TUGAS AKHIR – SM141501

PENERAPAN METODE *SINGLE SHOT DETECTOR* (SSD) UNTUK KLASIFIKASI JENIS KERUSAKAN JALAN PADA DATA VIDEO

ROBERTUS DIAWAN CHRIS
NRP 06111440000066

Dosen Pembimbing
Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

DEPARTEMEN MATEMATIKA
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya 2020

FINAL PROJECT – SM141501

IMPLEMENTATION OF SINGLE SHOT DETECTOR (SSD) *METHOD TO CLASSIFY PAVEMENT DISTRESS ON VIDEO DATA*

ROBERTUS DIAWAN CHRIS
NRP 06111440000066

Supervisor
Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

DEPARTMENT OF MATHEMATICS
Faculty of Science and Data Analytics
Institut Teknologi Sepuluh Nopember
Surabaya 2020

LEMBAR PENGESAHAN

PENERAPAN METODE *SINGLE SHOT DETECTOR* (SSD) UNTUK KLASIFIKASI JENIS KERUSAKAN JALAN PADA DATA VIDEO

IMPLEMENTATION OF SINGLE SHOT DETECTOR (SSD) *METHOD TO CLASSIFY PAVEMENT DISTRESS ON VIDEO DATA*

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Matematika
Pada bidang studi Ilmu Komputer
Program Studi S-1 Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
ROBERTUS DIAWAN CHRIS
NRP. 06111440000066

Menyetujui,

Dosen Pembimbing,

Dr. Dwi Ratna Sulistyaningrum, S.Si, MT
NIP. 19690405 199403 2 003

Mengetahui,
Kepala Departemen Matematika
FSAD ITS

Subchan, Ph.D
NIP. 19710513 199702 1 001
Surabaya, 16 September 2020

PENERAPAN METODE *SINGLE SHOT DETECTOR* (SSD) UNTUK KLASIFIKASI JENIS KERUSAKAN JALAN PADA DATA VIDEO

Nama : **Robertus Diawan Chris**
NRP : **06111440000066**
Departemen : **Matematika**
Dosen Pembimbing : **Dr. Dwi Ratna Sulistyaningrum, S.Si, MT**

ABSTRAK

Pemeriksaan kondisi jalan saat ini masih dilakukan secara konvensional menggunakan tenaga manusia dengan memeriksa langsung kondisi di jalan raya. Berdasarkan Dinas Pekerjaan Umum (DPU), salah satu faktor lamanya proses perbaikan jalan yaitu disebabkan oleh proses pencatatan yang kondisi kerusakan jalan yang masih dilakukan secara manual. Dengan perkembangan teknologi saat ini, telah banyak dilakukan penelitian untuk mendeteksi jenis kerusakan jalan secara otomatis dengan menggunakan pengolahan citra. Pada tugas akhir ini telah dilakukan klasifikasi kerusakan jalan pada data video dengan menggunakan salah satu metode *deep learning* yaitu *single shot detector* (SSD) yang memiliki akurasi lebih tinggi dari *you only look once* dan lebih cepat dari *faster* r-cnn. Tahap dari penelitian ini terdiri dari tiga tahap, yaitu tahap *pre training*, tahap *training*, dan tahap klasifikasi. Penelitian ini menghasilkan akurasi sebesar 75.45% dengan kerusakan jalan berukuran kecil paling sedikit.

***Kata Kunci* :** kerusakan jalan, pengolahan citra, *single shot detector*

“Halaman ini sengaja dikosongkan.”

IMPLEMENTATION OF DEEP LEARNING METHOD TO CLASSIFY PAVEMENT DISTRESS FROM VIDEO DATA

Name of Student : Robertus Diawan Chris
NRP : 06111440000066
Department : Mathematics
Supervisor : Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

ABSTRACT

Examination of road conditions is currently being carried out conventionally using human to manually check the conditions on the road. Based on Public Works Office, one of the factor that make road refinement took longer is because of manual process of recording road conditions. With current technology developments, many studies have been carried out to detect the type of pavement distress automatically using image processing. In this study, the author applied one of deep learning method, that is single shot detector (SSD) *method which is more accurate than you only look once method and faster than faster* r-cnn *method. There is three steps in this study*: *pre training*, *training*, *and classification*. *The result of this study is accuracy of 75.45% with the least small sized pavement distress*.

Keyword : *pavement distress*, *image processing*, *single shot detector*

“Halaman ini sengaja dikosongkan.”

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan ke hadirat Tuhan yang telah memberikan limpahan rahmat-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"Penerapan Metode *Single Shot Detector* (SSD) untuk Klasifikasi Jenis Kerusakan Jalan pada Data Video"** yang merupakan salah satu persyaratan akademis dalam menyelesaikan Program Sarjana Departemen Matematika, Fakultas Sains dan Analitika Data, Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dapat diselesaikan dengan baik berkat kerja sama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal itu, penulis ingin mengucapkan terima kasih dan penghargaan kepada:

1. Subchan, Ph.D selaku Kepala Departemen Matematika ITS yang telah memberikan dukungan dan motivasi selama perkuliahan hingga selesainya Tugas Akhir ini.
2. Dr. Dwi Ratna Sulistyaningrum, S.Si, MT selaku Dosen Pembimbing yang telah memberikan bimbingan, arahan, dan motivasi kepada penulis dalam mengerjakan Tugas Akhir ini sehingga dapat selesai dengan baik.
3. Dr. Budi Setiyono, S.Si., MT, Dr. Dieky Adzkiya, M.Si, dan Moh. Iqbal, S.Si, M.Si selaku dosen penguji yang telah memberikan saran demi perbaikan Tugas Akhir.
4. Drs. Sentot Didik Surjanto, M.Si selaku Dosen Wali yang telah memberikan dukungan dan motivasi selama perkuliahan hingga selesainya Tugas Akhir ini.
5. Seluruh jajaran dosen dan staf jurusan Matematika ITS yang tidak dapat penulis sebutkan satu-persatu.
6. Bapak, Ibu, dan seluruh keluarga penulis yang tidak hentinya memberikan dukungan secara moral dan materiil serta doa untuk kesuksesan penulis
7. Syifa Laili Hapsari Oktavian, Muhammad Muslim, Muhammad Zufar, Muhammad Rifki Meuheimin, dan

Reza Habibi yang telah membantu dengan berdiskusi selama pengerjaan tugas akhir ini.

8. Penghuni lab, Robby Hasanul Arief, Brian Nugraha Dwi Saputra, M. Farras Hanindito R., dan Zicky Lukman yang telah menemani saya selama pengerjaan tugas akhir di lab saat malam hari.
9. Geprek Bentuman dan Mie Astaganaga yang telah membuat makanan yang enak dengan harga terjangkau.
10. Banyak pihak yang tidak dapat ditulis satu persatu oleh penulis yang telah membantu selama penulisan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan saran dan kritik dari pembaca. Akhir kata, semoga Tugas Akhir ini bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, Agustus 2020

Penulis

DAFTAR ISI

“Halaman ini sengaja dikosongkan.”

DAFTAR TABEL

Halaman

“Halaman ini sengaja dikosongkan.”

DAFTAR GAMBAR

Halaman

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan mengenai hal-hal yang melatarbelakangi munculnya permasalahan yang dibahas dalam tugas akhir ini. Kemudian permasalahan tersebut disusun kedalam suatu rumusan masalah. Selanjutnya dijabarkan juga batasan masalah untuk mendapatkan tujuan yang diinginkan serta manfaat yang diperoleh.

1.1 Latar Belakang

Jalan merupakan salah satu prasarana transportasi yang diperlukan untuk menghubungkan daerah satu dengan daerah yang lain. Kondisi jalan dapat mempengaruhi berbagai aktivitas masyarakat. Tingginya aktivitas masyarakat akan berbanding lurus dengan tingginya beban lalu lintas. Beban lalu lintas yang tinggi akan meningkatkan beban jalan sehingga dapat menyebabkan kerusakan perkerasan jalan. Kerusakan jalan raya dapat menyebabkan ketidaknyamanan serta mengancam keselamatan pengguna jalan.

Penyebab munculnya kerusakan jalan dapat bermula dari retakan-retakan kecil yang muncul pada permukaan jalan. Seiring berjalannya waktu retakan-retakan tersebut dapat menjadi lubang-lubang besar yang disebabkan oleh faktor-faktor tertentu, seperti pengaruh cuaca dan banyaknya kendaraan yang memuat beban berat melewati jalan tersebut [1]. Oleh sebab itu, diperlukan pemeriksaan kondisi jalan raya sebagai salah satu cara untuk menjaga keamanan dan kenyamanan pengguna jalan. Pemeriksaan dilakukan untuk merencanakan proses perbaikan dan pemeliharaan jalan yang efektif dan untuk keberlanjutan kondisi jalan yang baik.

Pemeriksaan kondisi jalan saat ini masih dilakukan secara konvensional menggunakan tenaga manusia dengan memeriksa langsung kondisi di jalan raya. Berdasarkan Dinas Pekerjaan Umum (DPU), salah satu faktor lamanya proses

perbaikan jalan yaitu disebabkan oleh proses pencatatan yang kondisi kerusakan jalan yang masih dilakukan secara manual. Proses pendeteksian dan pencatatan secara manual oleh tenaga kerja manusia sepenuhnya bisa memakan waktu dua minggu untuk jalan sepanjang 1 km, belum lagi tingkat keakuratan yang rendah [2].

Dengan perkembangan teknologi saat ini, telah banyak dilakukan penelitian untuk mendeteksi jenis kerusakan jalan secara otomatis dengan menggunakan pengolahan citra. Pada beberapa penelitian sebelumnya telah dilakukan identifikasi dan klasifikasi jenis kerusakan jalan raya dengan beberapa metode dan menggunakan gambar serta video. Tetapi masih banyak penelitian sebelumnya yang menggunakan segmentasi secara manual sehingga memerlukan waktu yang lama untuk melakukan identifikasi dan klasifikasi. Saat ini penelitian dengan metode *deep learning* telah banyak dikembangkan dengan segmentasi secara otomatis sehingga dapat mempersingkat waktu pemrosesan, salah satu metode *deep learning* yang banyak digunakan adalah *Single Shot Detector.* Salah satu penelitian yang pernah dilakukan adalah identifikasi barang berbahaya dalam pemeriksaan keamanan seperti yang digunakan di bandara, stasiun, mall dan tempat lainnya yang banyak dikunjungi orang dengan metode *Single Shot Detector* [3]. Penelitian lain yang telah menerapkan metode *Single Shot Detector* adalah penelitian untuk mendeteksi kapal berdasarkan gambar *Synthetic Aperture Radar* (SAR) untuk memastikan pengawasan dan keamanan transportasi laut [4].

Saat ini penelitian dengan menggunakan metode *deep learning* semakin banyak dikembangkan dengan menggunakan segmentasi secara otomatis sehingga dapat mempersingkat waktu pemrosesan, salah satunya adalah metode *Single Shot Detector* (SSD). Oleh sebab itu, pada Tugas Akhir ini penulis mengusulkan metode *Single Shot*

Detector (SSD) untuk mengklasifikasikan jenis kerusakan jalan pada data video.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, permasalahan yang dibahas dalam Tugas Akhir ini dirumuskan sebagai berikut :

1. Bagaimana mengklasifikasi jenis kerusakan jalan dengan metode *Single Shot Detector* (SSD)?
2. Bagaimana tingkat akurasi hasil klasifikasi jenis kerusakan jalan dengan metode *Single Shot Detector* (SSD)?

1.3 Batasan Masalah

Untuk membatasi ruang lingkup pembahasan permasalahan pada Tugas Akhir ini maka ditentukan beberapa batasan sebagai berikut :

1. Citra digital yang diamati adalah video jalan beraspal
2. Posisi kamera pada saat pengambilan video adalah tegak lurus dengan jalan.
3. Data video yang digunakan merupakan data primer yang diambil secara langsung.
4. Objek yang akan dideteksi dan diklasifikasi yaitu kerusakan jalan jenis retak memanjang, retak melintang, retak aligator, dan lubang.

1.4 Tujuan

Berdasarkan rumusan masalah tersebut, didapat tujuan dari Tugas Akhir ini, yaitu:

1. Melakukan klasifikasi jenis kerusakan jalan dengan metode *Single Shot Detector* (SSD).
2. Mengetahui tingkat akurasi hasil klasifikasi jenis kerusakan jalan dengan metode *Single Shot Detector* (SSD).

1.5 Manfaat

Manfaat yang bisa diperoleh dari Tugas Akhir ini adalah sebagai berikut :

1. Mempermudah pencatatan kondisi kerusakan jalan berbasis video secara otomatis.
2. Meningkatkan akurasi pencatatan kondisi keruskan jalan.
3. Sebagai referensi untuk penelitian selanjutnya dan untuk dinas terkait.

1.6 Sistematika Penulisan Tugas Akhir

Sistimatika penulisan dalam laporan Tugas Akhir ini adalah sebagai berikut :

1. BAB I : PENDAHULUAN
   Bab ini berisi tentang gambaran umum dari penulisan Tugas Akhir yang meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan dan manfaat dari penelitian yang dilakukan, serta sistematika penulisan Tugas Akhir.
2. BAB II : TINJAUAN PUSTAKA
   Bab ini berisi tentang teori dasar yang mendukung dan digunakan penulis dalam Tugas Akhir ini, antara lain kerusakan jalan, citra digital, *deep learning*, operasi konvolusi, gambaran umum arsitektur *neural network*, *single shot detector* (SSD), dan evaluasi kinerja sistem.
3. BAB III : METODOLOGI PENELITIAN
   Bab ini menjelaskan tahap pengerjaan dalam menyelesaikan Tugas Akhir ini sehingga penelitian dapat dirancang secara sistematis dan diatur sebaik-baiknya.
4. BAB IV :PERANCANGAN DAN IMPLEMENTASI SISTEM
   Bab ini menjelaskan tahap persiapan pengolahan data dan penentuan parameter dalam metode yang digunakan sebagai acuan dalam implementasi sistem. Bab ini juga membahas proses untuk implementasi dengan

menggunakan bahasa pemrograman *python* berdasarkan perancangan sistem yang telah dibuat sebelumnya.

