada penelitian ini digunakan konfigurasi model SSD Mobilenet V1
3. Setelah konfigurasi *pipeline*, maka dilakukan *training database*. *Training database* bertujuan untuk membuat model baru menggunakan data set yang diperoleh setelah pelabelan citra dan *pipeline* yang sudah dikonfigurasi. Proses ini menghasilkan *checkpoint* otomatis dibuat oleh *Tensorflow* berbentuk *graph tensor* yang berfungsi

untuk menyimpan informasi selama proses *training*. Proses akan dilakukan sampai mendapatkan hasil yang terbaik dengan nilai *loss* yang kecil. Adapun proses dalam *training database* adalah sebagai berikut:

a. Tahap *convolution*, yaitu *input* akan melewati *convolution layer* yang akan dilakukan proses konvolusi pada data *input* sehingga *input* menjadi sebuah matriks dengan ukuran tertentu. Setelah itu terdapat fungsi aktivasi *Rectifier Linier Unit* (ReLU) yang bertujuan untuk mengubah nilai negatif menjadi nilai nol akibat proses konvolusi.
b. Tahap *pooling* untuk mengurangi dimensi dari *feature map* dengan metode *max pooling*.
c. Tahap *flattening*, yaitu dimana *feature map* hasil *pooling* diubah menjadi vektor satu dimensi untuk menjadi *input* dalam tahap *fully connected layer*.

4. Setelah proses *training* selesai, hasil dari *checkpoint* terakhir didapatkan *output* berupa model data yang siap untuk dilakukan pengujian deteksi.



Gambar 3.3 Diagram Alir Proses *Training*

3.7 Pengujian Deteksi Korosi

Model objek penelitian yang digunakan adalah Jembatan Petekan yang dapat dilihat pada Gambar 3.4 yang berlokasi di Jalan Jakarta, Perak Utara, Kecamatan Pabean Cantikan, Kota Surabaya, Jawa Timur. Jembatan Petekan seperti yang dapat dilihat pada Gambar 3.4 berada 1,7 meter di atas permukaan sungai pada saat air pasang dan 1,2 meter di bawah jalan raya.

Gambar 3.4 Jembatan Petekan
(Dokumen Pribadi)

Pada tahap pengujian deteksi korosi, dilakukan pengambilan data berupa video menggunakan *drone* DJI Mavic Air 2 dengan cara menerbangkannya secara horizontal sepanjang 5 meter pada daerah dengan tanda panah merah pada Gambar 3.4. Bagian jembatan yang dilakukan pengujian pendeteksian dapat dilihat pada Gambar 3.5. Pengambilan data video memperhatikan variasi berupa jarak *drone* dengan objek dan kecepatan *drone* yang dapat dilihat pada Tabel 3.5.

Gambar 3.5 Bagian Jembatan yang Dilakukan Pengujian
(Dokumen Pribadi)

Tabel 3.5 Rancangan Percobaan

Variasi Pengujian		
Kombinasi ke-	Kecepatan *Drone*	Jarak *Drone* dengan Objek
1	0.6 m/s	1 m
2		2 m
3	0.9 m/s	1 m
4		2 m
5	1.3 m/s	1 m
6		2 m

3.8 *Setting Hyperparameter Convolutional Neural Network*

Data video yang telah diambil menggunakan *drone*, akan dilakukan pengujian deteksi korosi dengan model menggunakan variasi *setting hyperparameter batch size* dan *learning rate*. Variasi dari setting *hyperparameter* dapat dilihat pada Tabel 3.6.

Tabel 3.6 Variasi *Setting Hyperparameter* CNN

Variasi *Hyperparameter*		
Model	*Batch Size*	*Learning Rate*
1	4	0.001
2		0.01
3	8	0.001
4		0.01

3.9 Interpretasi Hasil

Hasil dari deteksi ini adalah model mendeteksi korosi pada *frame* setiap detiknya dan menampilkan hasil deteksi berupa *bounding box*. *Bounding box* sendiri merupakan persegi panjang imajiner sebagai titik referensi deteksi objek. *Bounding box* digunakan menghitung nilai akurasi dengan bantuan tabel *confusion matrix* yang terdapat pada Tabel 2.2 dengan empat penilaian, yaitu: *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN).

3.10 Penarikan Kesimpulan dan Saran

Pada tahap ini akan membahas tentang hasil dari penelitian serta pemberian saran untuk membantu penelitian yang akan dilakukan selanjutnya.

BAB IV
DESAIN SISTEM

4.1 Collecting Data

Pendekatan yang digunakan untuk memperoleh data yang dibutuhkan adalah dengan mengambil citra dari dua sumber, yaitu internet (Google Image) dan data langsung dari logam yang mengalami *uniform corrosion* di lingkungan Institut Teknologi Sepuluh Nopember (ITS). Dari kedua sumber tersebut didapatkan beberapa tipe dan posisi *uniform corrosion* yang beragam. Data yang diambil langsung dikumpulkan menggunakan kamera aksi digital GoPro Hero 9 Black dengan resolusi 20 Megapiksel. Citra diambil dari beberapa tempat pada lingkungan ITS antara lain tangga besi yang terdapat di Departeman Teknik Mesin ITS. Sampel citra korosi yang terkumpul dapat dilihat pada Gambar 4.1.

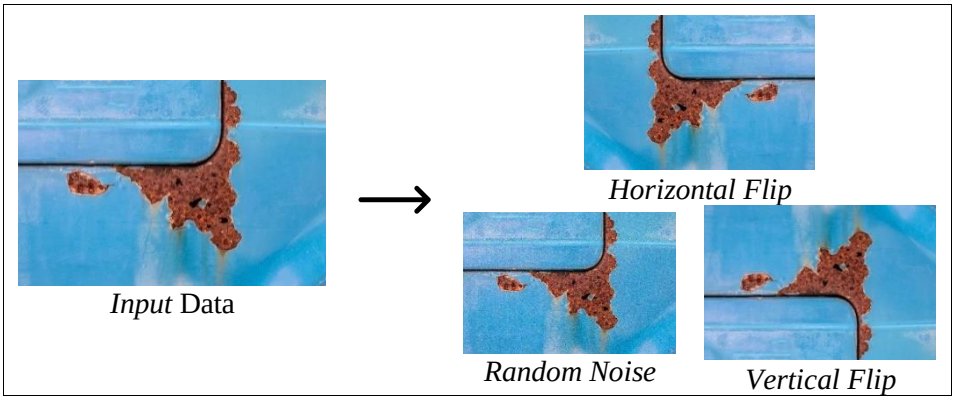
Gambar 4.1 Sampel Citra Korosi

4.2 *Preprocessing* Data

Prepocessing data mengacu pada semua transformasi data mentah sebelum proses *training* yang berguna untuk mempercepat proses *training*. Tahap *preprocessing* data dibagi menjadi dua bagian, yaitu augmentasi dan pelabelan data citra. Kedua hal ini diperlukan untuk membantu mempermudah program selama proses *training* dan *testing*.

4.2.1 Augmentasi Data

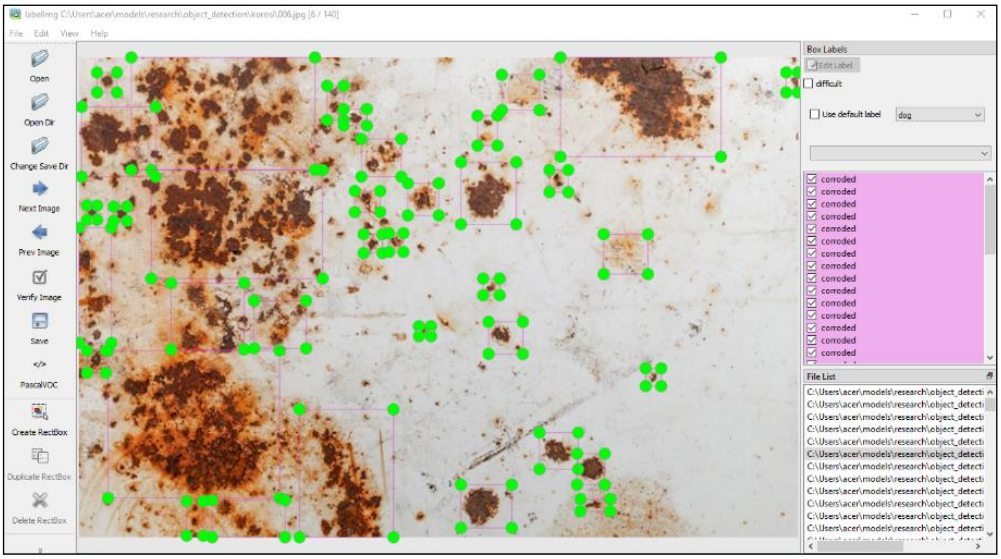
Augmentasi merupakan diperlukan untuk menambah jumlah citra dengan melakukan transformasi pada citra. Dengan melakukan augmentasi data, satu citra dapat menghasilkan banyak citra dengan transformasi yang berbeda. Pada penelitian ini dilakukan tiga bentuk transformasi, yaitu *random noise*, *horizontal*, dan *vertical flip* yang dapat dilihat pada Gambar 4.2, sedangkan *script* dari augmentasi data dapat dilihat pada Lampiran 3.



Gambar 4.2 Augmentasi Data

4.2.2 Pelabelan Citra

Pelabelan citra adalah langkah di mana data set akan diberi label satu per satu menggunakan aplikasi *open source* yaitu LabelImg. Citra akan diberi label sesuai dengan objek yang akan dideteksi, yaitu "korosi" seperti yang divisualisasikan pada Gambar 4.3. Tujuan dari pelabelan citra adalah untuk menyimpan informasi citra yang kemuduan disimpan dalam *file* dengan format XML. Data citra korosi dibagi menjadi dua kelompok yaitu *training* dan *testing*. Dengan total data set sebanayak 200, dengan pembagian data tersebut adalah sebanyak 85% *training* dan 15% *testing* yang artinya sebanyak 170 citra digunakan untuk data *training* dan 30 citra untuk data *testing*.



Gambar 4.3 Proses Pelabelan Korosi

4.2.2.1 Penggabungan dan Konversi XML ke CSV

Output dari pelabelan citra berupa 200 *file* dengan format XML (*Xtensible Markup Language*) seperti pada Gambar 4.4, yang kemudian akan digambungkan dengan cara dikonversi menjadi *file* dengan format CSV (*Comma-Separated Values*). Gambar 4.5 merupakan perintah yang digunakan untuk menjalankan *script* konversi format XML ke CSV.

```
<annotation>
    <filename>002.jpg</filename>
    <path>
C:\Users\acer\Desktop\korosi\images\train\002.jpg</path>
    <size>
        <width>484</width>
        <height>606</height>
        <depth>3</depth>
    </size>
    <object>
        <name>corroded</name>
        <bndbox>
            <xmin>66</xmin>
            <ymin>372</ymin>
            <xmax>189</xmax>
            <ymax>428</ymax>
        </bndbox>
    </object>
</annotation>
```

Gambar 4.4 Hasil Pelabelan Citra dalam Format XML

```
python xml_to_csv.py
```

Gambar 4.5 Perintah Konversi Format XML ke CSV

Dikarenakan terdapat 200 data set, maka akan terdapat 200 *file* XML. Untuk menggabungkan 200 *file* menjadi 1 *file* saja maka diperlukan perubahan format menjadi CSV. *Script* yang digunakan untuk melakukan konversi XML menjadi CSV terdapat pada Lampiran 4. Gambar 4.6 berikut adalah hasil konversi berupa *file* data *train* dan *test* dalam format CSV.

