

**PENERAPAN *DEEP LEARNING* PADA KLASIFIKASI CITRA SAMPAH  
DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK***

**SKRIPSI**

diajukan untuk menempuh ujian sarjana  
pada Fakultas Matematika dan Ilmu Pengetahuan Alam  
Universitas Padjadjaran

**RHEZA PANDYA ANDHIKAPUTRA**

**NPM 140810200023**



**UNIVERSITAS PADJADJARAN  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
PROGRAM STUDI TEKNIK INFORMATIKA  
SUMEDANG  
2024**

## SKRIPSI

### **PENERAPAN *DEEP LEARNING* PADA KLASIFIKASI CITRA SAMPAH DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK***

### **APPLICATION OF DEEP LEARNING TO GARBAGE CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORK METHOD**

Telah dipersiapkan dan disusun oleh

RHEZA PANDYA ANDHIKAPUTRA

NPM 140810200023

Telah dipertahankan di depan Tim Penguji  
pada tanggal 25 Januari 2024

#### Susunan Tim Penguji

- |  |                   |       |
|--|-------------------|-------|
| 1. <u>Dr. Juli Rejito, M.Kom.</u><br>NIP. 19680717 199303 1 003      | Ketua Tim Penguji | ..... |
| 2. <u>Dr. Intan Nurma Yulita, M.T</u><br>NIP. 19850704 201504 2 003  | Pembimbing        | ..... |
| 3. <u>Drs. Akik Hidayat, M.Kom</u><br>NIP. 19611018 198603 1 002     | Co-Pembimbing     | ..... |
| 4. <u>Dr. Setiawan Hadi, M.Sc.CS.</u><br>NIP. 19620701 199302 1 001  | Penguji           | ..... |
| 5. <u>Deni Setiana, S.Si. M.CS.</u><br>NIP. 19730925 200312 1 003    | Penguji           | ..... |
| 6. <u>Dr. R. Sudrajat, Drs., M.Si.</u><br>NIP. 19600212 198701 1 001 | Penguji           | ..... |

## KATA PENGANTAR

*Bismillahirrahmanirrahim.* Dengan menyebut nama Allah yang Maha Pengasih dan Maha Penyayang. Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul “**PENERAPAN *DEEP LEARNING* PADA KLASIFIKASI CITRA SAMPAH DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK***” sebagai salah satu syarat menempuh sarjana pada Program Studi S-1 Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Dalam proses penyusunan dan penulisan skripsi ini tidak terlepas dari bantuan, bimbingan, serta dukungan dari berbagai pihak. Oleh karena itu dalam kesempatan ini penulis mengucapkan terima kasih kepada Ibu Dr. Intan Nurma Yulita, M.T sebagai pembimbing utama, Bapak Drs. Akik Hidayat, M.Kom, sebagai pembimbing pendamping yang telah meluangkan waktu dan pikirannya sehingga penulis dapat menyelesaikan skripsi ini. Ucapan terima kasih juga diberikan kepada keluarga penulis yang selalu memberikan motivasi dan doa yang menjadi pendorong dalam penyelesaian skripsi ini. Penulis juga mengucapkan terima kasih sebanyak-banyaknya kepada:

1. Prof. Dr. Iman Rahayu, S.Si., M.Si., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.
2. Dr. Setiawan Hadi, M.Sc.CS., selaku Kepala Departemen Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

3. Dr. Juli Rejito, M.Kom., selaku Ketua Program Studi S-1 Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran sekaligus Dosen Wali penulis yang telah membimbing selama masa perkuliahan.
4. Dosen-dosen Teknik Informatika Unpad yang telah mengajar dan memberikan ilmu kepada penulis selama masa perkuliahan yang membawa penulis pada posisi sekarang ini.
5. Teman-teman yang telah membantu dan memberikan dukungan dalam penulisan skripsi ini.

Akhir kata, penulis berharap semoga skripsi ini dapat bermanfaat bagi pembaca.