5. BAB V : UJI COBA DAN PEMBAHASAN
   Bab ini membahas tentang pengujian sistem yang telah terimplementasi dengan melakukan proses verifikasi dan validasi beserta pengujian kinerja dari sistem yang telah dibuat.
6. BAB VI : PENUTUP
   Bab ini berisi kesimpulan Tugas Akhir yang diperoleh dari bab uji coba dan pembahasan serta saran untuk pengembangan penelitian selanjutnya.

“Halaman ini sengaja dikosongkan.”

BAB II
TINJAUAN PUSTAKA

Pada bab ini diuraikan mengenai tinjauan pustaka yang terdiri dari penelitian terdahulu dan dasar teori yang digunakan dalam penyusunan tugas akhir. Dasar teori yang dijelaskan dibagi menjadi beberapa sub bab yaitu kerusakan jalan, citra digital, *deep learning*, operasi konvolusi, gambaran umum arsitektur *neural network*, *single shot detector* (SSD), dan evaluasi kinerja sistem

2.1 Penelitian Terdahulu

Pada Tugas Akhir ini penulis merujuk pada beberapa penelitian sebelumnya sesuai dengan topik yang diambil. Terdapat beberapa penelitian terdahulu yang terkait kerusakan jalan. Pertama adalah tesis yang ditulis oleh Muhammad Muslim pada tahun 2019 yang berjudul "Identifikasi Lubang pada Jalan Menggunakan Pengolahan Citra Digital". Pada penelitian tersebut dilakukan identifikasi lubang pada citra video. Hasil penelitian tersebut menunjukkan bentuk struktur elemen terbaik adalah *rectangle* dengan ukuran 13 x 13. Namun, penelitian tersebut hanya diterapkan pada kerusakan jalan jenis lubang [5].

Selain penelitian tentang kerusakan jalan, terdapat beberapa penelitian terkait metode *Single Shot Detector* (SSD) yang akan digunakan. Pertama adalah jurnal yang ditulis oleh Xinyang Xu, Qiang Gao, Ye Tian, Weijie Shen, Ruifeng Hong, dan Jun Pan pada tahun 2019 yang berjudul "*Research on Image Detection and Aplication of Security Dangerous Goods Based on SSD*". Pada penelitian tersebut dilakukan deteksi barang-barang berbahaya pada saat pemeriksaan keamanan yang biasanya terdapat di bandara, stasiun, mall dan tempat-tempat yang banyak dikunjungi orang. Hasil dari penelitian tersebut menunjukkan metode SSD secara efektif meningkatkan stabilitas, kecepatan, dan

akurasi dari *image recognition* dan rata-rata akurasi yang dihasilkan lebih dari 90% [3].

Kedua adalah jurnal yang ditulis oleh Yuanyuan Wang, Chao Wang, dan Hong Zhang pada tahun 2017 yang berjudul "*Combining Single Shot Multibox Detector with Transfer Learning for Ship Detection Using Sentinel*-1 *images*". Pada penelitian tersebut dilakukan deteksi kapal berdasarkan gambar *synthetic aperture radar* (SAR) dengan metode SSD. Berdasarkan penelitian tersebut terdapat dua model SSD yang digunakan, yaitu SSD300 dan SSD512 dengan ukuran input masing-masing 300 x 300 dan 512 x 512. Hasil penelitian tersebut menunjukkan model SSD512 mempunyai peluang *false alarm* yang lebih rendah dari model SSD300 [4].

2.2 Kerusakan Jalan

Secara garis besar kerusakan jalan dapat dibedakan menjadi dua bagian, yaitu: kerusakan struktural dan kerusakan fungsional. Kerusakan struktural merupakan kerusakan yang mencakup kegagalan proses pengerasan jalan. Sedangkan kerusakan fungsional merupakan kerusakan yang mengakibatkan terganggunya keamanan dan kenyamanan pengguna jalan [7].

Terdapat berbagai macam kerusakan jalan, berikut merupakan beberapa jenis kerusakan jalan:

a. Retak Memanjang

   Retak memanjang merupakan retak yang sejajar dengan dengan sumbu jalan atau arah penghamparan [8].

b. Retak Melintang

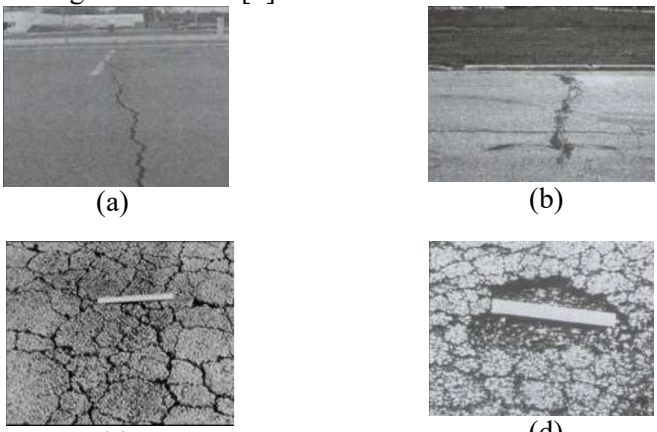
   Retak melintang merupakan retak yang terjadi pada arah lebar perkerasaan dan hampir tegak lurus sumbu jalan atau arah penghamparan [8].

c. Retak Aligator atau Retak Kulit Buaya

Retak aligator merupakan rangkaian retak saling berhubungan pada permukaan lapis beton aspal sebagai akibat keruntuhan telah oleh beban kendaraan yang berulang [8].

d. Lubang

Lubang merupakan cekungan pada permukaan perkerasan yang mempunyai diameter kecil, biasanya kurang dari 750mm [8].



(a) (b) (c) (d)

Gambar 2.1. Beberapa Jenis Kerusakan Jalan [8]: (a) Retak Memanjang, (b) Retak Melintang, (c) Retak aligator (d) Lubang

2.3 Citra Digital

Citra merupakan suatu fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Terdapat dua macam citra, yaitu citra kontinu dan citra diskrit (citra digital). Citra kontinu dihasilkan dari sistem optik yang menerima sinya analog, seperti pada mata manusia dan kamera analog. Sedangkan citra diskrit (citra digital) dihasilkan melalui proses digitalisasi terhadap citra kontinu [9].

Citra juga dapat dinyatakan sebagai suatu fungsi dua dimensi f(x,y), dengan (x,y) merupakan koordinat pada bidang dua dimensi sedangkan f merupakan intensitas cahaya pada titik (x,y). Representasi citra dari fungsi kontinu menjadi nilai-nilai diskrit disebut digitalisasi. Citra yang dihasilkan inilah yang disebut citra digital [9].

Citra digital dapat disajikan dalam bentuk sebuah matriks dengan dimensi M (baris) x N (kolom) yang mana masing-masing elemen matriks mewakili nilai intensitas keabuan dari sebuah citra. Representasi citra dalam bentuk matriks ditunjukkan dengan persamaan (2.1) [9]:

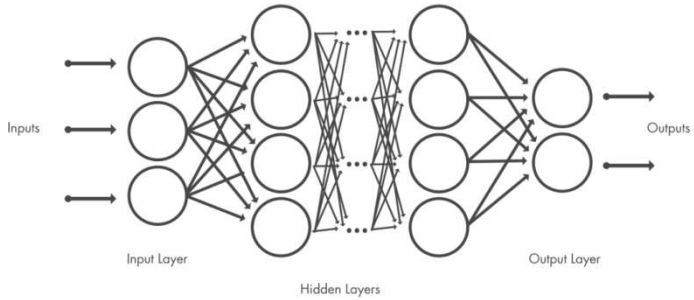
$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \cdots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \quad (2.1)$$

2.4 *Deep Learning*

Deep learning (pembelajaran mendalam) merupakan salah satu bidang *machine learning* yang berkembang pesat saat ini. *Deep learning* mulai dikembangkan sekitar tahun 1940. *Deep learning* terkesan baru karena tidak popular selama beberapa tahun dan juga memiliki nama yang berbeda-beda, nama *deep learning* baru digunakan beberapa tahun terakhir. Sekitar tahun 1940-1960 *deep learning* dikenal dengan nama *cybernetics*, seiring perkembangan teori-teori biologi dan implementasi model pertama, seperti *perceptron* yang memungkinkan pelatihan (*training*) pada

sebuah neuron. Selanjutnya, sekitar tahun 1980-1990 *cybernetics* berubah menjadi *connectionism* atau lebih dikenal sebagai *neural network* yang dikembangkan bersamaan dengan *back-propagation* untuk melatih *neural network* dengan satu atau dua *hidden layers*. Kemudian pada tahun 2006 sampai saat ini *connectionism* berubah menjadi *deep learning* [10].

Deep learning atau dapat juga disebut *deep neural network* menggabungkan beberapa *layers* pemrosesan *nonlinear*, menggunakan elemen sederhana yang dioperasikan secara paralel dan terinspirasi oleh sistem saraf biologi. *Deep learning* terdiri atas *input layers*, beberapa *hidden layers*, dan *output layers*. *Layers* tersebut saling terhubung melalui *nodes* atau *neurons*, dimana masing-masing *hidden layers* menggunakan *output* dari *layer* sebelumnya sebagai *input* [11].



Gambar 2.2. Arsitektur *Deep Learning* [11]

Berikut merupakan perbedaan antara *machine learning* dan *deep learning* [11]:

Tabel 2.1. Perbedaan *Machine Learning* dan *Deep Learning* [11]

Machine Learning	*Deep Learning*
(+) Membutuhkan sedikit *datasets*	(-) Membutuhkan banyak *datasets*
(+) Proses *training* relatif singkat	(-) Proses *training* relatif lama
(-) Ekstraksi fitur secara manual	(+) Ekstraksi fitur secara otomatis
(-) Akurasi menurun dari waktu ke waktu	(+) Akurasi tidak menurun dari waktu ke waktu

2.5 Operasi Konvolusi

Operasi konvolusi merupakan operasi mendasar dalam pengolahan citra. Operasi konvolusi merupakan operasi dari dua fungsi f dan g yang menghasilkan sebuah fungsi h [9]. Operasi konvolusi terdiri dari dua jenis, yaitu operasi konvolusi pada fungsi kontinu dan operasi konvolusi pada fungsi diskrit. Operasi konvolusi pada fungsi kontinu dapat didefinisikan sebagai berikut:

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da \tag{2.2}$$

untuk fungsi diskrit dapat didefinisikan sebagai berikut:

$$h(x) = f(x) * g(x) = \sum_{a=-\infty}^{\infty} f(a)g(x-a) \tag{2.3}$$

Pengolahan citra digital berlaku pada domain spasial sehingga konvolusi yang digunakan adalah konvolusi diskrit [12]. Citra digital merupakan fungsi dua dimensi sehingga

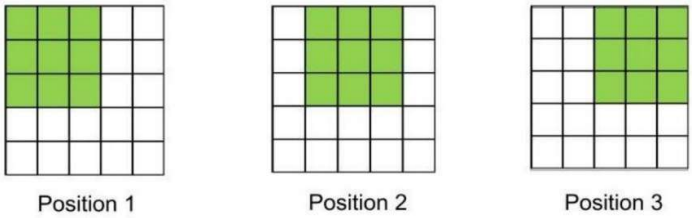
operasi konvolusi diskrit dapat didefinisikan sebagai berikut [9]:

$$h(x,y) = f(x,y) * g(x,y) \\ = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f(a,b) g(x-a, x-b) \quad (2.4)$$

dengan:

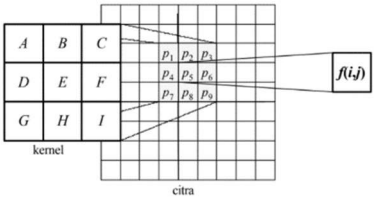
tanda * : menyatakan operator konvolusi
a, b : peubah bantu (*dummy variable*)
f : fungsi masukan (*input*)
g : kernel konvolusi (*filter*)
h : hasil operasi konvolusi

Operasi konvolusi dilakukan dengan menggeser kernel konvolusi *pixel* per *pixel* dan hasilnya disimpan dalam matriks yang baru. Contoh pergeseran pada operasi konvolusi citra ditunjukkan pada gambar 2.3



Gambar 2.3. Contoh Pergeseran Kernel Pada Citra [24]

Berikut merupakan ilustrasi operasi konvolusi:



Gambar 2.4. Ilustrasi Operasi Konvolusi [9]

Operasi konvolusi dapat dipandang kombinasi linier dari vektor *pixel* (citra) dengan vektor kernel dan diperoleh persamaan sebagai berikut [9]:

$$f(i,j) = (A \times p_1) + (B \times p_2) + (C \times p_3) + (D \times p_4) + (E \times p_5) + (F \times p_6) + (G \times p_7) + (H \times p_8) + (I \times p_9) \quad (2.9)$$

untuk mendapatkan hasil konvolusi citra, dilakukan perhitungan menggunakan persamaan di atas dari sudut kiri atas hingga kernel berada pada sudut kanan bawah.

Jika hasil konvolusi menghasilkan nilai *pixel* negatif, maka nilai tersebut dijadikan 0. Sedangkan jika hasil konvolusi menghasilkan nilai pixel lebih besar dari nilai keabuan maksimum, maka nilai tersebut dijadikan ke nilai keabuan maksimum [9].

2.6 Gambaran Umum Arsitektur *Neural Network*

Pada umumnya, arsitektur *neural network* terdiri dari beberapa *layers* yaitu: *input layers*, *hidden layers*, dan *output layers*. *Neural network* memiliki kemampuan dan kerumitan yang berbeda-beda, mulai dari yang paling sederhana hanya terdiri dari satu neuron (*single neuron*) hingga yang rumit seperti beberapa neuron di beberapa *layers* (*multiple neuron in multiple layers*) [13]. Berikut merupakan beberapa contoh arsitektur *neural network* berdasarkan jumlah *layers* [14]:

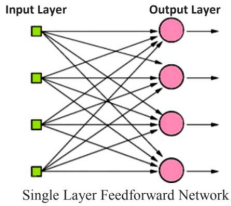
a. *Single Layer Neural Network*

Single layer neural network hanya memiliki satu neuron yang terhubung secara penuh (*fully-connected*) pada layer terakhir tanpa adanya pembagian koneksi antar neuron (*no hidden layer*). *Layer* terakhir pada keseluruhan jaringan bersifat terhubung secara penuh dan merupakan representasi dari nilai kelas klasifikasi.