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	001.png	640	480	corroded	1	346	180	480
3	001.png	640	480	corroded	182	348	640	480
4	001.png	640	480	corroded	208	1	515	345
5	001.png	640	480	corroded	166	133	195	181
6	001.png	640	480	corroded	156	187	193	241
7	001.png	640	480	corroded	1	61	13	159
8	001.png	640	480	corroded	518	165	640	349
9	001.png	640	480	corroded	515	1	562	79
10	001.png	640	480	corroded	556	80	583	165
11	001.png	640	480	corroded	116	303	134	320
12	001.png	640	480	corroded	114	335	128	349
13	001.png	640	480	corroded	62	333	78	346
14	001.png	640	480	corroded	180	82	195	126
15	002.jpg	640	480	corroded	95	221	245	295
16	002.jpg	640	480	corroded	315	213	503	296
17	002.jpg	640	480	corroded	527	244	555	274
18	003.png	640	480	corroded	354	312	430	397
19	003.png	640	480	corroded	484	357	541	439
20	003.png	640	480	corroded	561	155	640	272
21	003.png	640	480	corroded	521	377	608	480
22	003.png	640	480	corroded	365	47	640	107
23	003.png	640	480	corroded	198	159	252	198

Gambar 4.6 Hasil Konversi *File* CSV

Hasil konversi CSV berisi kolom nama citra serta *width* (lebar) dan *height* (tinggi) atau ukuran dimensi dari data *input*. Seperti pada Gambar 4.6, detail informasi pada citra 001.jpg memiliki dimensi gambar 640 x 480, sesuai dengan kolom *width* dan *height* pada data csv. Kolom *class* merupakan objek yang akan dideteksi, yaitu korosi. Kolom *xmin* dan *ymin* merupakan nilai minimum piksel pada masing-masing *width* dan *height*, sedangkan *xmax* dan *ymax* merupakan nilai maksimum piksel pada *width* dan *height* dari *bounding box* yang akan digunakan untuk menandai sebuah objek.

4.2.2.2 Konversi CSV ke TFRecord

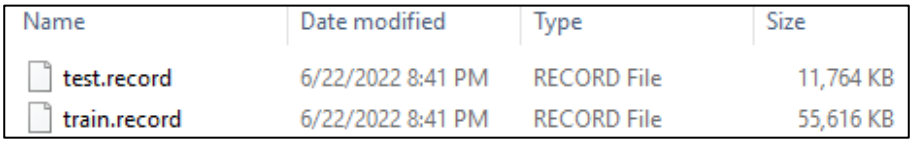
Pada tahap ini dilakukan konversi dari format *file* CSV ke TFRecord atau Tensorflow Record, hal ini dilakukan karena pada proses *training*, Tensorflow akan membaca data *input* atau yang disebut *feeding data*. Format TFRecord sendiri adalah format sederhana untuk menyimpan urutan dari *record* biner. Dengan mengubah *file* menjadi TFRecord maka penyimpanan akan lebih efisien karena membutuhkan ukuran yang kecil. *Script* yang digunakan untuk melakukan konversi *file* CSV ke TFRecord pada data *train* dan data *test* terdapat pada Lampiran 5. Gambar 4.7 merupakan perintah untuk menjalankan *scrip training* dan Gambar 4.8 merupakan perintah untuk menjalankan *script testing*. File yang telah dikonversi menjadi TFRecord dapat dilihat pada Gambar 4.9.

```
python generate_tfrecord.py --csv_input=images/train_labels.csv
--image_dir=images/train --output_path=train.record
```

Gambar 4.7 Perintah Konversi Data *Training* CSV ke TFRecord

```
python generate_tfrecord.py --csv_input=images/test_labels.csv
--image_dir=images/test --output_path=test.record
```

Gambar 4.8 Perintah Konversi Data *Testing* CSV ke TFRecord



Gambar 4.9 Hasil Konversi menjadi Format TFRecord

4.2.2.3 Konfigurasi Label *Map*

Konfigurasi label *map* adalah proses untuk memberi nama objek atau kelas yang akan dideteksi. Pada penelitian ini digunakan satu kelas, yaitu korosi. *Script* dari label *map* akan disimpan dengan format ".pbtxt" yang selanjutnya akan digunakan pada konfigurasi *pipeline*. Gambar 4.10 merupakan *script* label *map* yang digunakan pada penelitian ini.

```
item {
    id: 1
    name: 'korosi'
}
```

Gambar 4.10 Konfigurasi Label *Map*

4.3 Konfigurasi *Pipeline*

Pada penelitian ini digunakan model konfigurasi *pipeline* SSD Mobilenet V1 yang dilakukan pada beberapa parameter. Konfigurasi *pipeline* perlu dilakukan untuk menentukan *setting hyperparameter* yang dapat menghasilkan model dengan hasil yang paling baik.

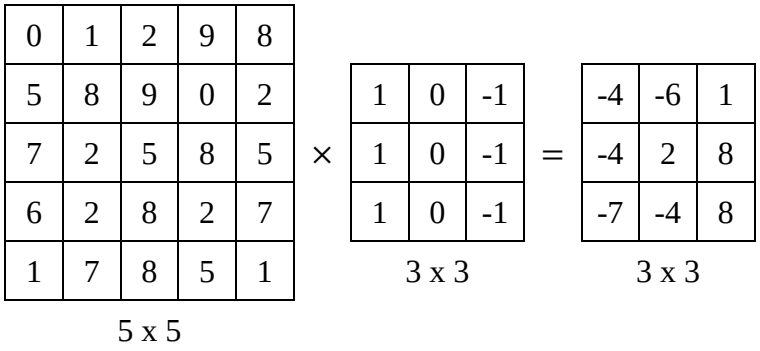
Konfigurasi *pipeline* pada penelitian ini menggunakan variasi *batch size* 4 dan 8, yang berarti jumlah sampel dari data yang disebarkan ke dalam *neural network* berjumlah 4 dan 8 dimana sampel tersebut akan diambil secara random dari semua sampel data set. Pada penelitian ini juga digunakan *learning rate* 0.001 dan 0.01, yang berguna untuk menghitung nilai koreksi bobot. Serta digunakan *number steps*: 200000 yang berarti jumlah iterasi maksimum dalam proses *training* adalah 200000 *steps*. *File* konfigurasi *pipeline* tersebut selanjutnya akan digunakan pada proses *training*. *Script* dari konfigurasi *pipeline* secara kesuluruhan dapat dilihat pada Lampiran 6.

4.4 Arsitektur Jaringan

Arsitektur jaringan yang digunakan dalam penelitian ini adalah SSD MobileNet V1 yang tersedia secara *default* oleh Tensorflow *Object Detection* API. Dengan memanfaatkan SSD Mobilenet V1 sebagai model pralatih, penelitian ini bertujuan untuk mendeteksi korosi yang terdapat pada material baja menggunakan kamera *drone*.

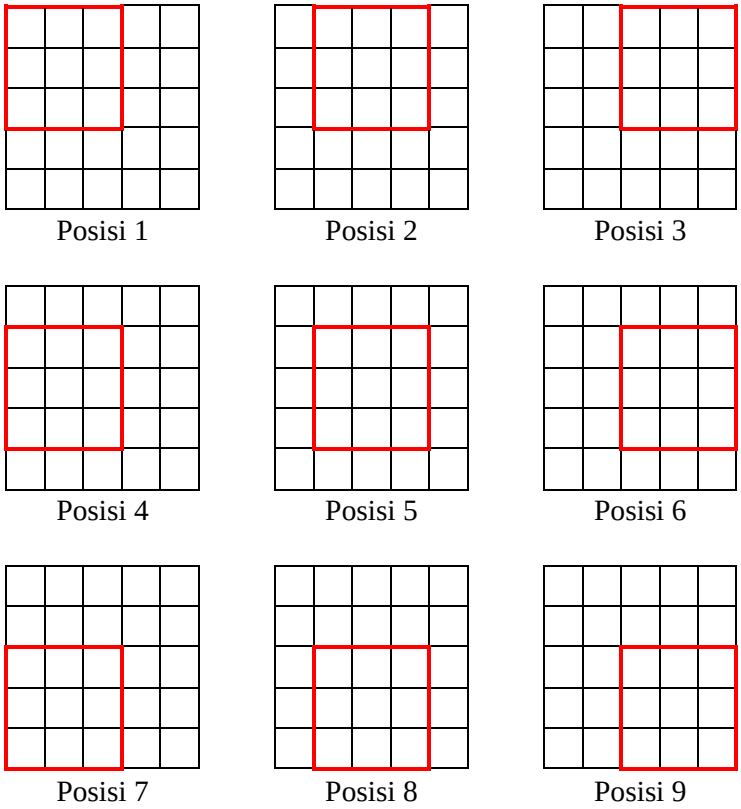
4.4.1 *Convolution Layer*

Convolution layer (lapisan konvolusi) adalah inti dari *Convolutional Neural Network.* Pada lapisan ini dihasilkan citra baru yang berisi *features* dari citra input. *Convolution layer* melakukan operasi konvolusi pada *output* dari *layer* sebelumnya. Proses yang terjadi pada *convolution layer* adalah pengaplikasikan *filter* pada citra *input*. Terdapat beberapa *filter* yang dapat diaplikasikan, bisa berukuran 1 x 1; 3 x 3; atau 5 x 5. Tujuan dari konvolusi pada citra adalah untuk mengekstraksi fitur dari citra *input*. Proses konvolusi akan menghasilkan transformasi linear dari data *input* sesuai dengan informasi spasial pada data.



Gambar 4.11 Proses Konvolusi

Pada Gambar 4.11 adalah contoh proses konvolusi. Digunakan sampel citra *input* dengan ukuran 5 x 5 dikarenakan keterbatasan penulisan dengan ukuran sebenarnya. *Filter* yang digunakan berukuran 3 x 3 dengan *stride* 1. Perhitungan konvolusi secara rinci dapat dilihat pada Lampiran 1. Proses perhitungan konvolusi dapat divualisasikan pada Gambar 4.12 berikut.



Gambar 4.12 Posisi Proses Konvolusi

Pada proses konvolusi, *filter* bergerak dari kiri atas (posisi 1) menuju ke kanan bawah (posisi 9) atau yang biasa disebut *sliding window*. Hasil konvolusi dari citra tersebut dapat dilihat pada Gambar 4.11 di sebelah kanan.

4.4.2 Fungsi Aktivasi

Fungsi aktivasi berfungsi untuk membuat *neural network* menjadi non-linear pada nilai hasil konvolusi. Pada penelitian ini, digunakan fungsi aktivasi ReLU. Fungsi aktivasi ReLU bekerja dengan cara mengubah nilai negatif menjadi nol (0) dan tetap mempertahankan nilai positif dari hasil konvolusi seperti pada Gambar 4.13.