Jatinangor, 25 Januari 2024

Penulis

## ABSTRAK

Permasalahan sampah menjadi isu global yang memprihatinkan, dengan 2,01 miliar ton limbah padat kota dihasilkan setiap tahun, dan 33 persen tidak dikelola secara ramah lingkungan. Di Indonesia, pertumbuhan penduduk meningkatkan urgensi masalah sampah, terlihat dari 63,53 persen dari 33,27 juta ton sampah yang berhasil dikelola. Rendahnya kesadaran masyarakat dan minimnya informasi tentang pemilahan dan pengelolaan sampah menjadi akar permasalahan. Penelitian ini memiliki tujuan memberikan solusi melalui pengembangan aplikasi untuk membantu mengklasifikasikan sampah secara mudah yang diharapkan dapat berkontribusi dalam mengatasi masalah sampah di Indonesia.

Penelitian ini menggunakan teknologi *Deep Learning* yaitu *Convolutional Neural Network* untuk mengklasifikasikan citra sampah ke dalam kelas *Cardboard*, *Glass*, *Metal*, *Paper*, *Plastic* dan *Trash*. Penelitian ini difokuskan pada perbandingan *hyperparameter* dengan beberapa skenario yang digunakan pada model *Deep Learning* dengan membandingkan penggunaan *Pretrained Model*, *Optimizer*, dan *Dropout Layer* yang berbeda untuk menemukan konfigurasi terbaik yang menghasilkan *F1 Score* terbaik.

Dari hasil penelitian didapat model *Convolutional Neural* dengan nilai *F1 Score* sebesar 0,9511. Dari hasil tersebut dapat disimpulkan bahwa model menghasilkan performa yang baik untuk melakukan klasifikasi citra sampah. Aplikasi ini dibuat dalam bentuk aplikasi berbasis *website* sederhana untuk melakukan klasifikasi citra sampah.

**Kata Kunci** : Sampah, *Deep Learning*, *Convolutional Neural Network*, *Hyperparameter*

## **ABSTRACT**

*The global issue of waste has become a concerning problem, with 2.01 billion tons of municipal solid waste generated annually, of which 33 percent is not environmentally managed. In Indonesia, population growth intensifies the urgency of waste management, evident in the successful management of only 63.53 percent of the 33.27 million tons of generated waste. Low public awareness and insufficient information on waste sorting and management are root challenges. This research aims to provide a solution through the development of a application for easy waste classification, contributing to waste management in Indonesia.*

*The research utilizes Deep Learning technology, specifically Convolutional Neural Network, to classify waste images into categories such as Cardboard, Glass, Metal, Paper, Plastic, and Trash. The focus lies in comparing hyperparameters under different scenarios, including the use of Pretrained Models, Optimizers, and varying Dropout Layer configurations, to identify the optimal setup yielding the highest F1 Score.*

*The research results reveal a Convolutional Neural Network model with F1 Score of 0.9511, indicating the model's proficient performance in image classification. The application is implemented as a simple web-based application for waste image classification.*

**Keywords** : Waste, Deep Learning, Convolutional Neural Network, Hyperparameter

## DAFTAR ISI

ABSTRAK .....	v
<i>ABSTRACT</i> .....	vi
DAFTAR ISI .....	vii
DAFTAR TABEL .....	x
DAFTAR GAMBAR .....	xi
DAFTAR LAMPIRAN .....	xiii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang.....	1
1.2 Identifikasi Masalah.....	2
1.3 Batasan Masalah .....	3
1.4 Maksud dan Tujuan Penelitian .....	3
1.5 Manfaat Penelitian .....	4
1.6 Metodologi Penelitian.....	4
1.7 Sistematika Penulisan .....	5
BAB II TINJAUAN PUSTAKA.....	7
2.1 Pengertian Sampah .....	7
2.2 <i>Deep Learning</i> .....	7
2.3 <i>Convolutional Neural Network</i> .....	9
2.3.1 <i>Convolution Layer</i> .....	10
2.3.2 <i>Pooling Layer</i> .....	12
2.3.3 <i>Fully Connected Layer</i> .....	13
2.3.4 <i>Dropout Layer</i> .....	13
2.3.5 <i>Optimizer</i> .....	15
2.3.6 <i>Learning Rate</i> .....	17
2.4 <i>Data Augmentation</i> .....	17
2.5 <i>K-Fold Cross Validation</i> .....	18
2.6 <i>Residual Network (ResNet)</i> .....	19
2.7 <i>Visual Geometry Group (VGG)</i> .....	21