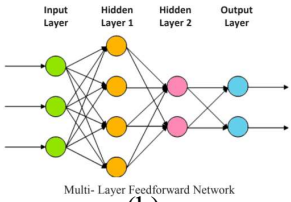
b. *Multilayers Neural Network*

Multilayers neural network memiliki tiga jenis *layers*, yaitu *input layers*, *hidden layers*, dan *output layers*. Setiap *hidden layers* tersusun dari kumpulan

neuron, dimana setiap neuron terhubung secara penuh (*fully-connected*) ke semua neuron pada *layer* sebelumnya. *Layer* terakhir pada keseluruhan jaringan bersifat terhubung secara penuh dan merupakan representasi dari nilai kelas klasifikasi.



(a)



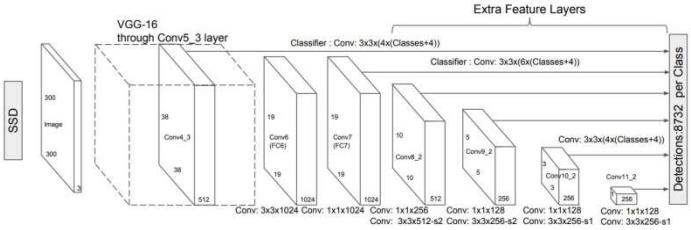
(b)

Gambar 2.5. Beberapa Contoh Arsitektur *Neural Network* [14]: (a) *Single Layer Neural Network*, (b) *Multilayers Neural Network*.

Apabila *neural network* diberikan input berupa gambar dengan dimensi $M \times N \times C$ (M = lebar, N = tinggi, C = *channel*) dihubungkan secara penuh (*fully-connected*) maka jumlah parameter bobot sama dengan $M \times N \times C$ bobot akan menjadi sangat besar ketika ukuran gambar sangat besar. *Fully-connected* kurang efektif ketika dihubungkan dengan banyak parameter yang nantinya dapat menyebabkan *overfitting*. Pada umumnya, *overfitting* merupakan kondisi ketika model memiliki performa yang bagus pada data *training* tetapi tidak mengeneralisasi secara baik pada data *testing* [25].

2.7 *Single Shot Detector* (SSD)

Single shot detector (SSD) merupakan salah satu metode *deep learning* yang diusulkan oleh Wei Liu, Dragomir Anguelov, Dumitru Erhan, Chirstian Szegedy, Scott Reed, Cheng-Yang Fu, dan Alexander C. Berg dalam jurnalnya yang berjudul SSD: *Single Shot Multibox Detector*. SSD dibuat berdasarkan *feed-forward convolutional neural network* yang menghasilkan koleksi *bounding boxes* dan *scores* untuk menunjukkan prediksi dari objek dalam *bounding boxes*. *Layers* pertama, biasanya disebut *base network*, menggunakan arsitektur *image classification* tanpa *classification layers*. *Base network* yang digunakan pada penelitian tersebut adalah VGG (*Visual Geometry Group*) [15]. Lima *convolutional layers* pada VGG digunakan sebagai *base network* [3] dan kemudian ditambahkan beberapa *feature layers* pada bagian akhir *base network* yang memprediksi dan memberikan *bounding box* pada objek dengan skala dan aspek rasio yang berbeda serta *confidence* bahwa objek tersebut sesuai dengan *class* yang telah ditetapkan [15].

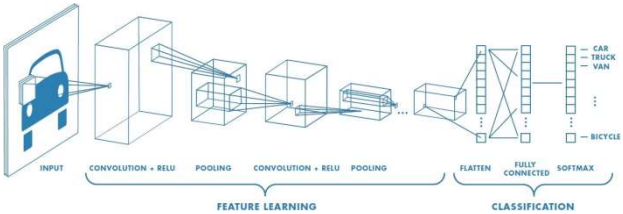


Gambar 2.6. Arsitektur *Single Shot Detector* [15].

2.7.1 Gambaran Umum Arsitektur *Convolusional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan *layer* yang memiliki susunan neuron tiga dimensi berukuran $W \times H \times D$, yang mana W (*width*)

dan *H* (*height*) merupakan ukuran *layer*, sedangkan *D* (*depth*) merupakan kedalaman jumlah *layer* [13]. Secara umum, arsitektur CNN ditunjukkan pada gambar 2.6.



Gambar 2.7. Arsitektur CNN [16].

Secara umum, *layers* pada CNN terdiri dari dua jenis, yaitu [13]:

a. *Feature Extraction Layers* (*Layers* Ekstraksi Fitur)

Feature extraction layers tersusun dari beberapa *layer*, yaitu *convolutional layer* dan *pooling layer*. Pada tahap ekstraksi fitur dilakukan konvolusi pada *convolutional layer* dan kemudian diberlakukan fungsi aktivasi. Selanjutnya, dilakukan *pooling* hingga diperoleh hasil berupa vektor (*feature maps*) untuk diolah pada *layer* selanjutnya.

b. *Classification Layers* (*Layers* Klasifikasi)

Classification layers tersusun dari beberapa *layer*, dimana tiap *layer* tersusun atas neuron yang terkoneksi secara penuh (*fully-connected*) dengan *layer* lain. *Layer* tersebut menerima input berupa *feature maps* dan kemudian ditransformasikan seperti *multi neural networks* dengan beberapa *hidden layers* hingga diperoleh

hasil berupa skor kelas (*class score*) klasifikasi [13].

2.7.2 Gambaran Umum *Visual Geometry Group* (VGG)

Visual Geometry Group (VGG) merupakan salah satu model *transfer learning*. *Transfer learning* merupakan suatu metode untuk mentransfer pengetahuan yang telah dipelajari pada tugas (*task*) sebelumnya dan menggunakannya untuk meningkatkan pembelajaran (*learning*) pada tugas yang baru [17]. *Transfer learning* bermanfaat untuk tugas yang memiliki sedikit data *training* dan tidak cukup untuk melatih model yang baik, sedangkan tugas yang mirip pada domain yang berbeda memiliki banyak data [4]. Contoh dari tugas yang dimaksud adalah klasifikasi gambar (*image classification*) dan domain yang dimaksud adalah objek yang diklasifikasikan pada gambar tersebut.

VGG diusulkan oleh K. Simonyan dan A. Zisserman dari *university of oxford* dalam jurnalnya yang berjudul *Very Deep Convolutional Network for Large-Scale Image Recognition*. VGG telah dilatih pada ImageNet *Large-Scale Visual Recognition Challenge* (ILSVRC) 2014 dataset yang mencakup 1000 kategori dengan 1.2 juta gambar dan berhasil menempati urutan pertama pada pelacakan posisi objek dan urutan kedua pada klasifikasi objek [19]. Oleh sebab itu, VGG sering digunakan sebagai *base network* pada SSD sehingga dapat menghasilkan performa deteksi objek yang baik dengan jumlah data *training* yang tidak banyak.

2.8 Evaluasi Kinerja Sistem

Evaluasi kinerja sistem merupakan evaluasi yang dilakukan untuk mengetahui kemampuan sistem mengklasifikasi jenis kerusakan jalan. Evaluasi kinerja sistem dilakukan dengan menghitung presisi, *recall*, dan akurasi.

Presisi merupakan tingkat ketepatan antara informasi yang dibutuhkan oleh pengguna dan jawaban yang diberikan oleh sistem. *Recall* merupakan tingkat keberhasilan sistem untuk menemukan kembali sebuah informasi. Akurasi merupakan tingkat kedekatan antara nilai prediksi dan nilai sebenarnya.

Berikut merupakan rumus presisi, *recall*, dan akurasi pada sistem klasifikasi [13]:

$$presisi = \frac{TP}{FP+TP} \times 100\% \tag{2.10}$$

$$recall = \frac{TP}{FN+T} \times 100\% \tag{2.11}$$

$$akurasi = \frac{TP+TN}{TP+TN+FP+F} \times 100\% \tag{2.12}$$

dengan:

TP (*True Positive*) : Jumlah data positif yang terklasifikasi benar oleh sistem

TN (*True Negative*) : Jumlah data negatif yang terklasifikasi benar oleh sistem

FP (*False Positive*) : Jumlah data positif yang terklasifikasi salah oleh sistem

FN (*False Negative*): Jumlah data negatif yang terklasifikasi salah oleh sistem

“Halaman ini sengaja dikosongkan.”

BAB III
METODOLOGI PENELITIAN

Pada bab ini diuraikan mengenai langkah-langkah yang digunakan, prosedur dan proses setiap langkah yang akan dilakukan dalam penyelesaian tugas akhir ini.

3.1 Tahap Penelitian

Tahap-tahap yang dilakukan pada tugas akhir ini sebagai berikut:

Gambar 3.1. Diagram Tahap Penelitian

a. Studi Literatur

Pada tahap ini dilakukan pengamatan dan mempelajari jurnal-jurnal nasioal dan internasional, *text book* terkait kerusakan jalan, pengolahan citra digital, dan klasifikasi dengan metode *single shot detector* (SSD), serta mempelajari penelitian-penelitian terdahulu untuk mengetahui pengembangan yang diperlukan pada penelitian tersebut. Tujuan studi literatur ini yaitu untuk memperoleh dan memahami teori-teori yang diperlukan serta mendukung proses penelitian.

b. Pengumpulan Data

Pada tahap ini dilakukan pengumpulan data video kerusakan jalan. Pengumpulan data video dilakukan dengan pengambilan data video secara langsung menggunakan mobil dengan posisi kamera tegak lurus dengan jalan.

c. Perancangan Program

Pada tahap ini dilakukan perancangan program yang dapat melakukan klasifikasi kerusakan jalan menggunakan metode *single shot detector* (SSD). Dalam perancangan program ini terdapat dua tahap yaitu *pre-processing* data dan perancangan sistem klasifikasi dengan *training* metode SSD.

d. Implementasi Program

Pada tahap ini dilakukan implementasi dengan membuat program untuk mengklasifikasi kerusakan jalan beraspal dengan metode SSD. Input pada program ini berupa video jalan beraspal dengan kerusakan jalan. Output pada program ini berupa video dengan *bounding box* dan label jenis kerusakan jalan pada kerusakan jalan yang diklasifikasikan.

e. Uji Coba dan Evaluasi

Pada tahap ini dilakukan uji coba dengan menggunakan beberapa data video kerusakan jalan yang berbeda dari yang digunakan pada tahap *training*. Hal tersebut bertujuan untuk menguji kemampuan program dalam mengklasifikasikan jenis kerusakan jalan sehingga dapat dilakukan evaluasi pada program yang telah dibuat. Evaluasi yang dilakukan dibagi menjadi evaluasi tahap klasifikasi dan evaluasi pada program antarmuka (*interface*) untuk mengetahui kualitas program yang telah dibuat.

f. Penarikan Kesimpulan

Pada tahap ini dilakukan penarikan kesimpulan yang merupakan ringkasan akhir dari hasil implementasi, pengujian, dan evaluasi yang telah dilakukan untuk menjawab tujuan penelitian terhadap program yang telah dibuat serta memberikan saran terkait kekurangan yang terdapat pada program yang telah dibuat agar menjadi pertimbangan untuk pengembangan penelitian selanjutnya.

g. Penyusunan Laporan Tugas Akhir

Pada tahap ini dilakukan penulisan laporan tugas akhir berdasarkan seluruh proses yang telah dilakukan pada penelitian ini dan hasil penelitian yang diperoleh.

3.2 Tahap Perancangan

Proses klasifikasi jenis kerusakan jalan pada tugas akhir ini dibagi menjadi beberapa tahap sebagai berikut:

a. Pengumpulan Data

Pada tahap ini dilakukan pengambilan data video kerusakan jalan secara langsung dengan posisi kamera tegak lurus dengan jalan. Pengambilan data video dilakukan di jalan kertajaya indah timur ix, jalan kertajaya indah timur x, dan jalan kertajaya inda timur xi.

b. *Pre Training*

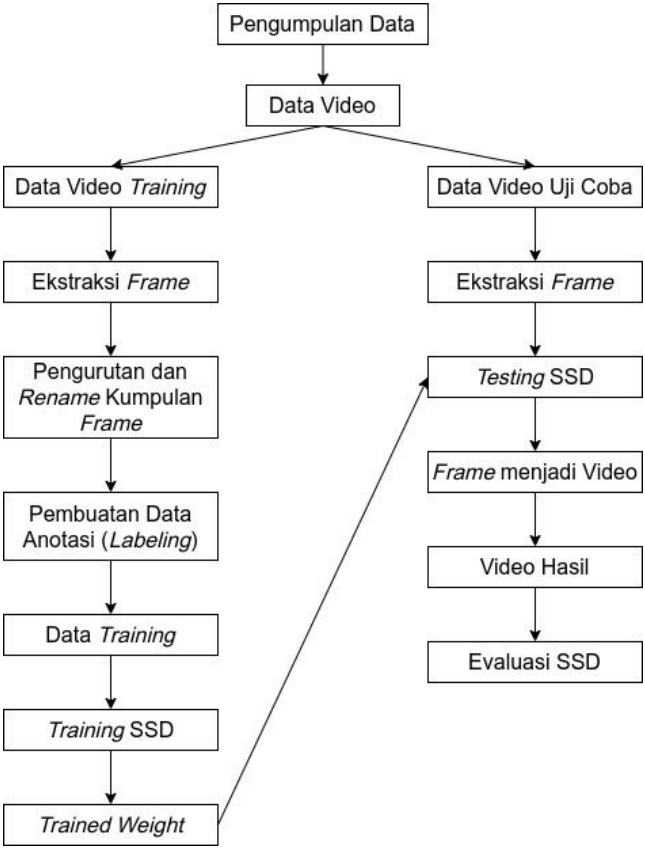
Tahap ini terdiri dari tiga proses yaitu ekstraksi *frame* yang memiliki kerusakan jalan dari data video *training* yang telah diambil, mengurutkan dan *rename* kumpulan *frame*, dan pembuatan data anotasi (*labeling*). Pengurutan dilakukan agar kumpulan *frame* dari beberapa data video dapat terurut pada satu folder dengan nama file yang berurutan. Pembuatan data anotasi (*labeling*) bertujuan untuk membuat *ground truth* sebagai acuan saat melakukan tahap *training*.

c. *Training* Metode *Single Shot Detector* (SSD)

Pada tahap ini dilakukan *training* terhadap metode *single shot detector* (SSD) sehingga dapat mengenali jenis kerusakan jalan secara baik. Proses *training* ini bertujuan untuk menentukan bobot terbaik. Input dari tahap ini berupa kumpulan *frame* yang telah diurutkan beserta anotasinya, sedangkan output dari tahap ini berupa bobot terlatih atau disebut *trained weight*.

d. Klasifikasi Jenis Kerusakan Jalan

Pada tahap ini dilakukan klasifikasi jenis kerusakan jalan pada data video uji coba dengan menggunakan *trained weight* yang diperoleh pada tahap *training*. Output pada tahap ini berupa video dengan *bounding box* dan label jenis kerusakan jalan pada kerusakan jalan yang diklasifikasikan.