-4	-6	1	=	0	0	1
-4	2	8		0	2	8
-7	-4	8		0	0	8
3 x 3				3 x 3		

Gambar 4.13 Ilustrasi Fungsi Aktivasi ReLU

4.4.3 *Pooling Layer*

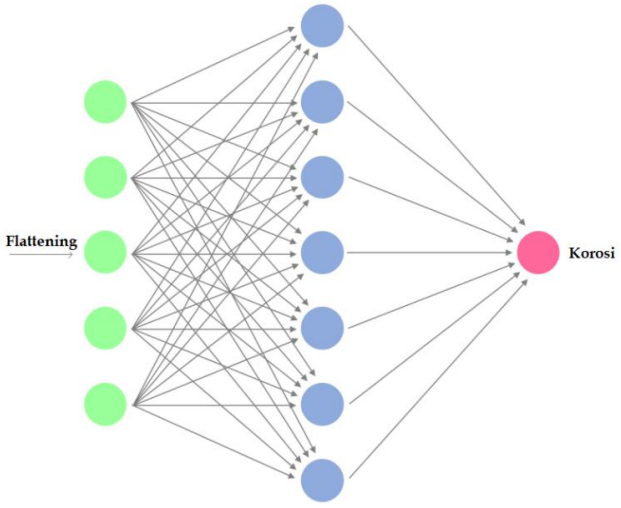
Pooling layer digunakan untuk mengurangi dimensi dari *feature map*, sehingga dapat mengurangi jumlah komputasi yang dilakukan. Pada Gambar 4.14 merupakan proses *pooling* dengan metode *max pooling* yang bertujuan untuk mengambil nilai maksimum pada citra *input*. Pada Gambar 4.14 terdapat lapisan dengan ukuran 3 x 3 menggunakan *filter* 2 x 2 dengan nilai *stride* 1, sehingga diperoleh hasil *pooling* dengan ukuran 2 x 2.

5	8	9	=	8	9
7	2	5		7	8
6	2	8		2 x 2	
3 x 3					

Gambar 4.14 *Pooling Layer*

4.4.4 *Fully Connected Layer*

Gambar 4.15 merupakan ilustrasi dari *fully connected layer* dengan 1 *hidden layer*. Proses *fully connected layer* dimulai dengan *input* hasil *flattening* kemudian melewati *hidden layer* dengan beberapa *node* yang menghasilkan klasifikasi objek. Fungsi aktivasi yang digunakan pada *fully connected layer* untuk melakukan klasifikasi adalah Softmax. Softmax akan menghsilkan *output* yang berfungsi untuk menghasilkan probabilitas untuk sebuah objek dan nilai tertinggi. Dengan memilih *output* yang mempunyai nilai probabilitas tinggi, maka akan menghasilkan nilai klasifikasi yang tepat.



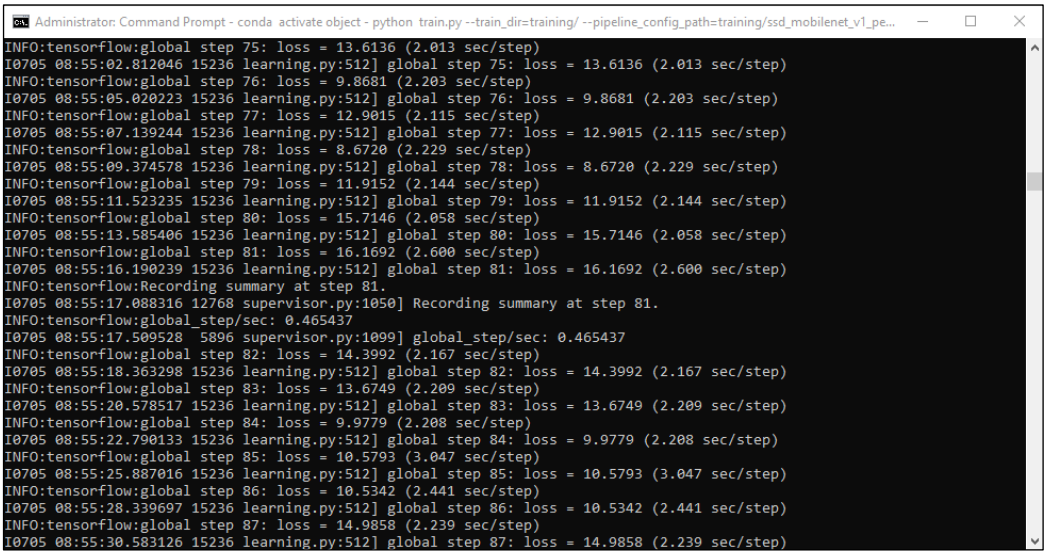
Gambar 4.15 Ilustrasi *Fully Connected Layer*

4.5 *Training* Model

Tahap *training* adalah tahap utama dimana *neural network* dilatih untuk mempelajari suatu pola yang diinginkan untuk menghasilkan suatu pengenalan objek dengan nilai akurasi yang tinggi dan *loss* yang rendah. Gambar 4.16 adalah perintah untuk menjalankan proses *training* dan Gambar 4.17 adalah proses dari *training* itu sendiri.

```
python train.py --train_dir=training/ --pipeline_config_path
=training/ssd_mobilenet_v1_pets.config -logtostderr
```

Gambar 4.16 Perintah *Training* Model

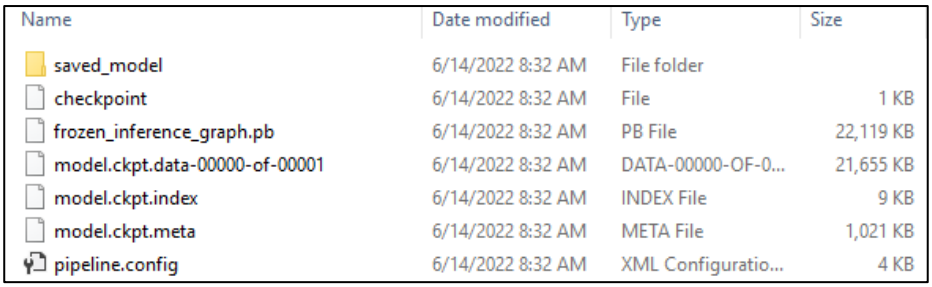


Gambar 4.17 Proses *Training* Model

Selama proses *training* model, Tensorflow akan membuat *checkpoint* untuk menyimpan informasi selama proses *training* seperti yang digamabarkan pada Gambar 4.17. Setelah proses *training* selesai, hasil dari *checkpoint* terakhir akan dikonversi menjadi *file* berupa *frozen inference graph* sehingga dapat digunakan untuk pengujian untuk pendeteksian objek. Gambar 4.18 adalah perintah untuk menjalankan ekspor checkpoint terakhir, yaitu pada penelitian ini adalah 200000. Model yang telah diekspor dapat dilihat pada Gambar 4.19.

```
python export_inference_graph.py --input_type image_tensor
--pipeline_config_path training/ssd_mobilenet_v1_pets.config
--trained_checkpoint_prefix training/model.ckpt-200000
--output_directory new_graph
```

Gambar 4.18 Perintah Ekspor Model



Gambar 4.19 Model Hasil *Training*

Pemantauan proses *training* dapat menggunakan modul yang telah tesedia pada Tensorflow, yaitu Tensorboard. Tensorboard adalah *toolkit open-source* yang memungkinkan untuk memvisualisasikan berbagai informasi berguna tentang model *neural network*. Gambar 4.20 adalah perintah untuk menjalankan Tensorboard melalui command prompt:

```
tensorboard --logdir=training
```

Gambar 4.20 Perintah untuk Menjalankan Tensorboard

Setelah melakukan perintah untuk menjalankan Tensorboard, untuk menampilkannya maka harus membuka alamat localhost:6006 pada *browser*. Tensorboard akan memanggil *file checkpoint* yang dibuat selama proses *training*. Pada Tensorboard dapat dilakukan analisis terhadap beberapa parameter yang dihasilkan dari proses *training*, seperti *global step* dan nilai total *loss*.

4.6 Pendeteksian Korosi

Sistem akan dicoba untuk mendeteksi korosi yang terdapat pada Jembatan Petekan Surabaya yang sudah direkam menggunakan *drone* DJI Mavic Air 2. Ketika sistem mendeteksi objek berupa, model akan mendeteksi *frame* setiap detiknya dan menampilkan hasil deteksi berupa *bounding box*. Uji coba model akan menggunakan bahasa pemrograman *python* dengan *script* yang terdapat pada Lampiran 7.

BAB V
HASIL DAN PEMBAHASAN

5.1 Model Hasil *Training*

Dalam melakukan *training* model yang akan dipakai untuk pendeteksian korosi, dilakukan variasi *hyperparameter* untuk mencari *setting* yang terbaik agar didapatkan nilai minimum *loss* yang paling kecil. Variasi *hyperparameter* yang dilakukan adalah nilai *batch size* dan *learning rate*. Pemilihan nilai *batch size* dapat ditentukan mulai dari satu dan seterusnya. Pada umumnya, nilai *batch size* dipilih lebih dari satu menggunakan bilangan terdekat dari pangkat dua. Pada penelitian ini digunakan nilai 4 dan 8. Nilai *batch size* yang besar dapat menyebabkan nilai total *loss* lama menuju konvergen karena data yang diproses saat satu kali iterasi (*step*) lebih banyak. Sedangkan pemilihan nilai *learning rate* dapat ditentukan dengan *range* nilai 0 – 1 yang ditentukan secara manual karena tidak ada metode yang baku dalam menentukan nilai yang paling baik. Pada penelitian ini digunakan 0.001 dan 0.01. Untuk dapat mencapai kondisi konvergen, pemilihan *learning rate* sangat berpengaruh dalam pembelajaran, jika *learning rate* terlalu kecil maka membutuhkan waktu lama untuk mendekati nilai total *loss* minimum. Nilai *batch size* dan *learning rate* merupakan parameter yang mempengaruhi dalam kecepatan proses *training*. Semakin besar nilai *batch size* dan semakin kecil *learning rate*, maka akan membutuhkan waktu lama dalam proses *training*.