2.8	<i>Densely Connected Convolutional Networks (DenseNet)</i> .....	22
2.9	Python.....	23
2.10	TensorFlow .....	23
2.11	OpenCV .....	24
2.12	<i>Confusion Matrix</i> .....	25
2.13	<i>Unified Modelling Language (UML)</i> .....	27
2.13.1	<i>Use Case Diagram</i> .....	27
2.13.2	<i>Activity Diagram</i> .....	28
2.13.3	<i>Sequence Diagram</i> .....	29
2.13.4	<i>Deployment Diagram</i> .....	30
2.14	Penelitian Terdahulu .....	31
<b>BAB III ANALISIS DAN PERANCANGAN</b> .....		33
3.1	Fase Analisis.....	33
3.1.1	Kebutuhan Perangkat Lunak.....	33
3.1.2	Kebutuhan Perangkat Keras.....	34
3.2	Fase Penelitian .....	34
3.2.1	Pengumpulan Data .....	34
3.2.2	<i>Data Preprocessing</i> .....	35
3.2.3	<i>Data Splitting</i> .....	37
3.2.4	Perancangan Model.....	38
3.2.5	Pelatihan Model .....	43
3.2.6	Evaluasi Model .....	43
3.2.7	Konversi Model .....	43
3.3	Fase Perancangan Aplikasi .....	44
3.3.1	<i>Use Case Diagram</i> .....	44
3.3.2	<i>Activity Diagram</i> .....	45
3.3.3	<i>Sequence Diagram</i> .....	46
3.3.4	<i>Deployment Diagram</i> .....	47
3.3.5	<i>Wireframe</i> Aplikasi.....	48
<b>BAB IV HASIL DAN PEMBAHASAN</b> .....		51
4.1	Implementasi Pembuatan Model .....	51



4.1.1	<i>Import Library</i> .....	51
4.1.2	<i>Data Preprocessing</i> .....	52
4.1.3	<i>Data Splitting</i> .....	55
4.1.4	Pembuatan Model .....	55
4.1.5	Pelatihan Model .....	57
4.1.6	Evaluasi Model .....	59
4.1.7	Konversi Model .....	60
4.2	Analisis dan Perhitungan Performa .....	61
4.3	Implementasi Pembuatan Aplikasi .....	72
4.3.1	<i>Landing Page</i> .....	73
4.3.2	<i>Classification Page</i> .....	73
4.3.3	<i>Prediction Page</i> .....	77
4.3.4	<i>History Page</i> .....	77
4.3.5	Flask Application .....	78
BAB V KESIMPULAN DAN SARAN .....		81
5.1	Kesimpulan .....	81
5.2	Saran .....	82
DAFTAR PUSTAKA .....		84
LAMPIRAN .....		86
RIWAYAT HIDUP .....		104