Gambar 3.2. Diagram Tahap Perancangan

“Halaman ini sengaja dikosongkan.”

BAB IV
PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bab ini dijelaskan mengenai perancangan dan implementasi sistem yang berisi tentang penjelasan perancangan data dan perancangan proses pada sistem.

4.1 Pengambilan Data dan Perancangan Data

Pada sub bab ini dijelaskan mengenai perancangan data yang dibutuhkan untuk implementasi metode *single shot detector* (SSD) untuk klasifikasi jenis kerusakan jalan. Berikut merupakan uraian pengambilan data dan perancangan data yang dilakukan:

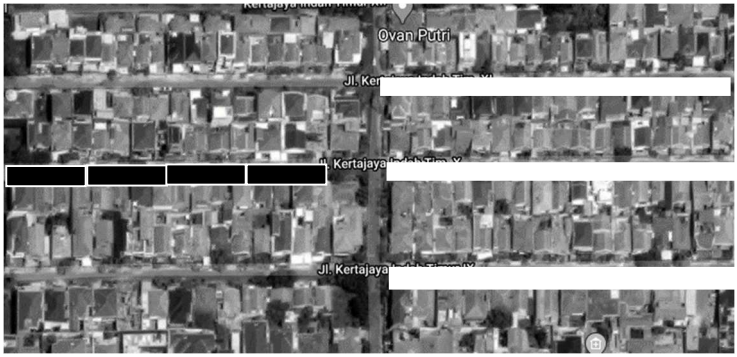
4.1.1 Pengambilan Data

Data yang digunakan merupakan data primet yang terdiri dari beberapa jenis kerusakan jalan, yaitu: retak aligator, retak memanjang, retak melintang, dan lubang. Lokasi pengambilan data berada di Surabaya, di jalan kerjaya indah timur IX, jalan kertajaya indah timur X, dan jalan kertajaya indah timur XI. Data diambil dengan melakukan perekaman disepanjang jalan yang dilalui dengan menggunakan kamera yang diletakkan dibagian belakang mobil dengan ketinggian dari aspal ke kamera adalah 200 cm. Data yang berhasil didapatkan sebanyak 7 video, yang terdiri dari 3 video *training* dan 4 video uji coba dengan format *.mp4. Posisi kamera pada mobil ditunjukkan pada gambar 4.1.

Yang membedakan video *training* dan *testing* adalah lokasi pengambilan video dan waktu pengambilan video. Pengambilan video *training* dilakukan pada pagi hari sekitar jam 06:00 dalam kondisi cerah, sedangkan pengambilan video uji coba dilakukan pada siang hari sekitar jam 14:00 dalam kondisi berawan (setelah hujan). Pembagian tempat untuk

pengambilan video *training* dan video uji coba ditunjukkan pada gambar 4.2.

Gambar 4.1. Posisi Kamera pada Mobil



Gambar 4.2. Pembagian Tempat Pengambilan Video *Training* (bagian berwarna putih) dan Video Uji Coba (bagian berwarna hitam)

4.1.2 Persiapan Data *Training*

Data video *training* diolah pada tahap *pre training* yang terdiri dari ekstrasi *frame* yang terdapat kerusakan jalan, pengurutan dan *rename* kumpulan *frame*, dan terakhir adalah pembuatan data anotasi (*labeling*) terhadap kerusakan jalan yang terdapat pada *frame* tersebut. Dalam tugas akhir ini didapat 652 citra jalan untuk tahap *training* dengan 367 retak aligator, 951 retak memanjang, 243 retak melintang, dan 161 lubang.

Data citra tersebut dibagi menjadi dua dataset terpisah untuk digunakan sebagai data *training* dan data validasi. Data *training* digunakan untuk melakukan *training* model, sedangkan data validasi digunakan untuk mengevaluasi *trained weight*. Data *training* yang digunakan sebanyak 467 citra yang didapatkan dari dua video *training* dan data validasi *trained weight* yang digunakan sebanyak 185 yang didapatkan dari satu video *training*.

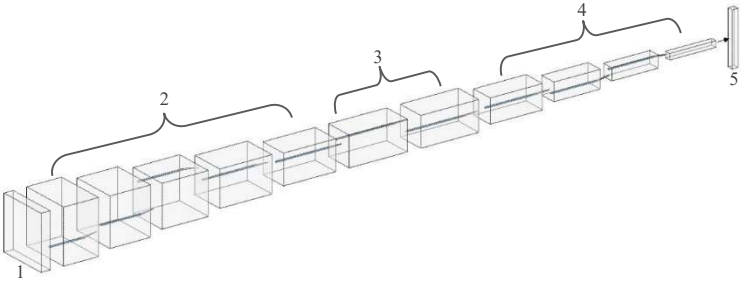
Diperlukannya data validasi untuk melihat apakah model dapat mengenali objek dengan benar pada data yang belum pernah diberikan. Jika model mampu mengenali sebagian objek maka model tersebut dapat dikatakan baik

4.2 Perancangan Sistem

Pada sub bab ini dijelaskan mengenai perancangan sistem dalam implementasi metode *single shot detector* (SSD) untuk klasifikasi jenis kerusakan jalan. Perancangan sistem terdiri dari desain arsitektur *Single Shot Detector* (SSD), tahap *pre training*, tahap *training*, dan tahap klasifikasi.

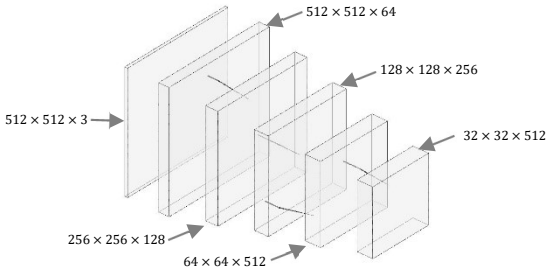
4.2.1 Desain Arsitektur *Single Shot Detector* (SSD)

Desain arsitektur *single shot detector* (SSD) yang digunakan pada tugas akhir ini ditunjukkan pada gambar 4.1.



Gambar 4.3. Arsitektur *Single Shot Detector* (SSD): (1) *Input Image*, (2) *Base Network*, (3) *Layer* 6 *and Layer* 7, (4) *Extra Feature Layers*, (5) *Collections of Boxes and Scores*

Berdasarkan gambar 4.3 diketahui bahwa tahapan utama pada arsitektur SSD dibagi menjadi tiga, yaitu: tahap pada *base network*, tahap merubah *fully-connected layer* menjadi *convolutional layer*, dan tahap *extra feature layers*. Rincian dari tahap pada *base network* ditunjukkan pada gambar 4.4.



Gambar 4.4. Rincian *Layer* Pada *Base Network*

Base network yang digunakan pada tugas akhir ini adalah *visual geometry group* (VGG), sehingga arsitektur pada tahap *base network* menggunakan arsitektur *visual geometry group* (VGG). Input dari VGG berupa citra RGB yang dalam penelitian ini berukuran 512×512 sesuai dengan input arsitektur SSD. Pada *base network* tidak menggunakan seluruh bagian dari arsitektur VGG, yang digunakan hanyalah *layer* pertama hingga *layer* kelima. Pada *fully-coneccted layer* keenam (fc6) dan ketujuh (fc7) *base network* diubah menjadi *convolutional layer*, perubahan tersebut dilakukan agar tidak terjadi *computational bottleneck* (penurunan performa komputasi). Sedangkan untuk *fully-connected* kedelapan (fc8) dan *softmax* pada *base network* tidak digunakan karena pada fc8 dan *softmax* merupakan konfigurasi khusus untuk klasifikasi 1000 *class* pada *ImageNet Large Scale Visual Recognition Competition* (ILSVRC). Selanjutnya pada bagian akhir *base network* (fc7) ditambahkan *extra feature extraction layer*.

VGG digunakan sebagai *base network* karena VGG memiliki keungulan yaitu dapat mentransfer kemampuan pembelajarannya (*transfer learning*) yang didapatkan dari proses *training* pada dataset yang lain (*pre-trained*). Menggunakan VGG sebagai *base network* memiliki beberapa keuntungan, yaitu: proses *training* menjadi lebih cepat dikarenakan proses pembelajaran tidak mulai dari awal (model telah mengatahui *basic* dari *object detection*) dan data yang dibutuh pada proses *training* lebih sedikit daripada *network* yang lain.

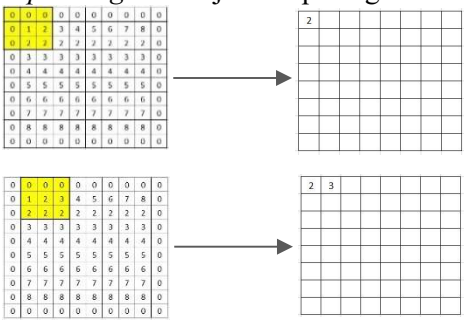
Pada *layer* pertama dari *base network* dilakukan operasi konvolusi dengan kernel filter berukuran 3×3 dan *stride* (jumlah pergeseran filter) sebanyak 1 *pixel*. Proses tersebut dilakukan sebanyak dua kali dan selanjutnya dilakukan *max pooling* dengan filter 2×2 dan *stride* 2 sehingga diperoleh hasil berukuran $256 \times 256 \times 128$.

Pada *layer* kedua dari *base network* dilakukan operasi konvolusi dengan kernel filter berukuran 3×3 dan *stride* sebanyak 1 *pixel*. Proses tersebut dilakukan sebanyak dua kali dan selajutnya dilakukan *max pooling* dengan filter 2×2 dan *stride* 2 sehingga diperoleh hasil berukuran $128 \times 128 \times 256$.

Pada *layer* ketiga dari *base network* dilakukan operasi konvolusi dengan kernel filter berukuran 3×3 dan *stride* sebanyak 1 *pixel*. Proses tersebut dilakukan sebanyak tiga kali dan selanjutnya dilakukan *max pooling* dengan filter 2×2 dan *stride* 2 sehingga diperoleh hasil berukuran $64 \times 64 \times 512$.

Pada *layer* keempat dari *base network* dilakukan operasi konvolusi dengan kernel filter berukuran 3×3 dan *stride* sebanyak 1 *pixel*. Proses tersebut dilakukan sebanyak tiga kali dan selanjutnya dilakukan *max pooling* dengan filter 2×2 dan *stride* 2 sehingga diperoleh hasil berukuran $32 \times 32 \times 512$.

Pada *layer* kelima dari *base network* dilakukan operasi konvolusi dengan kernel filter berukuran 3×3 dan *stride* sebanyak 1 *pixel*. Proses tersebut dilakukan sebanyak tiga kali. Pada *layer* ini *max pooling* dilakukan dengan kernel filter 3×3 dan *stride* 1 dengan *padding* (penambahan nilai intensitas nol di daerah sekitar input gambar) sebanyak 1 sehingga diperoleh hasil berukuran $32 \times 32 \times 512$. Ilustrasi *max pooling* berukuran 8×8 dengan kernel filter 3×3 dan *stride* 1 dengan *padding* 1 ditunjukkan pada gambar 4.5.



Gambar 4.5. Ilustrasi *Max Pooling*

Berikut merupakan persamaan untuk mengetahui tinggi dan lebar setelah dilakukan proses *max pooling* [22]:

$$H_{out} = \left(\left(\frac{H_{in} + (2 \times padding[0]) - (dilation[0] \times (kernel\_size[0] - 1))}{stride[0]} \right) + 1 \right) \quad (4.1)$$

$$W_{out} = \left(\left(\frac{W_{in} + (2 \times padding[1]) - (dilation[1] \times (kernel\_size[1] - 1))}{stride[1]} \right) + 1 \right) \quad (4.2)$$

Keterangan:

- $kernel\_size$, $stride$, $padding$, dan *dilation* dapat berupa satu integer (dalam hal ini nilai yang digunakan sama untuk tinggi dan lebar) atau dua integer (dalam hal ini integer pertama digunakan sebagai tinggi dan integer kedua digunakan sebagai lebar).
- H_{out}: Tinggi yang dihasilkan
- H_{in}: Tinggi input
- W_{ou} : Lebar yang dihasilkan
- W_{in}: Tinggi input
- $padding$: Banyaknya penambahan intesitas nol disekitar input gambar
- $dilation$: jarak antara nilai di kernel (*dilation* pada operasi konvolusi pada umumnya adalah 1)
- $kernel\_size$: ukuran kernel filter
- $stride$: jumlah pergeseran filter

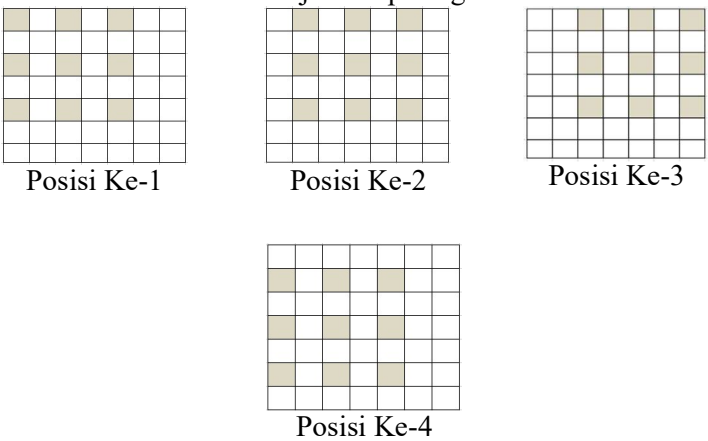
Berikut merupakan ilustrasi perhitungan *max pooling* dengan input 32×32 dengan kernel filter 3×3, *stride* 1, *padding* 1, dan *dilation* 1:

$$H_{out} = \left(\left(\frac{32 + (2 \times 1) - (1 \times (3 - 1))}{1} \right) + 1 \right) = \left(\left(\frac{32 + 0}{1} \right) + 1 \right) = 32$$

$$W_{out} = \left(\left(\frac{32 + (2 \times 1) - (1 \times (3 - 1))}{1} \right) + 1 \right) = \left(\left(\frac{32 + 0}{1} \right) + 1 \right) = 32$$

sehingga didapatkan hasil berukuran 32×32.