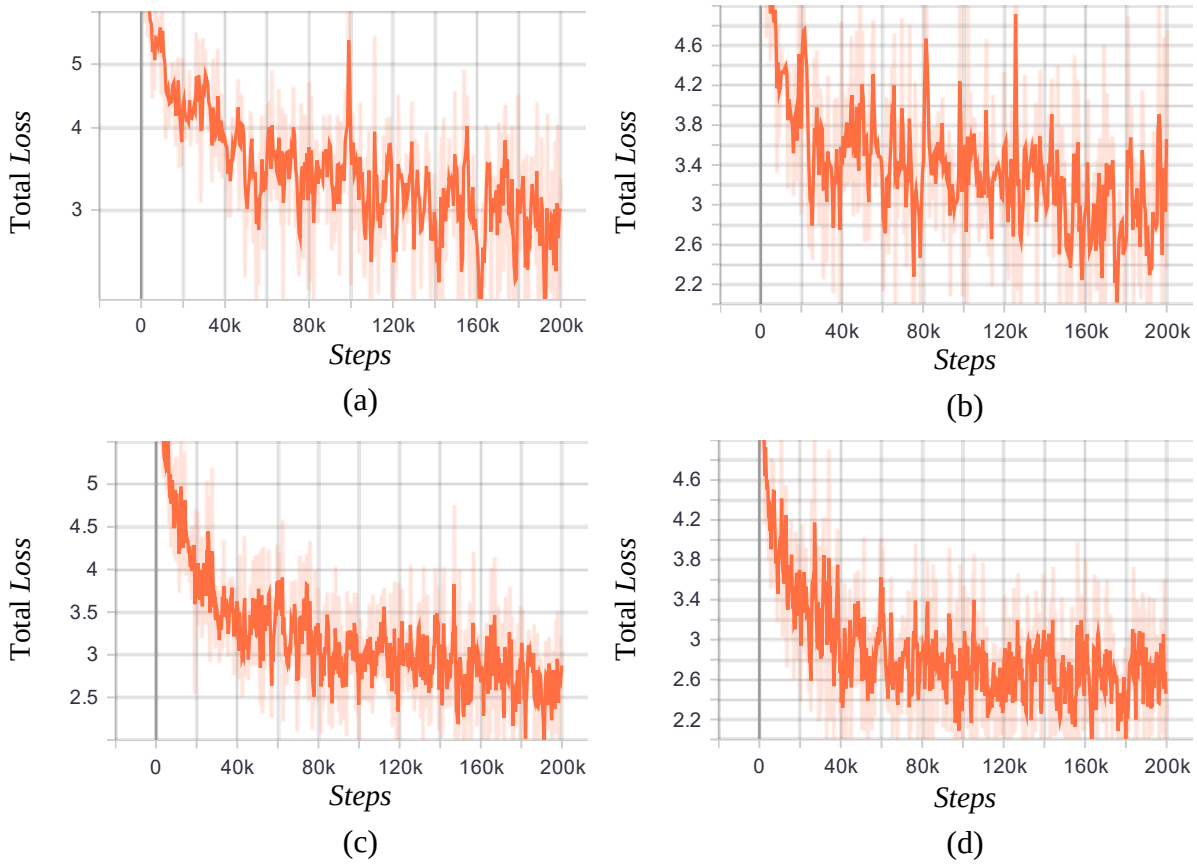
Nilai total *loss* merupakan salah satu parameter yang merepresentasikan atau menentukan performa dari model dalam akurasinya mendeteksi objek. Nilai total *loss* berfungsi untuk memberikan informasi dari total kerugian dari hasil pelatihan model yang dihasilkan pada saat awal proses *training* sampai dengan selesai sesuai dengan jumlah iterasi yang dilakukan yaitu sebanyak 200000 *steps*. Nilai minimum total *loss* yang terjadi pada setiap variasi dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil Pengujian *Hyperparameter*

Nilai Total *Loss* Variasi *Hyperparameter*			
Batch Size	*Learning Rate*	Total Waktu *Training*	Minimum Total *Loss*
4	0.001	25 jam 50 menit 17 detik	1.647
	0.01	25 jam 1 menit 58 detik	1.389
8	0.001	50 jam 52 menit 41 detik	1.673
	0.01	49 jam 1 menit 38 detik	1.561

Adapun *stop condition* pada penelitian ini adalah jumlah iterasi yaitu 200000 *steps*, karena nilai total *loss* pada saat proses *training* mulai konvergen dan tidak ada perubahan yang signifikan setelah iterasi 160000 *steps*, sehingga nilai 200000 *steps* dianggap cukup sebagai *threshold*. Berdasarkan Tabel 4.1, dengan jumlah iterasi 200000 *steps*, nilai minimum total *loss* terendah terdapat pada variasi nilai *batch size* 4 dan *learning rate* 0.01 dengan nilai 1.389, sedangkan nilai minimum total *loss* tertinggi terdapat pada varisi nilai *batch size* 8 dan nilai *learning rate* 0.001 dengan nilai 1.673.

Pada Gambar 5.1 ditunjukkan grafik dari total *loss* yang menunjukkan adanya penurunan nilai total *loss* namun tidak secara signifikan. Dari awal proses *training* dapat diketahui bahwa nilai total *loss* sangat tinggi. Namun, semakin banyak iterasi membuat nilai total *loss* mulai menurun. Pada grafik juga dapat dilihat bahwa semakin besar nilai *batch size*, maka nilai total *loss* yang didapatkan semakin besar dan juga semakin kecil nilai *learning rate*, maka nilai total *loss* yang didapatkan semakin besar.



Gambar 5.1 Grafik Total *Loss* dengan *Batch Size* dan *Learning Rate* (a) 4 – 0.001 (b) 4 – 0.01 (c) 8 – 0.001 (d) 8 – 0.01

5.2 Hasil Deteksi

Untuk menjawab tujuan penelitian, dilakukan pengujian deteksi korosi dengan variasi *hyperparameter batch size* 4 dan 8 dengan *learning rate* 0.001 dan 0.01 serta variasi jarak *drone* dengan objek 1 dan 2 m serta kecepatan *drone* 0.6 m/s; 0.9 m/s; dan 1.3 m/s. Pengujian dilakukan empat kali mulai dari *batch size* 4 dengan *learning rate* 0.001 dan 0.01 kemudian dilanjutkan dengan *batch size* 8 dengan *learning rate* 0.001 dan 0.01.

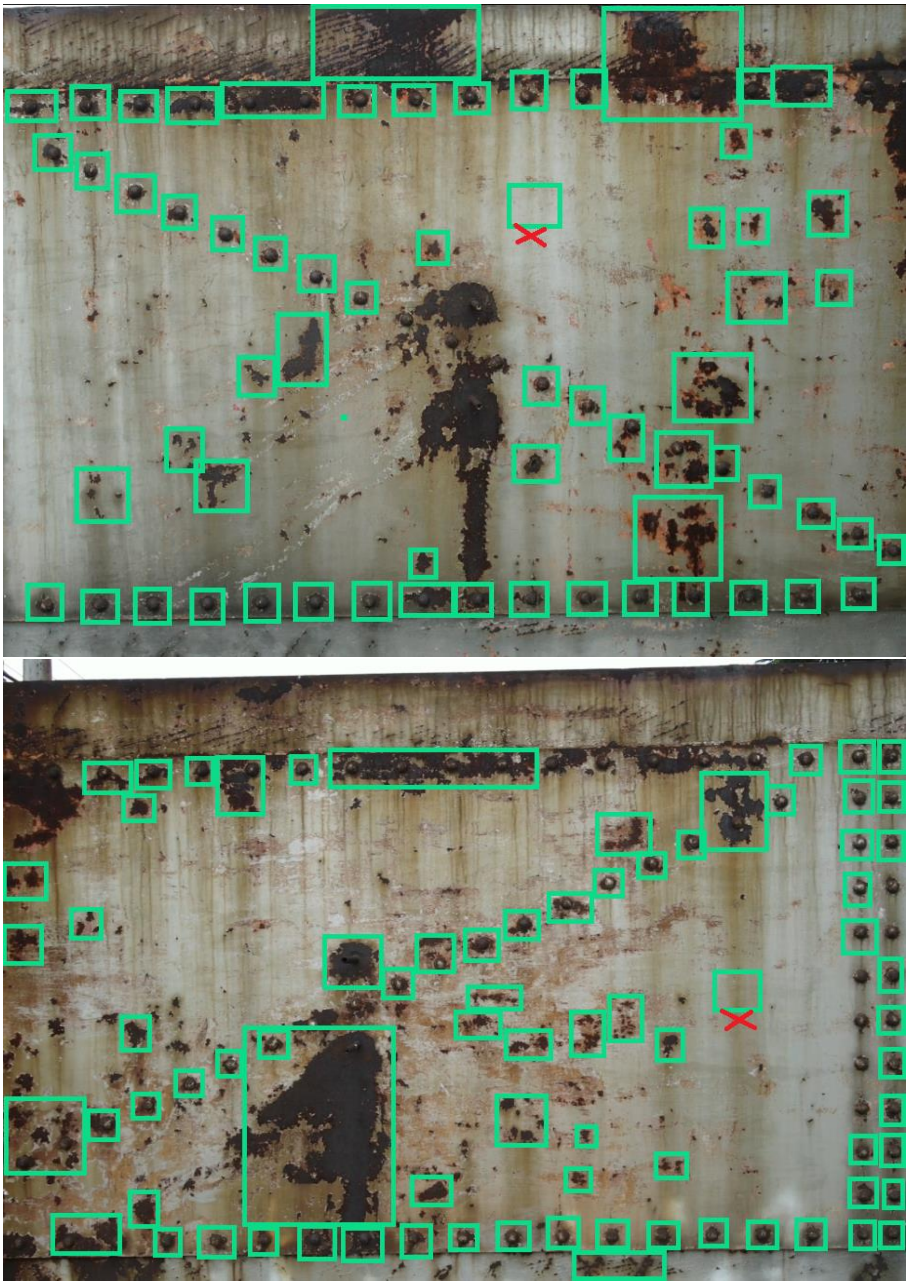
Nilai akurasi deteksi korosi dihitung pada keseluruhan *frame* pada saat melakukan pendeteksian objek. Untuk menghitung nilai akurasi, digunakan tabel *confusion matrix* yang terdapat pada Tabel 2.2 serta persamaan 5.1.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \; x \; 100\% \qquad (5.1)$$

Perhitungan akurasi pada persamaan 5.1 dapat dijelaskan sebagai berikut :

1. *True Positive* (TP) adalah jumlah korosi yang terdeteksi oleh model.
2. *False Negative* (FN) adalah jumlah korosi yang tidak terdeteksi oleh model.
3. *False Positive* (FP) adalah jumlah bukan korosi yang terdeteksi sebagai korosi oleh model.
4. *True Negative* (TN) adalah jumlah bukan korosi yang tidak terdeteksi oleh model.

Gambar 5.2 berikut merupakan contoh perhitungan nilai akurasi pada seluruh frame video dari pendeteksian korosi yang dilakukan dengan jarak *drone* terhadap objek 1 meter,



Gambar 5.2 Perhitungan Nilai Akurasi

Dari bagian jembatan yang dilakukan pengujian sudah ditentukan bahwa terdapat 161 jumlah korosi, kemudian dilakukan pendeteksian seperti pada Gambar 5.2. Dapat dilihat bahwa terdapat 139 *True Positive* (TP) yang ditandai oleh sistem dengan *bounding box* berwarna hijau, 22 *False Negative* (FN) yang didapat dari pengurangan nilai aktual dengan *True Positive* (TP), 2 *False Positive* (FP) yang ditandai oleh sistem dengan *bounding box* berwarna hijau dengan tambahan tanda silang yang penulis tandai, dan 0 *True Negative* (TN). Dengan persamaan 5.1 dapat dihitung nilai akurasi sebagai berikut:

$$Akurasi = \frac{139}{139 + 22 + 2}\ x\ 100\% = 85.27\%$$

Dari hasil perhitungan, didapatkan nilai akurasi sebesar 85.27%. Dengan cara tersebut, maka dapat dihitung nilai akurasi dari seluruh variasi parameter yang ada.