## DAFTAR TABEL

Tabel 2.1 Simbol-simbol pada <i>Use Case Diagram</i> .....	28
Tabel 2.2 Simbol-simbol pada <i>Sequence Diagram</i> .....	29
Tabel 2.3 Simbol-simbol pada <i>Sequence Diagram</i> .....	29
Tabel 2.4 Simbol-simbol pada <i>Deployment Diagram</i> .....	31
Tabel 2.5 Penelitian Terdahulu .....	32
Tabel 3.1 Skenario <i>Data Preprocessing</i> .....	36
Tabel 3.2 Skenario Perbandingan <i>Hyperparameter</i> .....	38
Tabel 4.1 <i>Precision, Recall, dan F1 Score</i> Hasil Skenario .....	61
Tabel 4.2 Sampel Kesalahan Klasifikasi.....	63
Tabel 4.3 Perhitungan Nilai <i>Precision</i> .....	66
Tabel 4.4 Perhitungan Nilai <i>Recall</i> .....	68
Tabel 4.5 Perhitungan Nilai <i>F1 Score</i> .....	69

## DAFTAR GAMBAR

Gambar 2.1 Ilustrasi <i>Deep Learning</i> (Goodfellow et al., 2016) .....	8
Gambar 2.2 Ilustrasi Arsitektur <i>Convolutional Neural Network</i> (Zhang et al., 2023) .....	9
Gambar 2.3 Ilustrasi <i>Convolution Layer</i> (Goodfellow et al., 2016) .....	11
Gambar 2.4 Ilustrasi <i>Max Pooling Layer</i> (Goodfellow et al., 2016) .....	12
Gambar 2.5 Ilustrasi <i>Dropout Layer</i> (Goodfellow et al., 2016).....	14
Gambar 2.6 Ilustrasi <i>5-Fold Cross Validation</i> (Ozdemir, 2016) .....	19
Gambar 2.7 Perbandingan <i>error</i> pada <i>Convolutional Neural Network 20 layers</i> dan <i>56 layers</i> (He et al., 2016) .....	20
Gambar 2.8 Ilustrasi <i>Residual Block</i> (He et al., 2016).....	21
Gambar 2.9 Ilustrasi DenseNet (Huang et al., 2017) .....	22
Gambar 2.10 Ilustrasi <i>Confusion Matrix</i> (Sammur & Webb, 2017).....	25
Gambar 3.1 Sampel Citra Sampah .....	35
Gambar 3.2 Skenario Arsitektur Model ResNet50 .....	40
Gambar 3.3 Skenario Arsitektur Model VGG19 .....	41
Gambar 3.4 Skenario Arsitektur Model DenseNet121 .....	42
Gambar 3.5 <i>Use Case Diagram</i> .....	44
Gambar 3.6 <i>Activity Diagram</i> .....	45
Gambar 3.7 <i>Sequence Diagram</i> .....	46
Gambar 3.8 <i>Deployment Diagram</i> .....	47
Gambar 3.9 <i>Wireframe Landing Page</i> .....	48
Gambar 3.10 <i>Wireframe Classification Page</i> .....	49
Gambar 3.11 <i>Wireframe Prediction Page</i> .....	49
Gambar 3.12 <i>Wireframe History Page</i> .....	50
Gambar 4.1 Visualisasi Sampel Data.....	54
Gambar 4.2 <i>Confusion Matrix</i> .....	62
Gambar 4.3 Tampilan <i>Landing Page</i> .....	73
Gambar 4.4 Tampilan <i>Classification Page</i> .....	74

Gambar 4.5 Tampilan <i>Prediction Page</i> .....	77
Gambar 4.6 Tampilan <i>History Page</i> .....	78

## DAFTAR LAMPIRAN

Lampiran 1 Kode Implementasi Pembuatan Model.....	86
Lampiran 2 Kode Implementasi Pembuatan Aplikasi <i>Website</i> .....	92

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Masalah sampah merupakan isu global yang mempengaruhi kehidupan di seluruh negara di dunia. Jumlah limbah padat kota yang dihasilkan secara global mencapai 2,01 miliar ton per tahun. Sekitar 33 persen dari jumlah tersebut tidak dikelola dengan cara yang ramah lingkungan. Rata-rata produksi sampah per orang per hari sekitar 0,74 kilogram. Namun, angka ini bervariasi dari 0,11 hingga 4,54 kilogram (Kaza et al., 2018).