Setelah dilakukan ekstraksi fitur pada *base network*, langkah selanjutnya adalah mengubah *fully-connected layer* pada *base network* menjadi *convolutional layer* dengan cara mengambil *subsample parameters* dari fc6 dan fc7 (salah satu contoh *parameter* yang diambil adalah ukuran yang dihasilkan pada *layer* kelima sebelum memasuki *fully-connected layer*). Operasi konvolusi yang dilakukan pada *layer* 6 dan 7 adalah *atrous convolutions* yang ilustrasinya pergeseran operasi *atrous convolutions* ditunjukkan pada gambar 4.6.

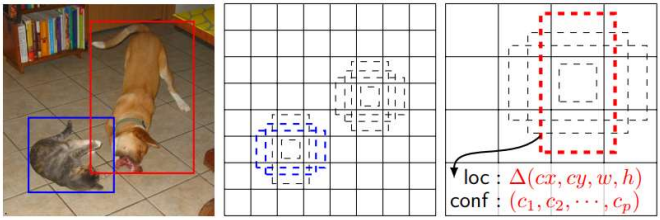


Gambar 4.6. Pergeseran Operasi *Atrous Convolutions* dengan *dilation* 2, kernel filter 3×3, dan *stride* 1

Dengan menggunakan *atrous convolutions* dapat meningkatkan area yang diamati untuk ekstraksi fitur sambil mempertahankan jumlah *parameters* yang relatif lebih sedikit operasi konvolusi pada umumnya.

Selanjutnya pada *extra feature layers* merupakan *layer* yang digunakan untuk melakukan prediksi. Untuk setiap *feature layer* berukuran $m \times n$ dengan p jumlah *layer*, elemen basis untuk *parameter* prediksi dari peluang objek yang dideteksi adalah $3 \times 3 \times p$ kernel yang menghasilkan antara *score* untuk kategori atau bentuk yang relatif terhadap *default*

box. Untuk setiap $m \times n$ lokasi yang diberi kernel menghasilkan nilai output. Nilai b*ounding box* output diukur relatif terhadap posisi *default box* untuk setiap lokasi *feature maps*. *Default box* merupakan kotak yang memiliki berbagai aspek rasio yang berada pada tiap lokasi dari beberapa *feature maps* dengan ukuran yang berbeda. Berikut merupakan ilustrasi *default box* pada setiap lokasi *feature maps*:



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Gambar 4.7. *Default Box* pada *Feature Maps* 8×8 dan 4×4 [15]

Pada *layer* yang terakhir merupakan kumpulan dari *default box* yang mendekati dengan *ground truth box* beserta *confidence score* dari *default box* tersebut dan didapatkan hasil klasifikasi dari objek yang terdeteksi.

4.2.2 Perancangan Tahap *Pre Training*

Tahap ini merupakan tahap yang dilakukan agar data dapat digunakan secra optimal pada tahap selanjutnya. Tahap *pre training* terdiri dari melakukan ekstraksi *frame* dari tiga video *training*, pengurutan dan *rename* kumpulan *frame* dan pembuatan data anotasi (*labeling*) pada file *screenwhot* yang telah diurutkan. Berikut merupakan penjelasan mengenai masing-masing proses yang akan dilakukan pada tahap *pre training*:

a. Ekstraksi *Frame* dari Video *Training*

Pengambilan data untuk proses *training* dilakukan dengan ekstraksi *frame* yang terdapat pada tiga video

training dan menyimpannya dalam folder sesuai dengan nama video tersebut. Penyimpanan hasil ekstraksi *frame* pada folder yang berbeda dilakukan agar tidak terjadi tumpang tindih antara *frame* yang sama (contoh: *frame* ke-10 pada video pertama dan *frame* ke-10 pada video kedua).

b. Pengurutan dan *Rename* Kumpulan *Frame*

Data yang diperoleh dari proses ekstraksi *frame* dari video *training* dilakukan pengurutan dan *rename* file, hal ini dilakukan untuk mempermudah proses pembuatan data anotasi (*labeling*). Setelah dilakukan pengurutan, data yang diperoleh dari video *training* pertama tidak digabung dengan data dari kedua video *training* dan dilakukan proses pembuatan data anotasi (*labeling*) terpisah.

c. Pembuatan Data Anotasi (*Labeling*)

Pembuatan anotasi dilakukan dengan memberikan label jenis kerusakan jalan gambar. Data anotasi digunakan sebagai *ground truth box* pada tahap *training*.

4.2.3 Perancangan Tahap *Training*

Tahap ini merupakan tahap yang dilakukan agar sistem dapat mengenali jenis kerusakan jalan secara tepat. Tahap *training* terdiri dari beberapa proses.

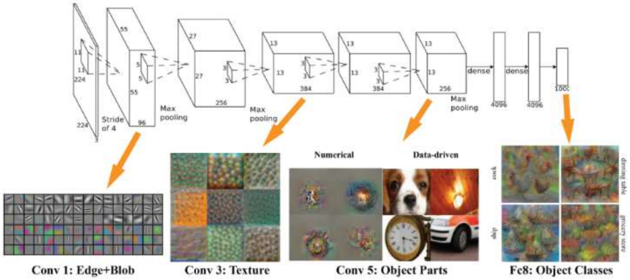
Berikut merupakan penjelasan mengenai masing-masing proses yang akan dilakukan pada tahap *training*:

a. Input File *Training*

Pada proses inisialisasi tahap *training* diberikan input file *training* yang telah diproses pada tahap *pre training*.

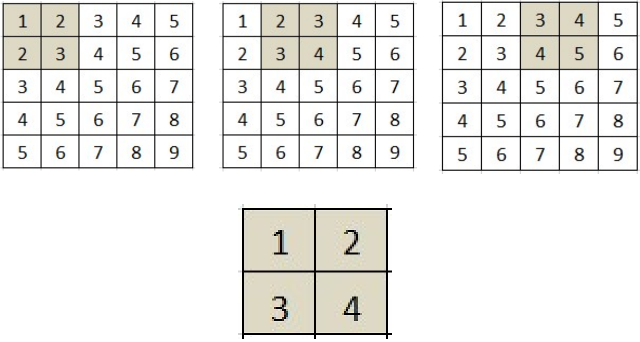
b. Proses Pada *Base Network*

Pada tahap *base network* dilakukan operasi konvolusi untuk mengubah input menjadi *feature maps*. *Feature maps* merepresentasikan dominan fitur pada suatu gambar pada skala yang berbeda.



Gambar 4.8. Visualisasi *Feature Maps* [20]

Ilustrasi dari operasi konvolusi ditunjukkan pada gambar 4.9.



1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8
5	6	7	8	9

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8
5	6	7	8	9

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8
5	6	7	8	9

1	2
3	4

Gambar 4.9. Ilustrasi Operasi Konvolusi

Berdasarkan gambar 4.9 diketahui bahwa citra $f(x, y)$ berukuran 5×5 dengan kernel $g(x, y)$ berukuran 2×2 dan *stride* 1. Perhitungan berdasarkan ilustrasi tersebut sebagai berikut:

1. $h(1,1)$
   $= (1 \times 1) + (2 \times 2) + (2 \times 3) + (3 \times 4)$
   $= 23$
2. $h(1,2)$
   $= (2 \times 1) + (3 \times 2) + (3 \times 3) + (4 \times 4)$
   $= 33$
3. $h(1,3)$
   $= (3 \times 1) + (4 \times 2) + (4 \times 3) + (5 \times 4)$
   $= 43$
4. $h(1,4)$
   $= (4 \times 1) + (5 \times 2) + (5 \times 3) + (6 \times 4)$
   $= 53$
5. $h(2,1)$
   $= (2 \times 1) + (3 \times 2) + (3 \times 3) + (4 \times 4)$
   $= 33$
6. $h(2,2)$
   $= (3 \times 1) + (4 \times 2) + (4 \times 3) + (5 \times 4)$
   $= 43$
7. $h(2,3)$
   $= (4 \times 1) + (5 \times 2) + (5 \times 3) + (6 \times 4)$
   $= 53$
8. $h(2,4)$
   $= (5 \times 1) + (6 \times 2) + (6 \times 3) + (7 \times 4)$
   $= 63$
9. $h(3,1)$
   $= (3 \times 1) + (4 \times 2) + (4 \times 3) + (5 \times 4)$
   $= 43$
10. $h(3,2)$
    $= (4 \times 1) + (5 \times 2) + (5 \times 3) + (6 \times 4)$
    $= 53$

11. $h(3,3)$
    $= (5 \times 1) + (6 \times 2) + (6 \times 3) + (7 \times 4)$
    $= 63$
12. $h(3,4)$
    $= (6 \times 1) + (7 \times 2) + (7 \times 3) + (8 \times 4)$
    $= 73$
13. $h(4,1)$
    $= (4 \times 1) + (5 \times 2) + (5 \times 3) + (6 \times 4)$
    $= 53$
14. $h(4,2)$
    $= (5 \times 1) + (6 \times 2) + (6 \times 3) + (7 \times 4)$
    $= 63$
15. $h(4,3)$
    $= (6 \times 1) + (7 \times 2) + (7 \times 3) + (8 \times 4)$
    $= 73$
16. $h(4,4)$
    $= (7 \times 1) + (8 \times 2) + (8 \times 3) + (9 \times 4)$
    $= 83$

Hasil operasi konvoluasi tersebut ditunjukkan pada gambar 4.10.

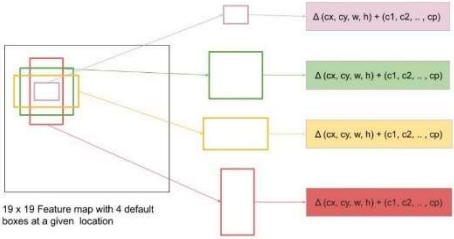
23	33	43	53
33	43	53	63
43	53	63	73
53	63	73	83

Gambar 4.10. Hasil Ilustrasi Operasi Konvolusi

c. Evaluasi *Default Box*

Default box merupakan kotak yang memiliki berbagai aspek rasio yang berada pada tiap lokasi dari beberapa *feature maps* dengan ukuran yang berbeda. Selama *training*, *default box* dibandingkan dengan *ground truth box* dan memilih *default box*

yang paling mendekati dengan *ground truth box*. Ilustrasi *default box* pada *feature map* ditunjukkan pada gambar 4.11.



Gambar 4.11. Ilustrasi *Default Box* pada 19×19 *Feature Map* [23]

4.2.4 Perancangan Tahap Klasifikasi

Pada tahap ini dilakukan klasifikasi jenis kerusakan jalan. Berikut merupakan penjelasan masing-masing proses yang akan dilakukan pada tahap klasifikasi:

a. Input Video Uji Coba

Pada proses inisialisasi tahap klasifikasi diberikan input berupa data video kerusakan jalan yang tidak digunakan dalam data training.

b. Input *Trained Weight*

Selain input video, juga diberikan input *trained weight*. *Trained weight* merupakan bobot yang dihasilkan pada tahap *training*.

c. Pengubahan Video menjadi *Frame*

Pada langkah awal proses klasifikasi, data video input diubah menjadi *frame* dan diproses per *frame*.

d. Proses pada *Base Network*

Setelah video diubah menjadi *frame*, dilakukan ekstraksi fitur per *frame* sesuai dengan jumlah *frame*

yang ada. Pada tahap ekstraksi fitur dilakukan proses konvolusi untuk menghasilkan *feature maps*.

e. Pemberian *Default Box*

Setelah dilakukan ekstraksi fitur, diberikan *default box* sesuai dengan jenis kerusakan jalan yang ada pada *frame*.

f. Pengubahan *Frame* menjadi Video

Setelah semua *frame* pada video diproses, selanjutnya semua *frame* tersebut diubah kembali menjadi video.

g. Hasil Klasifikasi

Hasil dari tahap klasifikasi ini berupa video yang berisi *default box* dan label jenis kerusakan jalan. Selain video juga terdapat informasi seberapa banyak kerusakan jalan pada video tersebut.

4.3 Implementasi Sistem

Pada sub bab ini akan dibahas langkah-langkah implementasi sistem berdasarkan tahap pada perancangan sistem. *Source code* pada tugas akhir ini merujuk pada penelitian yang berjudul "*high quality*, *fast*, *modular reference implementation of* SSD *in* pytorch" yang dilakukan oleh Congcong Li pada tahun 2018 [21]. Implementasi menggunakan bahasa pemrograman *python* yang meliputi implementasi antaramuka (*interface*), tahap *pre training*, tahap *training*, dan tahap klasifikasi.

4.3.1 Implementasi Antarmuka (*Interface*)

Pada tugas akhir ini, antaramuka dibuat menggunakan *library python* streamlit yang menghasilkan antarmuka web (web *interface*) yang dapat diakses offline menggunakan *local*

host. Antarmuka dibuat untuk mempermudah implementasi klasifikasi.