5.2.1 Hasil Uji Deteksi Korosi dengan *Batch Size* 4

Tabel 5.1 Hasil Pengujian pada *Batch Size* 4

Nilai Akurasi dengan *Batch Size* 4							
No	*Learning Rate*	Kecepatan	Jarak	TP	FN	FP	Akurasi
1	0.001	0.6 m/s	1 m	129	32	2	79.14%
2			2 m	100	61	4	60.61%
3		0.9 m/s	1 m	122	39	3	74.39%
4			2 m	92	69	5	55.42%
5		1.3 m/s	1 m	115	52	5	69.28%
6			2 m	89	72	6	53.29%
7	0.01	0.6 m/s	1 m	111	50	2	68.10%
8			2 m	72	89	4	43.64%
9		0.9 m/s	1 m	105	56	4	63.64%
10			2 m	68	93	6	40.72%
11		1.3 m/s	1 m	100	61	5	60.24%
12			2 m	64	97	7	38.10%

Hasil uji dengan variasi *hyperparameter batch size* 4 dengan *learning rate* 0.001 dan 0.01 pada Tabel 5.1 menunjukkan bahwa bahwa nilai akurasi cenderung turun seiring bertambahnya jarak dan kecepatan. Pada *batch size* 4, pengujian dengan jarak *drone* dengan objek dua kali lebih dekat, menghasilkan peningkatan akurasi sebesar 15 – 25%. Sedangkan pengujian dengan kecepatan *drone* yang lebih rendah (sekitar 1.5% lebih kecil) menghasilkan peningkatan 3 – 6% pada pengujian dengan jarak 1 m dan 2 – 5% pada pengujian dengan jarak 2 m.

5.2.2 Hasil Uji Deteksi Korosi dengan *Batch Size* 8

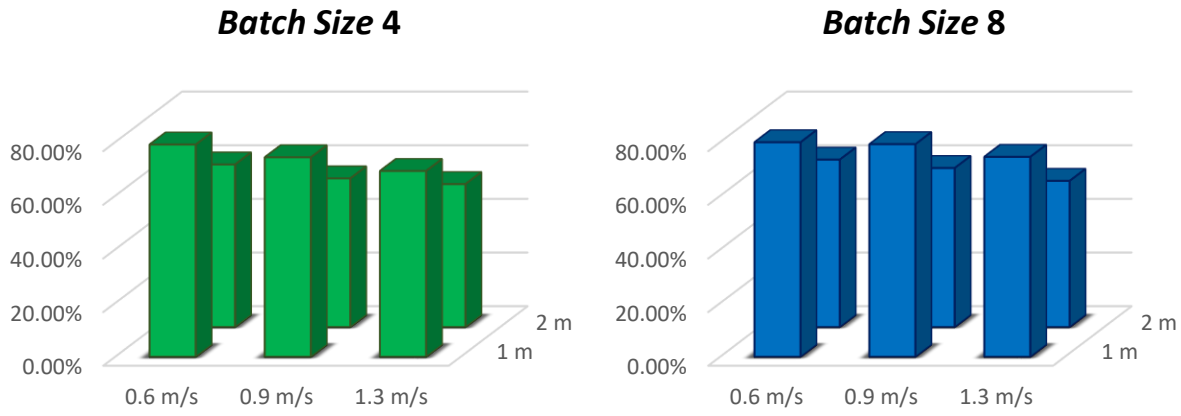
Tabel 5.2 Hasil Pengujian pada *Batch Size* 8

Nilai Akurasi dengan *Batch Size* 8							
No	*Learning Rate*	Kecepatan	Jarak	TP	FN	FP	Akurasi
1	0.001	0.6 m/s	1 m	138	23	2	84.66%
2			2 m	103	58	4	62.42%
3		0.9 m/s	1 m	130	31	3	79.27%
4			2 m	99	62	6	59.28%
5		1.3 m/s	1 m	123	38	4	74.55%
6			2 m	91	70	6	54.49%
7	0.01	0.6 m/s	1 m	124	37	2	76.07%
8			2 m	85	76	5	51.20%
9		0.9 m/s	1 m	113	48	4	68.48%
10			2 m	79	82	5	47.59%
11		1.3 m/s	1 m	108	53	5	65.06%
12			2 m	72	89	6	43.11%

Seperti halnya hasil pengujian dengan *batch size* 4, hasil uji dengan variasi *hyperparameter batch size* 8 dengan *learning rate* 0.001 dan 0.01 pada Tabel 5.2 menunjukkan bahwa nilai akurasi cenderung turun seiring bertambahnya jarak dan kecepatan. Pada *batch size* 8, pengujian dengan jarak *drone* dengan objek dua kali lebih dekat, menghasilkan peningkatan

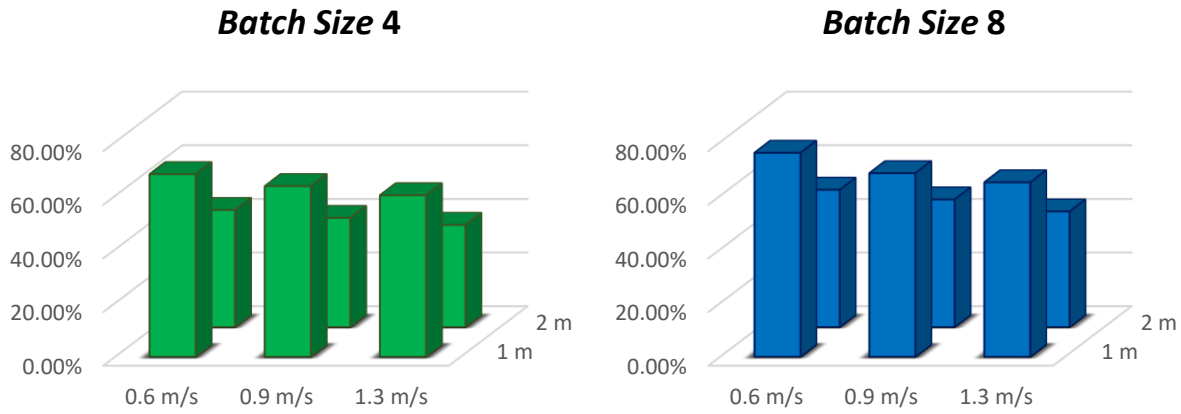
akurasi sebesar 19 – 25%. Sedangkan pengujian dengan kecepatan *drone* yang lebih rendah (sekitar 1.5% lebih kecil) menghasilkan peningkatan 3 – 8% pada pengujian dengan jarak 1 m dan 3 – 5% pada pengujian dengan jarak 2 m.

5.2.3 Perbandingan Nilai *Batch Size* dan *Learning Rate*



(a) (b)

Gambar 5.3 Grafik *Learning Rate* 0.001 pada (a) *Batch Size* 4 (b) *Batch Size* 8



(a) (b)

Gambar 5.4 Grafik *Learning Rate* 0.01 pada (a) *Batch Size* 4 (b) *Batch Size* 8

Pada Gambar 5.3 dan Gambar 5.4 dapat dilihat bahwa nilai *batch size* mempengaruhi nilai akurasi pendeteksian objek. Semakin besar nilai *batch size*, maka akurasinya akan semakin meningkat. Pengujian dengan *learning rate* 0.001, menggunakan *batch size* 8 jika dibandingkan dengan *batch size* 4 mempunyai selisih akurasi sebesar 1 – 6%. Pada *learning rate* 0.001, nilai akurasi paling tinggi didapat dari nilai *batch size* 8 dengan nilai akurasi sebesar 84.66%, sedangkan nilai akurasi paling rendah didapat dari nilai *batch size* 4 dengan nilai akurasi sebesar 53.29%. Sedangkan pengujian dengan *learning rate* 0.01, menggunakan *batch size* 8 jika dibandingkan dengan *batch size* 4 mempunyai selisih akurasi sebesar 4 – 7%. Pada *learning rate* 0.01, nilai akurasi paling tinggi didapat dari nilai *batch size* 8 dengan nilai akurasi sebesar 76.07%, sedangkan nilai akurasi paling rendah didapat dari nilai *batch size* 4 dengan nilai akurasi sebesar 38.10%.

Pada Gambar 5.3 dan Gambar 5.4 juga dapat dilihat bahwa nilai *learning rate* mempengaruhi nilai akurasi pendeteksian objek. Semakin kecil nilai *learning rate*, maka akurasinya akan semakin meningkat. Pengujian dengan *batch size* 4, menggunakan *learning rate* 0.001 jika dibandingkan dengan *learning rate* 0.01 mempunyai selisih akurasi sebesar 9 – 17%. Pada *batch size* 4, nilai akurasi paling tinggi didapat dari nilai *learning rate* 0.001 dengan nilai akurasi sebesar 79.14%, sedangkan nilai akurasi paling rendah didapat dari nilai *learning rate* 0.01 dengan nilai akurasi sebesar 38.10%. Sedangkan pengujian dengan *batch size* 8, menggunakan *learning rate* 0.001 jika dibandingkan dengan *learning rate* 0.01 mempunyai selisih akurasi sebesar 8 – 12%. Pada *batch size* 8, nilai akurasi paling tinggi didapat dari nilai *learning rate* 0.001 dengan nilai akurasi sebesar 84.66%, sedangkan nilai akurasi paling rendah didapat dari nilai *learning rate* 0.01 dengan nilai akurasi sebesar 43.11%.

5.3 Analisis Hasil

Mengacu pada penelitian-penelitian terdahulu yang pernah dilakukan, penelitian serupa pernah dilakukan oleh Taufik Reza Nurdiasnyah yang bertujuan untuk mendeteksi korosi pada struktur kapal menggunakan metode *Convolutional Neural Network.* Pada penelitian tersebut digunakan model pralatih SSD MobileNet V1 untuk mendeteksi tiga tipe korosi. Dengan data set sebesar 138 citra didapatkan nilai akurasi sebesar 63% dengan nilai total *loss* sebesar 0.440. Selain penelitian yang telah dilakukan Taufik Reza Nurdiasnyah, terdapat penelitian lain yang dilakukan oleh Achilleas Matthaiou. Pada penelitian tersebut juga dilakukan untuk mendeteksi korosi menggunakan metode *Convolutional Neural Network*. Pada penelitian tersebut digunakan model pralatih SSD MobileNet V1 untuk mendeteksi korosi. Dengan data set sebesar 155 citra didapatkan nilai akurasi sebesar 85% dengan nilai total *loss* sebesar 1.7931.

Sama halnya dengan apa yang telah dikerjakan Taufik Reza Nurdiasnyah dan Achilleas Matthaiou, pada penelitian ini dilakukan pengujian deteksi korosi pada material baja menggunakan metode *Convolutional Neural Network* dengan data pralatih SSD MobileNet V1 yang memanfaatkan kamera *drone*. Dengan menggunakan nilai akurasi yang diperoleh oleh Taufik Reza Nurdiasnyah, penelitian ini mencoba untuk mencari variasi dari *hyperparameter* nilai *learning rate* dan *batch size* serta variasi jarak dan kecepatan *drone* untuk memperoleh nilai akurasi lebih dari 63%. Didapatkan bahwa nilai akurasi pada penelitian ini adalah 84.66% dengan nilai minimum total *loss* sebesar 1.673 yang didapat menggunaka iterasi 200000 *steps* dengan variasi *hyperparameter batch size* sebesar 4 dan *learning rate* sebesar 0.001 dengan jarak 1 m dan kecepatan 0.6 m/s.. Model dilatih menggunakan data set sebesar 200 citra dibagi menjadi 170 citra untuk *training* dan 30 citra untuk *testing*. Penentuan nilai dari *batch size*, dan *learning rate* dapat mempengaruhi nilai akurasi, yaitu semakin besar *batch size* dan semakin kecil *learning rate*, maka nilai akurasi semakin tinggi. Namun, memperbesar nilai *batch size* dan memperkecil nilai *learning rate* dapat menyebabkan bertambahnya waktu yang diperlukan dalam proses *training*. Jarak dan kecepatan juga mempengaruhi nilai akurasi dari pendeteksian objek. Semakin dekat objek dan lambat kecepatan dari *drone*, maka akurasi yang didapat akan semakin tinggi.