4.3.2 Implementasi Tahap *Pre Training*

Sebelum memasuki tahap *training*, data untuk *training* diolah terlebih dahulu pada tahap *pre training*. Tahap *pre training* terdiri dari tahap ekstraksi *frame*, tahap pengurutan dan *rename* kumpulan *frame*, dan pembuatan anotasi (*labeling*). Berikut merupakan penjelasan terkait tahap *pre training* yang dilakukan:

a. Tahap Ekstraksi *Frame*

Proses ekstraksi *frame* dilakukan dengan memanfaatkan *python library* opencv yang memiliki fungsi *imwrite* yang mampu mengambil *frame* yang memiliki kerusakan jalan.

b. Tahap Pengurutan dan *Rename* Kumpulan *Frame*

Proses pengurutan hasil ekstraksi *frame* dilakukan dengan *python library* shutil dan os yang masing-masing berfungsi untuk menyalin file dan mengubah nama file sesuai urutan.

c. Pembuatan Anotasi (*Labeling*)

Pembuatan anotasi dilakukan dengan *python library* labelme dan file labelme2coco.py. Data input merupakan *frame* dari beberapa video yang kemudian diberi *box* dan label jenis kerusakan jalan. Hasil *labeling* tersebut disimpan dengan format *.json. Setelah semua gambar diberi label dengan labelme, selanjutnya adalah pengubahan anotasi dari labelme menjadi format dataset coco (*common object in context*). Format dataset coco merupakan salah format yang umum digunakan pada deteksi objek sehingga pada tugas akhir ini digunakan format dataset coco.

Pengubahan anotasi dari labelme menjadi format dataset coco dilakukan dengan menjalankan file labelme2coco.py yang mana file tersebut mengabungkan setiap informasi pada file anotasi .json pada setiap gambar menjadi satu file .json yang sesuai dengan format dataset coco.

Berikut merupakan inisialisasi tahap pembuatan anaotasi (*labeling*):

Fungsi : Inisialisasi tahap pembuatan anotasi (*labeling*)
Input : *Frame* beberapa video dan hasil anotasi labelme
Output : Hasil anotasi labelme dan gabungan hasil anotasi labelme (format dataset coco)

Command Line :
Inisialisasi labelme: pip install labelme
labelme (pada terminal)

Inisialisasi labelme2coco.py:
python labelme2coco.py
ssd/data/datasets/train/
--output
ssd/data/datasets/annotations/train.json

4.3.3 Implementasi Tahap *Training*

Tahap *training* bertujuan untuk menghasilkan *trained weight* yang akan digunakan pada tahap klasifikasi. Fungsi inisialiasi tahap *training* diimplementasikan sebagai berikut:

Fungsi	: Inisialisasi tahap *training*
Input	: Data *training* dan config file
Output	: *Trained weight*
Command Line	:

python train.py --config-file configs/config.yaml
(terminal)

!python train.py --config-file configs/config.yaml
(*google colaboratory*)

Pada *command line* tersebut fungsi python train.py bertujuan untuk inisialisasi tahap *training* pada terminal, jika menggunakan *google colaboratory* maka menggunakan *command line* !python train.py. Pada fungsi tersebut terdapat atribut config-file yang merupakan konfigurasi yang digunakan pada tahap *training*. Pada tugas akhir ini, konfigurasi yang digunakan merujuk pada konfigurasi penelitian Wei Liu, dkk yang berjudul "SSD: *Single Shot Multibox Detector*" [15]. Berikut ini merupakan implementasi tahap *training* yang dilakukan:

a. Input Data *Training*

Ketika melakukan input data *training*, hal pertama yang perlu dilakukan adalah mengumpulkan semua gambar yang akan digunakan pada tahap *training* dalam dua folder (dalam tugas akhir ini nama folder yang digunakan adalah "train" dan "test"). Folder tersebut berisi gambar dan file anotasi (.json).

Selanjutnya, mengubah file coco.py pada folder ssd/data/datasets menjadi my_dataset.py. File tersebut berisi fungsi untuk membaca gambar dan file anotasi pada folder yang telah ditentukan, selain itu file tersebut berisi jenis kerusakan jalan yang akan diklasifikasi. Untuk menambahkan folder yang akan digunakan dalam tahap *training* (folder train dan test), maka lokasi folder tersebut perlu ditambahkan pada file path_catlog.py yang berada pada folder ssd/config/

Berikut merupakan implementasi dari input data *training*:

Source Code :

```
import os
class DatasetCatalog:
    DATA_DIR = 'datasets'
    DATASETS = {
            'my_custom_dataset': {
                    'data_dir': 'train',
                    'ann_file': 'annotations/train.json'
            }, }
def get(name):
    if 'my_custom_dataset' in name:
        my_custom_dataset =
        DatasetCatalog.DATA_DIR
        attrs = DatasetCatalog.DATASETS[name]
        args = dict(
                data_dir =
                os.path.join(my_custom_dataset,
                attrs['data_dir']),
                ann_file =
                os.path.join(my_custom_dataset,
                attrs['ann_file']),)
        return dict(factory='MyDataset', args=args)
```

b. *Build Detection Model*

Build detection model merupakan proses untuk membangun model *single shot detection* (SSD) untuk digunakan pada tahap *training*. Berikut merupakan implementasi *build detection model*:

Fungsi	: *Build detection model*
Input	: Konfigurasi file
Output	: Model SSD
Source Code	:

```
model = build_detection_model(cfg)
```

c. *Make Data Loader*

Make data loader merupakan fungsi untuk memasukkan gambar yang digunakan pada tahap *training*. Berikut merupakan implementasi dari *make data loader*:

Fungsi	: *Make data loader*
Input	: Konfigurasi file
Output	: Data *loader*
Source Code	:

```
train_loader = make_data_loader(cfg,
is_train=True, distributed=args.distributed,
max_iter=max_iter,
start_iter=arguments['iteration']}
```

Pada *source code* tersebut cfg merupakan konfigurasi file yang berisi konfigurasi dari model yang akan digunakan. is_train merupakan argumen yang digunakan sebagai penanda untuk melakukan

transformasi pada gambar *training* seperti *random contrast* dan *random brightness*. distributed merupakan argumen yang digunakan sebagai penanda apakah proses *training* dibagi pada beberapa GPU atau tidak, sedangkan args.distributed merupakan argumen yang digunakan sebagai penanda banyaknya GPU yang digunakan. Jika banyaknya GPU yang digunakan lebih dari satu maka args.distributed bernilai *True* dan distributed=*True*, sedangkan jika banyaknya GPU yang digunakan sama dengan satu maka args.distributed bernilai *False* dan distributed=*False*. Dalam tugas akhir ini hanya menggunakan satu GPU sehingga distributed=*False*. max_iter dan start_iter merupakan iterasi maksimum dan awal iterasi.

d. *Do Train*

Do train merupakan fungsi untuk melakukan proses *training* setelah model dan data *training* siap untuk digunakan. Berikut merupakan implementasi dari *do train*:

Fungsi	: Proses *training*
Input	: Konfigurasi file, model dan train_loader
Output	: *Trained weight*
Source Code	:

```
model_train = do_train(cfg, model, train_loader,
optimizer, scheduler, checkpointer, device,
arguments, args)
```

Pada *source code* tersebut cfg merupakan konfigurasi file yang berisi konfigurasi dari model

yang akan digunakan. model merupakan model deteksi yang digunakan yaitu *single shot detector* (SSD). train_loader merupakan fungsi yang memasukkan data *training* pada model. optimizer merupakan fungsi yang mengoptimalkan proses seiring banyaknya GPU yang digunakan. scheduler merupakan fungsi untuk mengupdate parameter sehingga model dapat menghasilkan output yang mendekati dengan *ground truth*. checkpointer merupakan fungsi untuk menyimpan *checkpoint* ketika melakukan proses *training* sehingga proses *training* dapat dimulai dari *checkpoint* tersebut. device merupakan fungsi untuk menentukan menggunakan GPU atau CPU pada proses *training*. arguments digunakan untuk menentukan awal dari iterasi, jika belum ada *checkpoint* maka iterasi dimulai nol, sedangkan jika terdapat *checkpoint* maka iterasi dimulai dari *checkpoint* tersebut. args merupakan argumen yang diinputkan pada *command line* yaitu konfigurasi file.

4.3.4 Implementasi Tahap Klasifikasi

Tahap klasifikasi bertujuan untuk mengklasifikasikan jenis kerusakan jalan dengan metode *single shot detector* (SSD). Fungsi inisialisasi tahap klasifikasi diimplementasikan sebagai berikut:

Fungsi	: Inisialisasi tahap klasifikasi
Input	: Video, konfigurasi file, dan *trained weight*
Output	: Video dengan *default box* dan label
Command Line	:
streamlit run test.py	

Command line tersebut diakses melalui terminal (pada linux) atau anaconda *prompt* dan sejenisnya (pada windows). Berikut ini merupakan implementasi tahap klasifikasi yang dilakukan:

a. Input Video Uji Coba

Semua video uji coba diletakkan pada folder input yang nantinya program akan mengakses folder input tersebut untuk memasukkan video. Hal ini dilakukan karena *python library* streamlit belum memiliki *file uploader* sebagai sarana untuk memilih file. Berikut merupakan implementasi input video uji coba:

Fungsi	: Input video uji coba
Source code	:
video = video_uploader(‘./input’)	
clip = VideoFileClip(video)	

video_uploader merupakan fungsi yang digunakan untuk mencari setiap file pada folder input dan output dari fungsi tersebut merupakan lokasi dari video beserta nama dan ekstensi video tersebut (contoh: ‘./input/video.mp4’). Sedangkan VideoFileClip merupakan fungsi dari *python library* moviepy yang memiliki fungsi untuk membaca file video sehingga video dapat diproses.

b. Input *Trained Weight*

Setelah dilakukan proses *training*, *trained weight* disimpan dalam sebuah folder. Folder *trained weight* tersebut berada dalam folder outputs. Folder *trained weight* tersebut berisi file *trained_weight*.pth dan last_checkpoint.txt yang berisi informasi *checkpoint* terakhir yang disimpan pada file *trained_weight*.pth.

Berikut merupakan implementasi dari input *trained weight*:

Fungsi	: Input *trained weight*
Source code	:
output dir = checkpoint folder('./outputs')	

checkpoint_folder merupakan fungsi yang digunnakan untuk mencari setiap folder yang terdapat pada folder outputs yang mana folder tersebut merupakan folder yang menyimpan folder *trained weight*.

c. Perubahan Video menjadi *Frame*

Sebelum video diproses, video tersebut dirubah dulu menjadi *frame* sehingga *frame* tersebut yang akan diolah oleh model. Berikut merupakan implementasi dari perubahan video menjadi *frame*:

Fungsi	: Perubahan video menjadi *frame*
Source code	:
for frame in clip.iter_frames()	

iter_frames() merupakan fungsi dari *python library* moviepy yang dapat melakukan iterasi per frame pada video tanpa menyimpan tiap frame terlebih dahulu pada sebuah array atau folder tertentu.

d. Pemberian *Default Box*

Setelah proses mengubah video menjadi *frame*, langkah selanjutnya adalah memberi *default box* dan label jenis kerusakan jalan pada video tersebut.

Berikut merupakan implementasi dari pemberian *default box*:

Fungsi	: Pemberian *default box*
Source code	:
drawn_frame = draw_boxes(frame, boxes, labels, scores, class_name).astype(np.uint8)	

draw_boxes merupakan fungsi dari *python library* vizer yang berfungsi untuk memberikan *default box* dan label. frame merupakan salah satu *frame* pada video yang akan diolah. boxes merupakan *default box* prediksi pada *frame* tersebut. labels merupakan label prediksi pada *frame* tersebut. scores merupakan tingkat keyakinan (*confidence*) model terhadap prediksinya. class_name merupakan nama-nama jenis kerusakan jalan yang diklasifikasi. astype() merupakan fungsi dari *python library* numpy yang digunakan untuk menggubah tipe array. np.uint8 merupakan tipe data yang digunakan pada rentang integer yang spesifik yaitu 0 hingga 255.

e. Perubahan *Frame* menjadi Video

Setelah semua *frame* diproses, langkah selanjutnya ada mengubah *frame* menjadi video. Berikut merupakan implementasi dari perubahan *frame* menjadi video:

Fungsi	: Perubahan *frame* menjadi video
Source code	:
out = cv2.VideoWriter(output_video, cv2.VideoWriter_fourcc(*'MJPG'), 20.0, (width, height))	
out.write(drawn_frame)	

VideoWriter merupakan fungsi dari opencv yang digunakan untuk membuat video dari beberapa gambar/*frame*. VideoWriter_fourcc merupakan *codec* yang digunakan untuk memproses video. 20.0 merupakan fps (*frame per second*) dari video output. (width, height) merupakan tinggi dan lebah dari video output.

Python library streamlit masih belum dapat menampilkan hasil video dari opencv sehingga sebelum ditampilkan pada tampilan antarmuka (*interface*) perlu diformat terlebih dahulu sehingga dapat ditampilkan di tampilan antarmuka (*interface*). Berikut merupakan implementasinya:

Fungsi	: Perubahan *frame* menjadi video
Source code	:

```
result_clip = VideoFileClip(temp_name)
result_clip.write_videofile(video_filename)
```

VideoFileClip merupakan fungsi dari *python library* moviepy yang digunakan untuk memproses video. temp_name merupakan *temporary file* yang digunakan untuk menyimpan video output opencv sebelum diubah menjadi format yang dapat dibaca oleh *python library* streamlit. write_videofile merupakan fungsi dari *python library* moviepy yang digunakan untuk membuat video baru dari video yang sudah ada. video_filename merupakan nama yang digunakan pada video output, nama output tersebut memiliki format Tahun-Bulan-Tanggal_Jam-Menit-Detik_Konfigurasi.

BAB V
UJI COBA DAN PEMBAHASAN

Pada bab ini berisi hasil uji coba yang dihasilkan oleh implementasi sistem dan pembahasan terhadap hasil uji coba yang dilakukan.

5.1 Lingkungan Sistem

Pada implementasi sistem digunakan dua jenis lingkungan, yaitu *hardware* dan *software*. Spesifikasi lingkungan implementasi sistem secara lengkap sebagai berikut:

Tabel 5.1. Spesifikasi *Hardware* dan *Software*

Lingkungan		Spesifikasi
Hardware	CPU	Intel Core i7 4720HQ
	GPU	Nvidia GeForce GTX 950M
	RAM	8 GB DDR3L
Software	OS	Elementary OS 5.1.4
	Tools	Python 3.6
		Terminal
		Labelme
		Streamlit
		Google Colaboratory

5.2 Dataset Uji Coba

Uji coba pada program tugas akhir ini dilakukan terhadap video dengan format *.mp4. Jumlah data yang digunakan dalam uji coba ini adalah 4 video. Video tersebut diambil ketika cuaca mendung. Pengaturan kamera yang digunakan pada saat pengambilan video sebagai berikut: *shutter speed*: 1/6400, *aperture*: 4.5, ISO: 6400. *Shutter speed* merupakan kecepatan kamera dalam mengambil gambar sehingga dapat mengurangi *blur* pada gambar, *shutter speed* 1/6400 dipilih agar tidak terjadi *motion blur*.

Sedangkan *aperture* merupakan suatu mekanisme yang mengatur banyaknya cahaya yang masuk dari lensa ke sensor kamera, semakin kecil nilanyanya semakin banyak cahaya yang masuk dan semakin kecil titik fokusnya. *Aperture* 4.5 dipilh agar titik fokusnya tidak terlalu kecil dan cahaya yang masuk ke sensor kamera tidak kurang. ISO merupakan sensitivitas sensor kamera terhadap cahaya, semakin tinggi nilainya semakin sensitif kamera terhadap cahaya, tetapi jika nilainya terlalu tinggi maka akan muncul *grain* pada gambar atau video. ISO 6400 dipilih agar *grain* tidak muncul pada hasil video dan video yang hasilnya masih dapat terlihat dengan baik.

Kerusakan jalan yang terdapat pada video uji coba dapat dilihat pada table 5.2.

Tabel 5.2. Jumlah Kerusakan Jalan Pada Tiap Video Uji Coba

Video	Jenis Kerusakan Jalan	Jumlah Kerusakan Jalan
Video Uji Coba 1	Retak Aligator	3
	Retak Memanjang	29
	Retak Melintang	11
	Lubang	2
Video Uji Coba 2	Retak Aligator	8
	Retak Memanjang	6
	Retak Melintang	4
	Lubang	2
Video Uji Coba 3	Retak Aligator	8
	Retak Memanjang	15
	Retak Melintang	1
	Lubang	4
Video Uji Coba 4	Retak Aligator	22
	Retak Memanjang	46
	Retak Melintang	12

Video	Jenis Kerusakan Jalan	Jumlah Kerusakan Jalan
Video Uji Coba 4	Lubang	22

Video uji coba 4 memiliki kerusakan jalan yang berukuran kecil (sekitar kurang dari 5 cm ukuran asli) dan kerusakan jalan berukuran besar sehingga melebihi luas *frame* paling banyak. Sedangkan video uji coba 3 memiliki kerusakan jalan yang berukuran kecil paling sedikit dan video uji coba 1 tidak memiliki kerusakan jalan berukuran besar.

5.3 Pengujian Tahap *Pre Training*

Pada tahap ini dilakukan pengujian proses *pre training* untuk mengatahui bahwa proses yang dilakukan pada tahap *pre training* sudah benar sehingga hasil dapat digunakan sebagai input pada tahap selanjutnya. Tahap *pre training* terdiri dari tiga proses, yaitu: ekstraksi *frame*, pengurutan dan *rename* kumpulan *frame*, dan pembuatan anotasi (*labeling*).

5.3.1 Tahap Ekstraksi *Frame*

Tahap ekstraksi *frame* merupakan proses untuk mengambil *frame* yang terdapat kerusakan jalan pada video dengan menggunakan *python library* opencv. Tahap ini menggunakan tiga video *training*, yang mana ketiga video tersebut berbeda dengan video uji coba. *Frame* tersebut disimpan pada folder sesuai dengan nama video.

5.3.2 Tahap Pengurutan dan *Rename* Kumpulan *Frame*

Pada tahap ini dilakukan penggabungan hasil ekstraksi *frame* dari video *training* serta dilakukan pengurutan *frame* tersebut. Semua *frame* dimasukkan pada sebuah *array* yang nantinya akan diurutkan menggunakan *python library* natsort yang berfungsi untuk mengurutkan secara natural. *Python library* tersebut diperlukan karena pada proses pengurutan dua digit atau lebih terdapat masalah,

contoh: proses pengurutan angka 1 hingga 10, jika tanpa natsort menjadi 1, 10, 2, ..., 9, sedangkan jika menggunakan natsort menjadi 1, 2, 3, ..., 10.

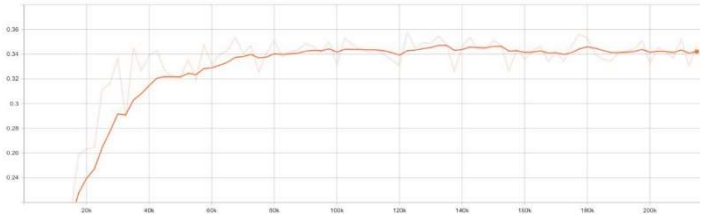
5.3.3 Tahap Pembuatan Anotasi (*Labeling*)

Tahap pembuatan data anotasi merupakan proses pemberian label jenis kerusakan jalan. Proses pembuatan anotasi dilakukan pada seluruh file dengan memberikan label sesuai dengan jenis kerusakan jalan. Hasil dari tahap ini berupa file dengan format *.json.

5.4 Pengujian *Trained Weight*

Pada tahap ini dilakukan pengujian terhadap *trained weight*. Setelah dilakukan *training*, langkah selanjutnya adalah melakukan evaluasi dengan meninjau grafik *loss training*, grafik loss validasi, dan grafik *mean average precision* (mAP) selama proses *training*. Grafik *loss training*, grafik *loss* validasi, dan grafik mAP yang dihasilkan selama proses *training* ditunjukkan pada gambar 5.1 dan gambar 5.2.

Gambar 5.1. Grafik *Loss Training* (atas) dan Grafik *Loss* Validasi (bawah)



Gambar 5.2. Grafik mAP

Berdasarkan gambar 5.2 didapatkan mAP tertinggi pada iterasi ke 180000. Proses *training* memerlukan proses yang lama dikarenakan jumlah data *training* yang relatif sedikit (kurang dari 1000 citra) sehingga model memerlukan waktu yang lama untuk mempelajari pola pada data *training*. Berdasarkan hasil kinerja di atas, pada tugas akhir ini menggunakan *trained weight* pada iterasi ke 180000.

5.5 Pengujian Tahap Klasifikasi

Pada tahap ini dilakukan pengujian terhadap proses klasifikasi untuk mengetahui kemampuan sistem dalam mengklasifikasikan jenis kerusakan jalan.

5.5.1 Evaluasi *Counting* Program

Evaluasi *counting* merupakan proses untuk mengecek perhitungan kerusakan jalan pada video. Seperti yang dijelaskan pada bab sebelumnya, proses klasifikasi dilakukan tiap *frame* pada video sehingga jika proses perhitungan dilakukan tiap *frame* juga maka hasil klasifikasi akan lebih banyak dari kerusakan jalan yang ada pada video tersebut. Sehingga pada proses ini dilakukan pengujian *counting* untuk menentukan tiap berapa *frame* untuk melakukan perhitungan.

Untuk proses ini dilakukan pengecekan tiap 10, 15, 20, 25, dan 30 *frame* untuk proses *counting*. Hal ini dilakukan agar tidak terjadi perhitungan lebih dari sekali. Contoh perhitungan ganda dapat dilihat pada gambar 5.3

Gambar 5.3. Pendeteksian Ganda pada *Frame* ke-10 (kiri) dan *Frame* ke-20 (kanan)

Setelah dilakukan pengecekkan didapatkan bahwa perhitungan tiap 20 *frame* merupakan solusi yang optimal karena memiliki pendeteksian ganda yang sedikit (lebih sedikit daripada tiap 10 *frame* dan tiap 15 *frame*) dan jumlah yang terdeteksi banyak (lebih banyak daripada tiap 25 *frame* dan tiap 30 *frame*).

Selanjutnya akan dilakukan evaluasi *counting* program tiap 20 *frame* untuk keempat video uji coba. Berikut merupakan perbandingan jumlah kerusakan jalan yang terhitung oleh program dan jumlah kerusakan jalan yang sebenarnya:

Tabel 5.3. Hasil Evaluasi *Counting*

Video	Jenis Kerusakan Jalan	Jumlah Kerusakan Jalan Terhitung	Jumlah Kerusakan Jalan Sebenarnya
Video Uji Coba 1	Retak Aligator	2	3
	Retak Memanjang	4	29
	Retak Melintang	8	11
	Lubang	1	2

Video	Jenis Kerusakan Jalan	Jumlah Kerusakan Jalan Terhitung	Jumlah Kerusakan Jalan Sebenarnya
Video Uji Coba 2	Retak Aligator	14	8
	Retak Memanjang	5	6
	Retak Melintang	1	4
	Lubang	0	2
Video Uji Coba 3	Retak Aligator	21	8
	Retak Memanjang	7	15
	Retak Melintang	1	1
	Lubang	2	4
Video Uji Coba 4	Retak Aligator	23	22
	Retak Memanjang	13	46
	Retak Melintang	4	12
	Lubang	5	22

5.5.2 Evaluasi Tahap Klasifikasi

Setelah dilakukan klasifikasi, kemudian dilakukan evaluasi dengan menghitung presisi, *recall*, dan akurasi. Evaluasi dilakukan dengan menghitung kerusakan jalan yang terdapat setiap 20 *frame* (*frame* ke-20, *frame* ke-40, dan seterusnya). Hasil *true positive*, *true negative*, *false positive*, *false negative* dari masing-masing jenis kerusakan jalan pada video uji coba dapat dilihat pada table 5.4.

Tabel 5.4. Hasil *True Positive*, *True Negative*, *False Positive*, *False Negative*

Video	Jenis Kerusakan Jalan	TP, TN, FP, FN	Total
Video Uji Coba 1	Retak Aligator	*True Positive*	2
		True Negative	11
		False Positive	0
		False Negative	1
	Retak Memanjang	*True Positive*	4
		True Negative	9
		False Positive	1
		False Negative	25
	Retak Melintang	*True Positive*	6
		True Negative	7
		False Positive	1
		False Negative	5
	Lubang	*True Positive*	1
		True Negative	12
		False Positive	0

Video	Jenis Kerusakan Jalan	TP, TN, FP, FN	Total
Video Uji Coba 1	Lubang	*False Negative*	1
Video Uji Coba 2	Retak Aligator	*True Positive*	7
		True Negative	3
		False Positive	7
		False Negative	1
	Retak Memanjang	*True Positive*	2
		True Negative	8
		False Positive	2
		False Negative	4
	Retak Melintang	*True Positive*	1
		True Negative	9
		False Positive	0
		False Negative	3
	Lubang	*True Positive*	0
		True Negative	10
		False Positive	0
		False Negative	2

Video	Jenis Kerusakan Jalan	TP, TN, FP, FN	Total
Video Uji Coba 3	Retak Aligator	*True Positive*	8
		True Negative	7
		False Positive	10
		False Negative	0
	Retak Memanjang	*True Positive*	4
		True Negative	11
		False Positive	2
		False Negative	11
	Retak Melintang	*True Positive*	1
		True Negative	14
		False Positive	0
		False Negative	0
	Lubang	*True Positive*	2
		True Negative	13
		False Positive	0
		False Negative	2
Video Uji Coba 4	Retak Aligator	*True Positive*	10

Video	Jenis Kerusakan Jalan	TP, TN, FP, FN	Total
Video Uji Coba 4	Retak Aligator	*True Negative*	14
		False Positive	8
		False Negative	12
	Retak Memanjang	*True Positive*	8
		True Negative	16
		False Positive	3
		False Negative	38
	Retak Melintang	*True Positive*	2
		True Negative	22
		False Positive	2
		False Negative	10
	Lubang	*True Positive*	4
		True Negative	20
		False Positive	0
		False Negative	18

Berikut merupakan contoh perhitungan dari presisi, *recall*, dan akurasi pada video uji coba 1:

Retak aligator

$$presisi = \frac{2}{(2+0)} \times 100\% = 100\%$$

$$recall = \frac{2}{(2+1)} \times 100\% = 66.67\%$$

$$akurasi = \frac{(2+11)}{(2+11+0+1)} \times 100\% = 48.15\%$$

Retak memanjang

$$presisi = \frac{4}{(4+1)} \times 100\% = 80\%$$

$$recall = \frac{4}{(4+25)} \times 100\% = 13.79\%$$

$$akurasi = \frac{(4+9)}{(4+9+1+25)} \times 100\% = 33.33\%$$

Retak melintang

$$presisi = \frac{6}{(6+1)} \times 100\% = 85.71\%$$

$$recall = \frac{6}{(6+5)} \times 100\% = 54.55\%$$

$$akurasi = \frac{(6+7)}{(6+7+1+5)} \times 100\% = 68.42\%$$

Lubang

$$presisi = \frac{1}{(1+0)} \times 100\% = 100\%$$

$$recall = \frac{1}{(1+1)} \times 100\% = 50\%$$

$$akurasi = \frac{(1+12)}{(1+12+0+1)} \times 100\% = 92.86\%$$

Berdasarkan tabel 5.4 dan contoh perhitungan sebelumnya didapatkan presisi, *recall*, dan akurasi sebagai berikut:

Tabel 5.5. Presisi, *Recall*, dan Akurasi

Video	Jenis Kerusakan Jalan	Presisi (%)	*Recall* (%)	Akurasi (%)
Video Uji Coba 1	Retak Aligator	100	66.67	48.15
	Retak Memanjang	80	13.79	33.33
	Retak Melintang	85.71	54.55	68.42
	Lubang	100	50	92.86
Video Uji Coba 2	Retak Aligator	50	87.5	55.56
	Retak Memanjang	50	33.33	62.5
	Retak Melintang	100	25	76
	Lubang	0	0	83.33
Video Uji Coba 3	Retak Aligator	44.44	100	60
	Retak Memanjang	66.67	26.67	53.57
	Retak Melintang	100	100	100
	Lubang	100	50	88.24
Video Uji Coba 4	Retak Aligator	55.56	45.45	54.54
	Retak Memanjang	72.73	17.39	36.92
	Retak Melintang	50	16.67	66.67