BAB VI
KESIMPULAN

6.1 Kesimpulan

Setelah dilakukannya penelitian ini, diperoleh beberapa poin yang dapat disimpulkan dari hasil yang di dapat yaitu sebagai berikut:

1. Implementasi *Convolutional Neural Network* (CNN) dapat dilakukan untuk mendeteksi korosi pada material baja dengan kamera *drone* yang dihasilkan dari model pralatih SSD Mobilenet V1 yang tersedia secara *default* pada Tensorflow *Object Detection* API.
2. Jarak dan kecepatan *drone* mempengaruhi nilai akurasi pada pendeteksian korosi. Nilai akurasi cenderung turun seiring bertambahnya jarak dan kecepatan. Variasi yang menghasilkan nilai akurasi yang paling baik didapat pada dengan jarak 1 m dengan kecepatan 0.6 m/s dengan nilai akurasi 84.66%.
3. *Setting* parameter *Convolutional Neural Network* (CNN) pada penelitian yang paling baik sehingga mendapatkan akurasi yang paling tinggi yaitu dengan data set 200 citra menggunakan iterasi sebanyak 200000 *steps* dengan nilai *batch size* 8 dan *learning rate* 0.001 yang menghasilkan minimum total *loss* sebesar 1.673.

6.2 Saran

Dengan selesainya penelitian ini, diperoleh beberapa saran guna pengembangan lanjutan dari laporan ini. Adapun saran dari penelitian ini adalah sebagai berikut:

1. Dalam pengumpulan dataset, peneliti selanjutnya perlu mengumpulkan dan menggunakan gambar dari segala arah dan jarak, yang mana akan digunakan dalam proses *training*, sehingga dapat menghasilkan akurasi yang tinggi.
2. Perhitungan jumlah korosi yang terdeteksi masih dilakukan secara manual sehingga memerlukan waktu yang lama, oleh karena itu perlu ditambahkan sistem yang dapat menghitung jumlah korosi yang terdeteksi.
3. Dalam melakukan *training* data lebih baik menggunakan spesifikasi perangkat keras seperti *Random Access Memory* (RAM) yang tinggi dan menggunakan *Graphics Processing Unit* (GPU) dengan performa tinggi, sehingga dapat mempercepat dan menjalankan kalkulasi dengan jumlah *input* citra yang lebih banyak dan resolusi yang tinggi.

DAFTAR PUSTAKA

Aini, N. (2016). Perilaku Korosi Baja AISI 1021 dan AISI 304 dalam Berbagai Media Asam. Surabaya: Institut Teknologi Sepuluh Nopember.

Akbar, K. I. (2018). Analisis Akurasi Horizontal Peta Ortofoto Skala 1:1000 Menggunakan DJI Mavic Pro. Surabaya: Institut Teknologi Sepuluh Nopember.

Akbari, M. I. (2021). Perancangan Subsistem Antarmuka dan Deteksi Retakan pada Investigasi Visual Kondisi Struktur Jembatan Menggunakan Drone Otonom.

Dewi, S. R. (2018). Deep Learning Object Detection pada Video Menggunakan Tensorflow dan Convolutional Neural Network.

DJI. (2020). *Mavic Air 2*. Retrieved March 27, 2022, from https://www.dji.com/id/mavic-air-2

Eisenbeiss, H. (2009). *UAV Photogrammetry*. ETH Zürich.

Fontana, M. G. (1986). *Corrosion Engineering* (3rd ed.). New York: McGraw-Hill.

Geitgey, A. (2016). *Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks*. Retrieved January 14, 2022, from https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). CA 95472: O'Reilly Media, Inc.

Jalled, F., & Voronkov, I. (2016). *Object Detection Using Image Processing.* Ithaca, New York: Cornell University Library.

Lina, Q. (2019). *Apa itu Convolutional Neural Network?* Retrieved March 1, 2022, from medium: https://medium.com/@16611110/apa-itu-convolutional-neural-network-836f70b193a4

Liu, W. (2016). SSD: Single Shot MultiBox Detector. *European Conference on Computer Vision*. doi:10.48550/arXiv.1512.02325

Mashita, S. N. (2020). *Implementasi Deep Learning Object Detection Rambu K3 pada Video Menggunakan Metode Convolutional Neural Network (CNN) dengan Tensorflow.* Yogyakarta: Universitas Islam Indonesia.

Matthaiou, A. (2021). *Corrosion Detection with Computer Vision and Deep Learning.* School of Naval Architecture and Marine Engineering National Technical University of Athens.

Ng, A. (2017). *Why Do You Need Non-Linear Activation Functions?* Retrieved March 20, 2022, from coursera: https://www.coursera.org/lecture/neural-networks-deep-learning/why-do-you-need-non-linear-activation-functions-OASKH

Nurdiansyah, T. R. (2021). *Pengaplikasian Convolutional Neural Network sebagai Metode Pendeteksi Korosi pada Struktur Kapal.* Surabaya: Institut Teknologi Sepuluh Nopember.

Putri, A. S., Setyawan, G. E., & Tibyani. (2018, September). Sistem Deteksi Warna pada Quadcopter Ar.Drone Menggunakan Metode Color Filtering Hue Saturation and

Value (HSV). *Jurnal Pengembangan Teknlogi Informasi dan Ilmu Komputer (J-PTIIK) , 2*, 3202-3207.

Sanjana, B. (2020). *SSD MobileNetV1 Architecture*. Retrieved July 8, 2022, from OpenGenus: https://iq.opengenus.org/ssd-mobilenet-v1-architecture/

Sena, S. (2017). *Pengenalan Deep Learning Part 7: Convolutional Neural Network (CNN)*. Retrieved February 22, 2022, from https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94

Tempo. (2018). *tempo.co*. (E. Y. Saputra, Editor) Retrieved February 4, 2022, from https://dunia.tempo.co/read/1119184/sebelum-roboh-ahli-sebut-kabel-jembatan-genoa-italia-berkarat

Tian, Z., & Zhang, G. (2020). *Corrosion Identification of Fittings Based on Computer Vision.* Dubin: IEEE. doi:10.1109/AIAM48774.2019.00123

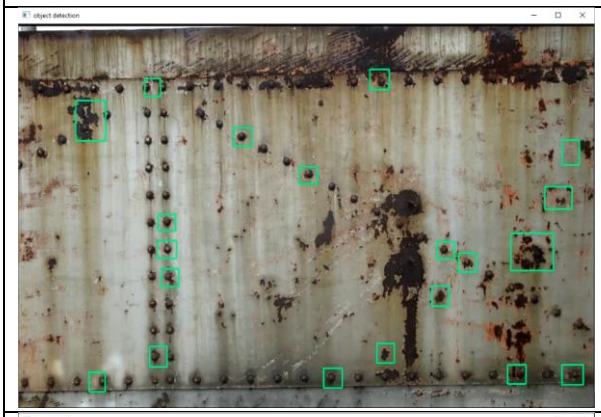
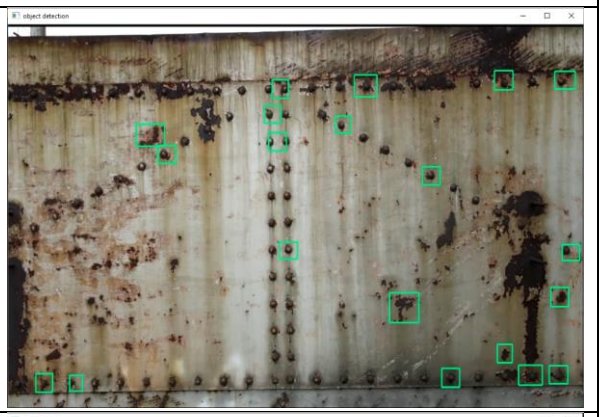
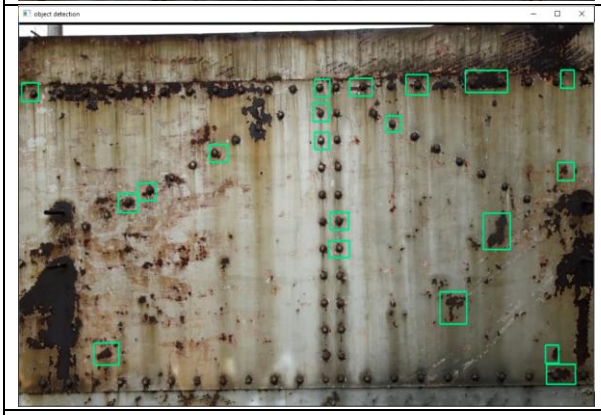
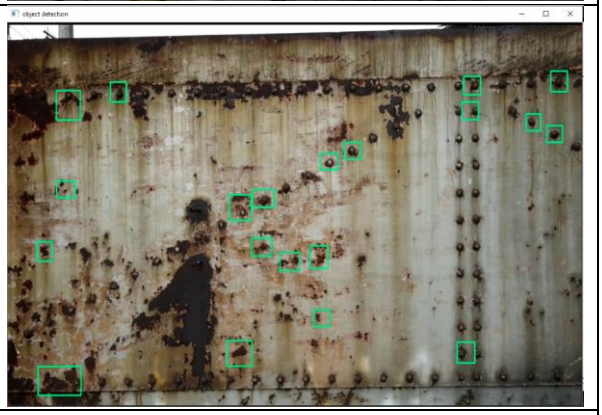
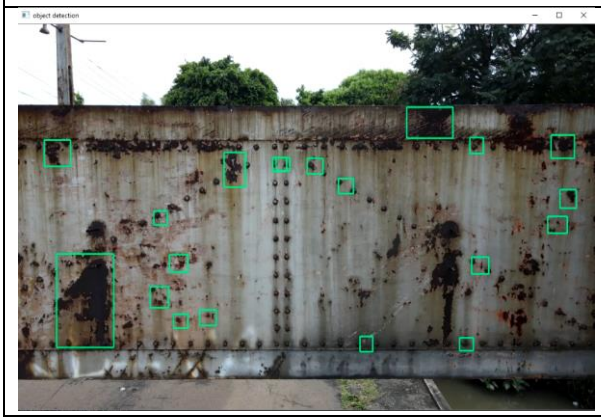
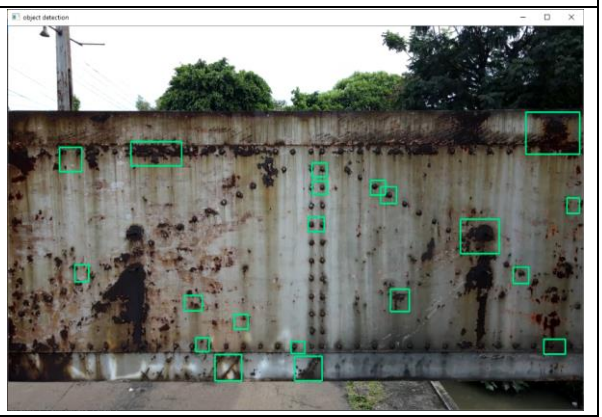
Verma, C. (2017). Corrosion Inhibitors for Ferrous and Non-Ferrous Metals and Alloys in Ionic Sodium Chloride Solutions: A Review. *Journal of Molecular Liquids*, 927–942.

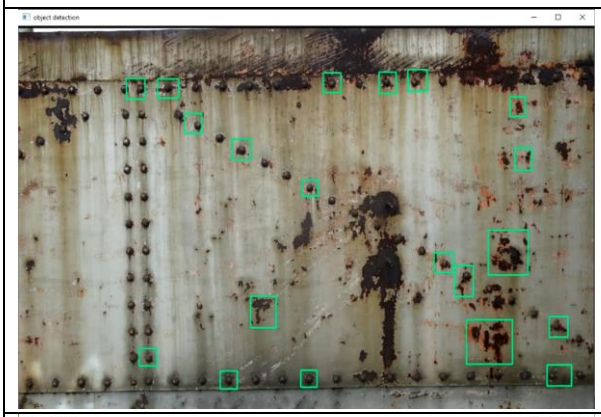
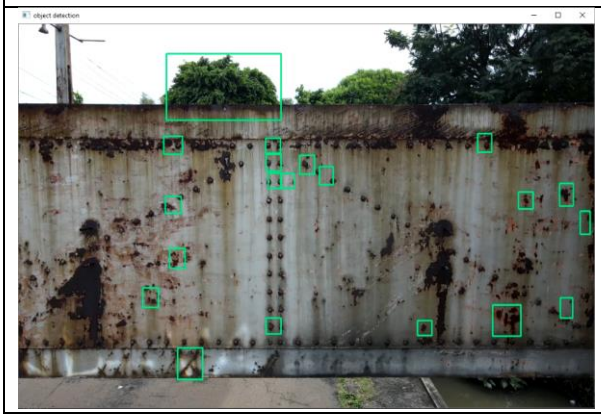
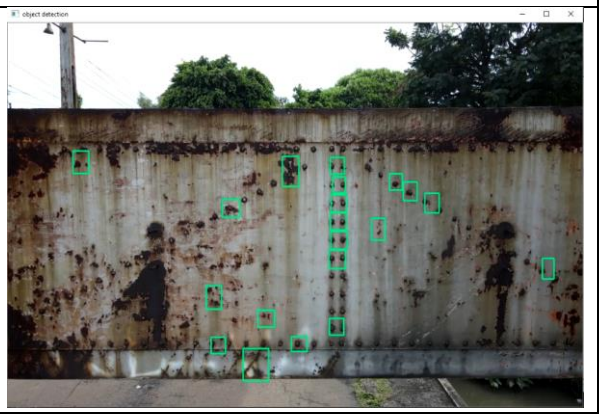
William D Callister, J., & Rethwisch, D. G. (2019). *Materials Science and Engineering: An Introduction* (8th ed.). Hoboken, NJ : John Wiley & Sons: John Wiley and Sons.

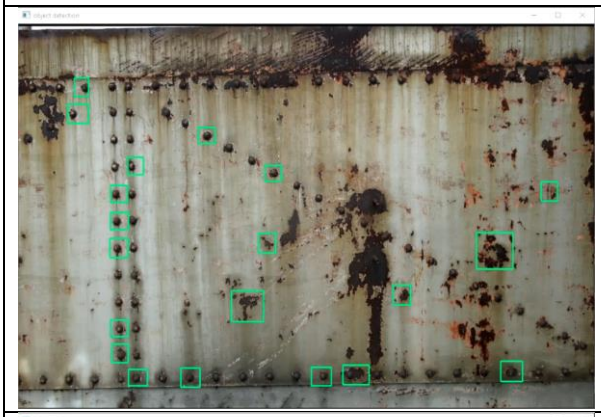
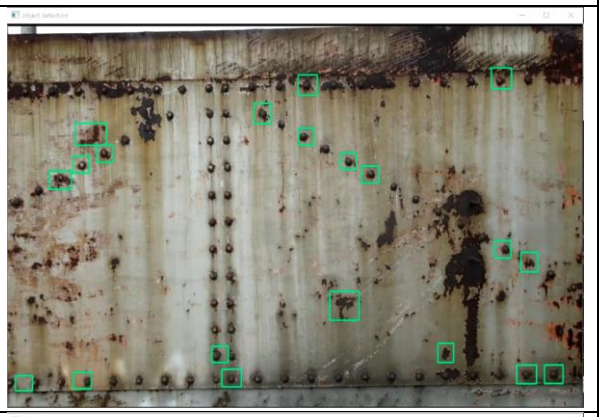
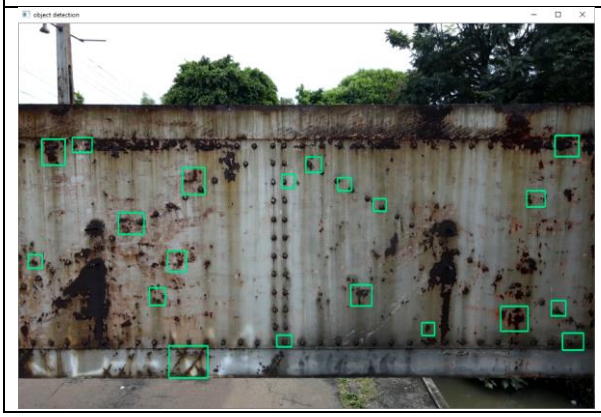
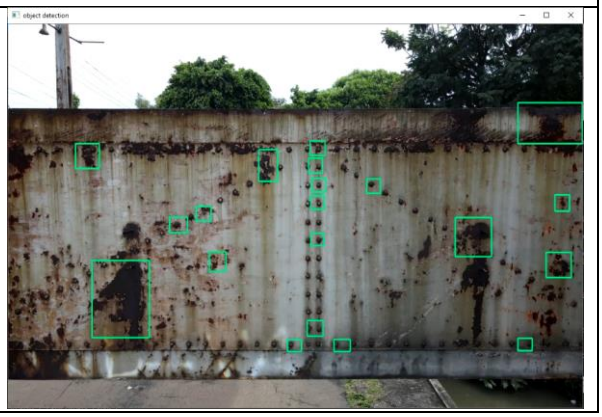
Zakharov, B. (1962). *Heat Treatment of Metal.* Moscow: Peace Publishers.

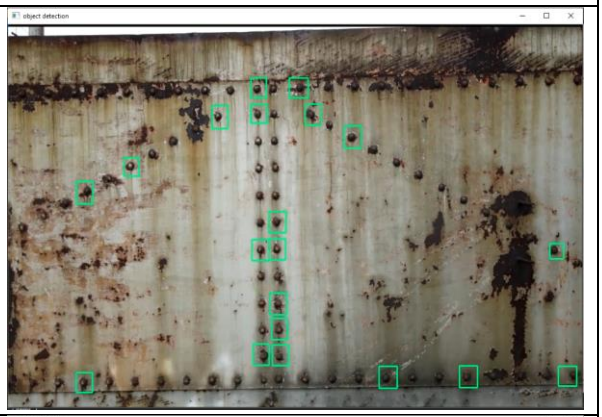
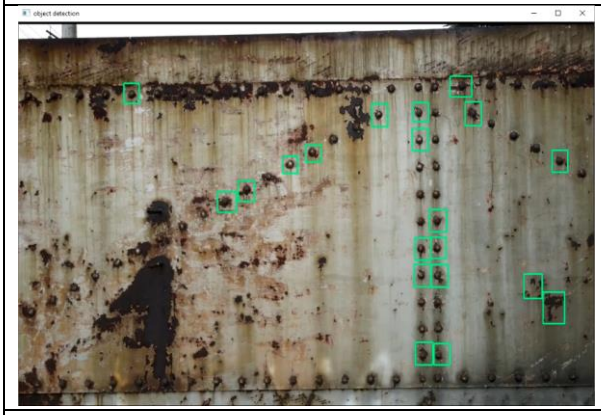
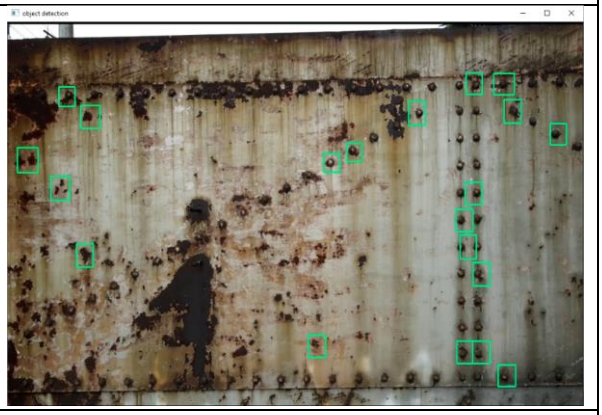
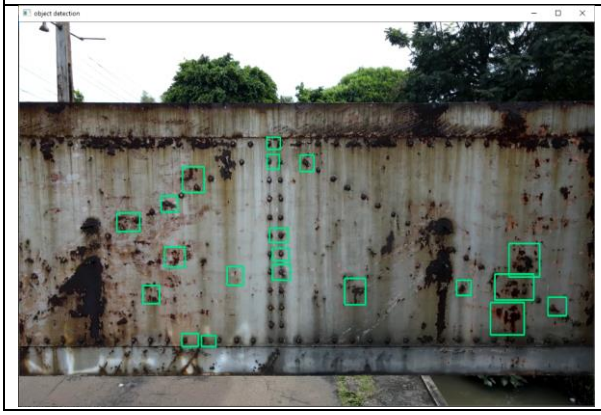
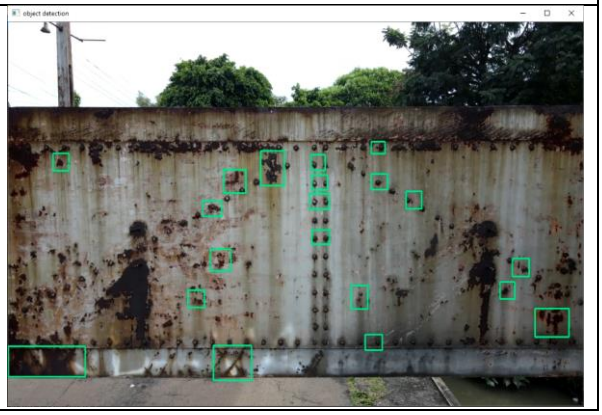
LAMPIRAN