Video	Jenis Kerusakan Jalan	Presisi (%)	*Recall* (%)	Akurasi (%)
Video Uji Coba 4	Lubang	100	18.18	57.14

5.6 Pembahasan Hasil Klasifikasi

Pada tahap ini akan dibahas hasil evaluasi tahap klasifikasi. Berikut merupakan rata-rata hasil evaluasi sistem:

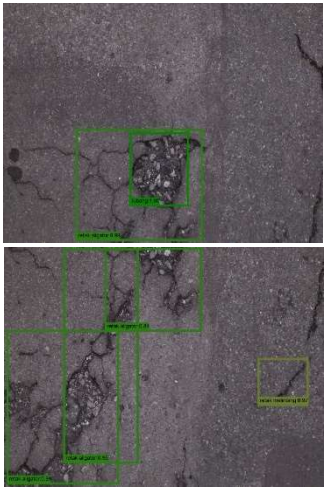
Tabel 5.6. Rata-rata Hasil Evaluasi Sistem

Video	Presisi (%)	*Recall* (%)	Akurasi (%)
Video Uji Coba 1	91.43	46.25	60.69
Video Uji Coba 2	50	36.45	69.58
Video Uji Coba 3	77.78	69.17	75.45
Video Uji Coba 4	69.57	24.42	53.81

Berdasarkan tabel 5.6 diketahui bahwa hasil terbaik dihasilkan oleh video uji coba 3 yang memiliki presisi sebesar 77.78%, *recall* sebesar 69.17%, dan akurasi sebesar 75.45%. Pembeda dari keempat video uji coba adalah banyaknya kerusakan jalan yang berukuran kecil. Sehingga, tidak terdeteksi oleh sistem dan menyebabkan penurunan hasil evaluasi klasifikasi. Hal tersebut dapat terjadi karena pada data video yang digunakan untuk *training* tidak terdapat kerusakan jalan yang berukuran kecil sehingga model SSD tidak dapat mendeteksi kerusakan jalan tersebut.

Gambar 5.4. Perbandingan Kerusakan Jalan yang Terdeteksi (kiri) dan yang Tidak Terdeteksi (kanan)

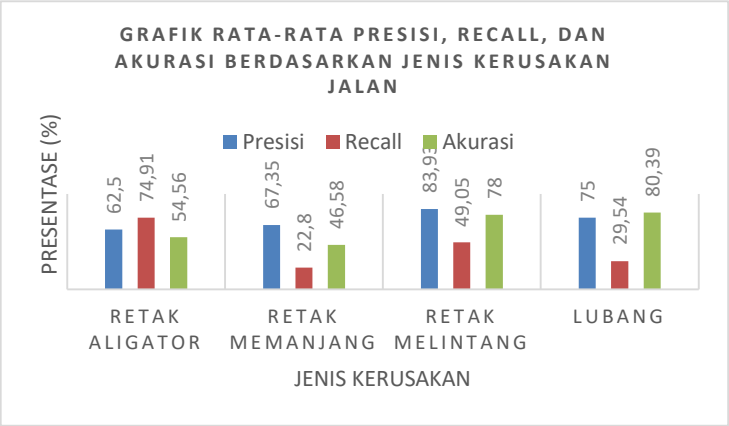
Selain itu, ukuran kerusakan jalan yang lebih besar dari luas *frame* sehingga akhirnya terbagi dalam beberapa *frame* menyebabkan peningkatkan jumlah *counting* kerusakan jalan dan terhitung dalam *false positive* dan menyebabkan hasil evaluasi menurun.



Gambar 5.5. Contoh Retak Aligator yang Terbagi Dalam Dua *Frame*

Pada gambar 5.5 terlihat bahwa satu retak aligator yang terbagi dalam dua *frame* terdeteksi sebanyak empat kali oleh sistem, hal ini merupakan salah satu penyebab hasil evaluasi sistem menurun.

Perhitungan rata-rata presisi, *recall*, dan akurasi seluruh video uji coba berdasarkan jenis kerusakan jalan ditunjukkan pada gambar 5.6.



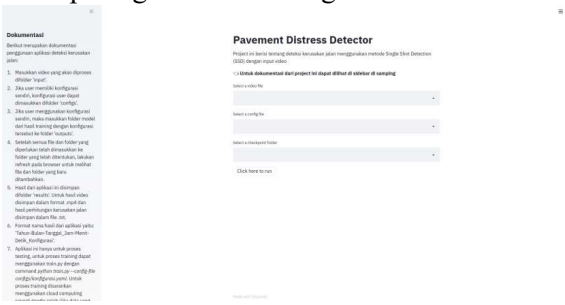
Gambar 5.6. Grafik Rata-rata Presisi, *Recall*, dan Akurasi Berdasarkan Jenis Kerusakan Jalan

Berdasarkan gambar 5.6 diketahui bahwa nilai rata-rata presisi terbaik terdapat pada jenis kerusakan jalan lubang dengan nilai 75%. Nilai rata-rata *recall* terbaik terdapat pada jenis kerusakan jalan retak alligator dengan nilai 74.91%. Nilai akurasi terbaik terdapat pada jenis kerusakan jalan lubang dengan nilai 80.39%. Sedangkan nilai rata-rata presisi terendah terdapat pada jenis kerusakan jalan retak alligator dengan nilai 62.5%. Nilai rata-rata *recall* terendah terdapat pada jenis kerusakan jalan retak memanjang dengan nilai 22.8%. Nilai rata-rata akurasi terendah terdapat pada jenis kerusakan jalan retak memanjang dengan nilai 46.48%.

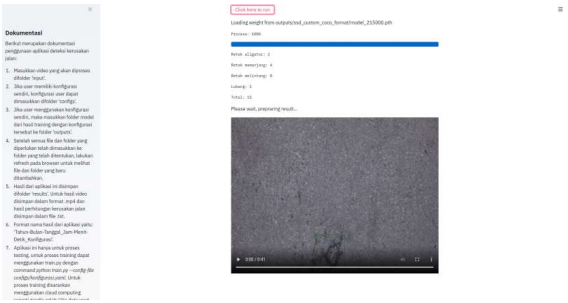
Hasil tersebut menunjukkan bahwa sistem dapat menemukan lubang secara baik, terbukti dengan nilai rata-rata presisi tertinggi pada lubang. Sistem dapat memisahkan retak aligator secara baik, terbukti dengan nilai rata-rata *recall* tertinggi pada retak aligator.

Rendahnya hasil evaluasi tahap klasifikasi disebabkan oleh kurangnya variasi ukuran kerusakan jalan pada data *training* dan *field of view* (bidang pandang) dari kamera yang terlalu sempit sehingga tidak mencakup satu kerusakan jalan yang berukuran besar dalam satu *frame*. Tetapi dengan kekurangan seperti itu, performa klasifikasi jenis kerusakan jalan retak aligator tidaklah buruk. Sehingga *trained weight* pada tugas akhir ini lebih tepat digunakan untuk mengklasifikasikan jenis kerusakan jalan retak aligator.

Hasil pengujian sistem antarmuka klasifikasi jenis kerusakan jalan terbagi menjadi: tampilan awal (sebelum dilakukan proses klasifikasi) dan tampilan setelah dilakukan klasifikasi. Untuk untuk gambar tampilan antarmuka ditunjukkan pada gambar 5.7 dan gambar 5.8.



Gambar 5.7. Tampilan Antarmuka Awal



Gambar 5.8. Tampilan Antarmuka Setelah Klasifikasi

BAB VI
PENUTUP

Bab ini berisi tentang beberapa kesimpulan berdasarkan penelitian yang telah dilaksanakan. Selain itu, pada bab ini juga dimasukkan beberapa saran yang dapat digunakan jika penelitian ini ingin dikembangkan.

6.1 Kesimpulan

Berdasarkan analisis terhadap hasil pengujian program, maka dapat diambil kesimpulan sebagai berikut:

1. Sistem dengan model *single shot detector* (SSD) dapat mengklasifikasikan jenis kerusakan dengan tahap *pre training*, *training*, dan klasifikasi.
2. Sistem dapat mengklasifikasi jenis kerusakan jalan dengan rata-rata akurasi 75.45% dengan kondisi jumlah kerusakan jalan berukuran kecil (sekitar kurang dari 5 cm ukuran asli).

6.2 Saran

Dengan melihat hasil yang dicapai pada penelitian ini terdapat beberapa hal yang penulis sarankan untuk dikembangkan yaitu:

1. Menambahkan data *training* dengan jenis kerusakan jalan yang beragam serta ukuran kerusakan jalan yang beragam pula.
2. Menggunakan lensa kamera yang lebih luas (*ultra wide lens*) sehingga kerusakan jalan tidak terbagi dalam beberapa *frame*.
3. Melakukan analisis lebih lanjut terhadap pengaruh sudut pengambilan video.

“Halaman ini sengaja dikosongkan.”

DAFTAR PUSTAKA

[1] Lukman Al Hakim, M. (2015). "Studi Evaluasi Pelaksanaan Kebijakan Pemeliharaan Jalan di Kota Surabaya". Kebijakan dan Manajemen Publik, Volume 3, Nomor 1, Januari-April 2015.

[2] Mahardika, A., dkk. (2018). "Sistem Temu Kembali Citra Lubang Jalan Aspal Berdasarkan Tingkat Kerusakan Menggunakan Ekstrasi Fitur *Gray Level Co-occurrence Matrix*". Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 2, no. 10, pp. 3811 – 3821.

[3] Xiang Xu, dkk. (2019). "*Research on Image Detection and Application of Security Dangerous Goods Based on SSD*". 2019 IEEE 1st *International Conference on Civil Aviation Safety and Information Technology* (ICCASIT).

[4] Yuanyuan Wang, dkk. (2017). "*Combining Single Shot Multibox Detector with Transfer Learning for Ship Detection Using Sentinel*-1 *images*". 2017 SAR *in Big Data Era*: *Models, Methods and Applications* (BIGSARDATA).

[5] Muslim, Muhammad. (2019). "Identifikasi Lubang pada Jalan Menggunakan Pengolahan Citra Digital". Tesis: Institut Teknologi Sepuluh Nopember (ITS).

[6] Pramestya, Ravy Hayu. (2018). "Deteksi dan Klasifikasi Kerusakan Jalan Aspal Menggunakan Metode YOLO Berbasis Citra Digital". Tesis: Institut Teknologi Sepuluh Nopember (ITS).

[7] Sulaksono, W., Sony. (2001). “Rekayasa Jalan”. Bandung: Institut Teknologi Bandung.

[8] Hamdani, D., Nono. (2016). “Penentuan Indeks Kondisi Perkerasan (IKP)”. Kementrian Pekerjaan Umum dan Perumahan Rakyat.

[9] Munir, R. (2004). “Pengolahan Citra Digital dengan Pendekatan Algoritma”. Bandung: Penerbit Informatika.

[10] Goodfellow, Ian., Bengio, Yoshua., Courville, Aaron. (2016). “*Deep Learning*”. London: The MIT Press.

[11] The Matworks, Inc., (2018). “*Introducing Deep Learning with MATLAB*”.

[12] Anita, Juli Nur. (2018). “Pengukuran Dimensi Retakan Pada Citra Jalan Menggunakan Filter *Gabor*”. Tugas Akhir: Institut Teknologi Sepuluh Nopember (ITS).

[13] Oktavian, Syifa Laili Hapsari. (2020). “Penerapan Metode *Faster Regional – Convolutional Neural Network* (*Faster R-CNN*) untuk Klasifikasi Jenis Kerusakan Jalan”. Tugas Akhir: Institut Teknologi Sepuluh Nopember (ITS).

[14] Zufar, Muhammad. (2016). “*Convolutional Neural Networks* untuk Pengenalan Wajah Secara *Real-Time*”. Tugas Akhir: Institut Teknologi Sepuluh Nopember (ITS).

[15] Liu, W., dkk. (2016). “SSD: *Single Shot Multibox Detector*”. 14th *European Conference on Computer VisionECCV*.

[16] Sena, Samuel. (2017). “Pengenalan *Deep Learning Part* 7: *Convolutional Neural Network* (CNN)”. https://medium.com/@samuelsena/pengenalan-deep-

learning-part-7-convolutional-neural-network-cnn-b003b477dc94. (Diakses tanggal 18 Juni 2020).

[17] Torrey, L., Shavlik, J. (2009). "*Transfer Learning*". USA: IGI Global.

[18] Simonyan, K., Zisserman, A. (2015). "*Very Deep Convolutional Networks for Large-Scale Image Recognition*". ICLR 2015.

[19] http://image-net.org/challenges/LSVRC/2014/results#clsloc. (diakses tanggal 18 Juni 2020).

[20] Forson, Eddie. (2017). "*Understanding* SSD *Multibox – Real-Time Object Detection in Deep Learning*". https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab. (Diakses tanggal 26 Juni 2020).

[21] Li, Congcong. (2018). "*High Quality*, *Fast*, *Modular Reference Implementation of* SSD *in* PyTorch". https://github.com/lufficc/SSD.

[22] https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html (diakses tanggal 17 agustus 2020).

[23] https://manalelaidouni.github.io/manalelaidouni.github.io/Single%20shot%20object%20detection.html (diakses tangal 17 agustus 2020)

[24] Nurhikmat, T. (2018). "Implementasi *Deep Learning* untuk *Image Classification* Menggunakan Algoritma *Convolutional Neural Network* (CNN) pada Citra Wayang Golek".

[25] Geron, Aurelien. (2017). "*Hands-on Machine Learning with Scikit-learn and Tensorflow*". *United States of America*.

“Halaman ini sengaja dikosongkan.”

BIODATA PENULIS

Penulis memiliki nama lengkap Robertus Diawan Chris. Saat ini penulis menempuh pendidikan S1 di Departemen Matematika Fakultas Sains dan Analitika Data Institut Teknologi Sepuluh Nopember Surabaya. Demikian biodata tentang penulis. (*Source code*: github.com/bruhtus/pavement_distress_ssd).