Lampiran 1 Hasil Pengujian Deteksi Korosi

Batch Size 4 dengan *Learning Rate* 0.001	
Jarak 1 meter	
	
	
Jarak 2 meter	
	

Batch Size 4 dengan *Learning Rate* 0.01	
Jarak 1 meter	
	
	
Jarak 2 meter	
	

Batch Size 8 dengan *Learning Rate* 0.001	
Jarak 1 meter	
	
	
Jarak 2 meter	
	

Batch Size 8 dengan *Learning Rate* 0.01	
Jarak 1 meter	
	
	
Jarak 2 meter	
	

Lampiran 2 Hasil Perhitungan *Convolutional Layer*

Posisi Kernel	Perhitungan Pada Setiap Sel	*Output*
1	$(0 \times 1) + (1 \times 0) + \big(2 \times (-1)\big) + (5 \times 1) + (8 \times 0) + \big(9 \times (-1)\big) + (7 \times 1) + (2 \times 0) + \big(5 \times (-1)\big)$	-4
2	$(1 \times 1) + (2 \times 0) + \big(9 \times (-1)\big) + (8 \times 1) + (9 \times 0) + \big(0 \times (-1)\big) + (2 \times 1) + (5 \times 0) + \big(8 \times (-1)\big)$	-6
3	$(2 \times 1) + (9 \times 0) + \big(8 \times (-1)\big) + (9 \times 1) + (0 \times 0) + \big(2 \times (-1)\big) + (5 \times 1) + (8 \times 0) + \big(5 \times (-1)\big)$	1
4	$(5 \times 1) + (8 \times 0) + \big(9 \times (-1)\big) + (7 \times 1) + (2 \times 0) + \big(5 \times (-1)\big) + (6 \times 1) + (2 \times 0) + \big(8 \times (-1)\big)$	-4
5	$(8 \times 1) + (9 \times 0) + \big(0 \times (-1)\big) + (2 \times 1) + (5 \times 0) + \big(8 \times (-1)\big) + (2 \times 1) + (8 \times 0) + \big(2 \times (-1)\big)$	2
6	$(9 \times 1) + (0 \times 0) + \big(2 \times (-1)\big) + (5 \times 1) + (8 \times 0) + \big(5 \times (-1)\big) + (8 \times 1) + (2 \times 0) + \big(7 \times (-1)\big)$	8
7	$(7 \times 1) + (2 \times 0) + \big(5 \times (-1)\big) + (6 \times 1) + (2 \times 0) + \big(8 \times (-1)\big) + (1 \times 1) + (7 \times 0) + \big(8 \times (-1)\big)$	-7
8	$(2 \times 1) + (5 \times 0) + \big(8 \times (-1)\big) + (2 \times 1) + (8 \times 0) + \big(2 \times (-1)\big) + (7 \times 1) + (8 \times 0) + \big(5 \times (-1)\big)$	-4
9	$(5 \times 1) + (8 \times 0) + \big(5 \times (-1)\big) + (8 \times 1) + (2 \times 0) + \big(7 \times (-1)\big) + (8 \times 1) + (5 \times 0) + \big(1 \times (-1)\big)$	8

Lampiran 3 *Script* Augmentasi Data

```
import os
import random
from scipy import ndarray

import skimage as sk
from skimage import transform
from skimage import util
from skimage import io

def random_noise(image_array: ndarray):
    return sk.util.random_noise(image_array)

def horizontal_flip(image_array: ndarray):
    return image_array[:, ::-1]

def vertical_flip(image_array: ndarray):
    return image_array[::-1, :]

available_transformations = {
    'noise': random_noise,
    'horizontal_flip': horizontal_flip,
    'vertical_flip': vertical_flip
}

folder_path = 'rust'
num_files_desired = 6

images = [os.path.join(folder_path, f) for f in
os.listdir(folder_path) if
os.path.isfile(os.path.join(folder_path, f))]

num_generated_files = 0
while num_generated_files <= num_files_desired:
    image_path = random.choice(images)
    image_to_transform = sk.io.imread(image_path)
    num_transformations_to_apply = random.randint(1,
len(available_transformations))

    num_transformations = 0
    transformed_image = None
    while num_transformations <= num_transformations_to_apply:
        key = random.choice(list(available_transformations))
        transformed_image =
available_transformations[key](image_to_transform)
        num_transformations += 1

        new_file_path = '%s/augmentasi_%s.jpg' % (folder_path,
num_generated_files)

        io.imsave(new_file_path, transformed_image)
    num_generated_files += 1
```

Lampiran 4 *Script* Konversi Dataset XML ke CSV

```
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET


def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text)
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class', 'xmin',
'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def main():
    for folder in ['train', 'test']:
        image_path = os.path.join(os.getcwd(), ('images/' +
folder))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv(('images/'+folder+'_labels.csv'),
index=None)
        print('Successfully converted xml to csv.')


main()
```

Lampiran 5 *Script* Konversi Dataset CSV ke TFRecord

```
from __future__ import division
from __future__ import print_function
from __future__ import absolute_import

import os
import io
import pandas as pd

from tensorflow.python.framework.versions import VERSION
if VERSION >= "2.0.0a0":
    import tensorflow.compat.v1 as tf
else:
    import tensorflow as tf

from PIL import Image
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict

flags = tf.app.flags
flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
flags.DEFINE_string('output_path', '', 'Path to output TFRecord')
flags.DEFINE_string('image_dir', '', 'Path to images')
FLAGS = flags.FLAGS


def class_text_to_int(row_label):
    if row_label == 'corroded':
        return 1
    else:
        return None


def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in
zip(gb.groups.keys(), gb.groups)]


def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path,
'{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
```

```
    classes_text = []
    classes = []

    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))

    tf_example =
tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin':
dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax':
dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin':
dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax':
dataset_util.float_list_feature(ymaxs),
        'image/object/class/text':
dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label':
dataset_util.int64_list_feature(classes),
    }))
    return tf_example


def main(_):
    writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(FLAGS.image_dir)
    examples = pd.read_csv(FLAGS.csv_input)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())

    writer.close()
    output_path = os.path.join(os.getcwd(), FLAGS.output_path)
    print('Successfully created the TFRecords:
{}'.format(output_path))


if __name__ == '__main__':
    tf.app.run()
```

Lampiran 6 Script Konfigurasi Pipeline

```
model {
  ssd {
    num_classes: 1
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
      }
    }
    similarity_calculator {
      iou_similarity {
      }
    }
    anchor_generator {
      ssd_anchor_generator {
        num_layers: 6
        min_scale: 0.2
        max_scale: 0.95
        aspect_ratios: 1.0
        aspect_ratios: 2.0
        aspect_ratios: 0.5
        aspect_ratios: 3.0
        aspect_ratios: 0.3333
      }
    }
    image_resizer {
      fixed_shape_resizer {
        height: 300
        width: 300
      }
    }
    box_predictor {
      convolutional_box_predictor {
        min_depth: 0
        max_depth: 0
        num_layers_before_predictor: 0
        use_dropout: false
        dropout_keep_probability: 0.8
        kernel_size: 1
        box_code_size: 4
        apply_sigmoid_to_scores: false
        conv_hyperparams {
```

```
          activation: RELU_6,
          regularizer {
            l2_regularizer {
              weight: 0.00004
            }
          }
          initializer {
            truncated_normal_initializer {
              stddev: 0.03
              mean: 0.0
            }
          }
          batch_norm {
            train: true,
            scale: true,
            center: true,
            decay: 0.9997,
            epsilon: 0.001,
          }
        }
      }
      feature_extractor {
        type: 'ssd_mobilenet_v1'
        min_depth: 16
        depth_multiplier: 1.0
        conv_hyperparams {
          activation: RELU_6,
          regularizer {
            l2_regularizer {
              weight: 0.00004
            }
          }
          initializer {
            truncated_normal_initializer {
              stddev: 0.03
              mean: 0.0
            }
          }
          batch_norm {
            train: true,
            scale: true,
            center: true,
            decay: 0.9997,
            epsilon: 0.001,
          }
        }
      }
      loss {
        classification_loss {
          weighted_sigmoid {
          }
        }
        localization_loss {
          weighted_smooth_l1 {
```

```
        }
      }
      hard_example_miner {
        num_hard_examples: 3000
        iou_threshold: 0.99
        loss_type: CLASSIFICATION
        max_negatives_per_positive: 3
        min_negatives_per_image: 0
      }
      classification_weight: 1.0
      localization_weight: 1.0
    }
    normalize_loss_by_num_matches: true
    post_processing {
      batch_non_max_suppression {
        score_threshold: 1e-8
        iou_threshold: 0.6
        max_detections_per_class: 100
        max_total_detections: 100
      }
      score_converter: SIGMOID
    }
  }
}

train_config: {
  batch_size: 2
  optimizer {
    rms_prop_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.001
          decay_steps: 800720
          decay_factor: 0.95
        }
      }
      momentum_optimizer_value: 0.9
      decay: 0.9
      epsilon: 1.0
    }
  }
  fine_tune_checkpoint:
"ssd_mobilenet_v1_coco_2018_01_28/model.ckpt"
  from_detection_checkpoint: true
  load_all_detection_checkpoint_vars: true
  num_steps: 200000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    ssd_random_crop {
    }
  }
}
```

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "data/train.record"
  }
  label_map_path: "data/object-detection.pbtxt"
}

eval_config: {
  metrics_set: "coco_detection_metrics"
  num_examples: 1100
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "data/test.record"
  }
  label_map_path: "training/object-detection.pbtxt"
  shuffle: false
  num_readers: 1
}
```

Lampiran 7 *Script* Deteksi Korosi

```
import numpy as np
import os
import cv2
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
tf.config.experimental.list_physical_devices('GPU')

from utils import label_map_util
from utils import visualization_utils as vis_util

import cv2

cap = cv2.VideoCapture("resources/0.6 - 1 m.mp4")

MODEL_NAME = 'new_graph'

PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

PATH_TO_LABELS = os.path.join('data', 'object-detection.pbtxt')

NUM_CLASSES = 1

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories =
label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape(
        (im_height, im_width, 3)).astype(np.uint8)

PATH_TO_TEST_IMAGES_DIR = 'test_images'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR,
'image{}.jpg'.format(i)) for i in range(1, 3) ]

IMAGE_SIZE = (12, 8)

with detection_graph.as_default():
    with tf.Session(graph=detection_graph) as sess:
        while True:
            ret, image_np = cap.read()

            image_np_expanded = np.expand_dims(image_np, axis=0)
```

```
            image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')

            boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')

            scores =
detection_graph.get_tensor_by_name('detection_scores:0')
            classes =
detection_graph.get_tensor_by_name('detection_classes:0')
            num_detections =
detection_graph.get_tensor_by_name('num_detections:0')

            (boxes, scores, classes, num_detections) =
sess.run([boxes, scores, classes,
num_detections],feed_dict={image_tensor: image_np_expanded})

            vis_util.visualize_boxes_and_labels_on_image_array(
                image_np,
                np.squeeze(boxes),
                np.squeeze(classes).astype(np.int32),
                np.squeeze(scores),
                category_index = category_index,
                use_normalized_coordinates=True,
                line_thickness=5,
                skip_labels=True,
                skip_scores=True)

            cv2.imshow('object detection', cv2.resize(image_np,
(1080, 720)))
            if cv2.waitKey(25) & 0xFF == ord('q'):
                cv2.destroyAllWindows()
                break
```

BIODATA PENULIS

Bara Atmaja lahir di Purworejo pada tanggal 14 Januari 1999, merupakan anak pertama dari dua bersaudara. Penulis telah menempuh pendidikan formal yaitu di SDIT Bakti Insani, SMPN 1 Sleman, dan SMAN 4 Yogyakarta. Setelah lulus dari SMA, Penulis melanjutkan pendidikan ke jenjang S-1 di Institut Teknologi Sepuluh Nopember (ITS) dengan bidang studi Teknik Mesin di Fakultas Teknologi Industri dan Rekayasa Sistem.

Sebagai mahasiswa pendidikan tinggi, penulis turut aktif dalam berbagai kegiatan akademik dan nonakademik. Beberapa pelatihan seperti LKMW TD, LKMM TD, dan PKTI TD, begitu juga dengan berbagai forum komunikasi ilmiah serta beberapa *short couse* untuk menunjang terselesaikannya Tugas Akhir ini.

Informasi, pertanyaan maupun saran mengenai Tugas Akhir ini dapat disampaikan kepada penulis melalui e-mail: atmajabara@gmail.com.