Masalah sampah di Indonesia semakin mendesak karena pertumbuhan penduduk menyebabkan peningkatan jumlah dan jenis sampah setiap tahunnya. Jumlah timbulan sampah di Indonesia mencapai 33,27 juta ton per tahun. Namun, baru 63,53 persen atau sekitar 21,1 juta ton sampah yang berhasil dikelola dengan 38,22 persen diantaranya merupakan sampah rumah tangga (Direktorat Penanganan Sampah, 2022). Situasi ini menandakan bahwa Indonesia sedang mengalami keadaan darurat sampah.

Salah satu solusi untuk mengatasi masalah sampah yang semakin bertambah adalah dengan memilah dan mengelola sampah dengan lebih baik. Hal ini dapat membantu mengatasi masalah sampah di sekitar tempat tinggal dan lahan Tempat Pembuangan Akhir (TPA) yang semakin terbatas dan mencemari lingkungan. Tingkat kesadaran masyarakat yang rendah dan kurangnya informasi tentang pengelolaan sampah menjadi akar permasalahan. Hal ini merupakan faktor

penyebab banyak sampah yang tidak terpilah dengan baik sehingga menimbulkan masalah sampah.

Dalam upaya mengatasi masalah sampah, teknologi *Artificial Intelligence*, khususnya *Deep Learning* dapat menjadi solusi yang efektif. Salah satu implementasi dari teknologi tersebut adalah *image classification*. Teknologi ini telah diterapkan dalam berbagai aspek kehidupan. Teknologi ini dibuat dengan merekayasa sistem yang terinspirasi dari cara kerja otak manusia yang disebut jaringan saraf atau *neural network* (Goodfellow et al., 2016).

*Convolutional Neural Network* adalah salah satu metode *neural network* yang telah banyak digunakan dalam kebutuhan *image classification* dalam berbagai bidang. Dalam penelitian ini, penulis menerapkan model *Convolutional Neural Network* dengan tujuan untuk mengklasifikasi citra sampah dan mengelompokkan menjadi beberapa kelas berdasarkan bahan dasar sampah. Penelitian ini berfokus pada perbandingan beberapa *hyperparameter* pada model *Convolutional Neural Network* yang dibuat dalam beberapa skenario pelatihan model untuk mencari konfigurasi *hyperparameter* model yang menghasilkan *F1 Score* yang paling optimal. Pembuatan program dan evaluasi model akan dibuat menggunakan bahasa pemrograman Python diikuti dengan implementasi dalam aplikasi berbasis *website*.

## 1.2 Identifikasi Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, masalah yang akan dicari solusinya dalam penelitian ini adalah sebagai berikut:

1. Bagaimana cara menerapkan model *Convolutional Neural Network* untuk klasifikasi citra sampah?

2. Apa konfigurasi *hyperparameter* pada model *Convolutional Neural Network* yang terbaik sehingga diperoleh hasil *F1 Score* yang optimal?
3. Bagaimana hasil performa model *Convolutional Neural Network* yang terbaik berdasarkan *F1 Score* yang paling optimal?
4. Bagaimana cara menerapkan model *Convolutional Neural Network* pada aplikasi berbasis *website*?

### 1.3 Batasan Masalah

Dari identifikasi masalah tersebut, maka dalam penelitian ini terdapat beberapa batasan masalah pada sebagai berikut:

1. Metode yang digunakan adalah *Convolutional Neural Network* menggunakan *framework* TensorFlow dengan bahasa pemrograman Python.
2. *Hyperparameter* yang dibandingkan dalam skenario pelatihan yaitu penggunaan *Pretrained Model*, *Optimizer*, dan *Dropout Layer*.
3. Data citra yang digunakan merupakan *dataset* Garbage Classification yang diperoleh dari *platform* Kaggle dan telah memiliki *label* sebanyak 6 kelas, yaitu *Cardboard*, *Glass*, *Metal*, *Paper*, *Plastic*, dan *Trash*.
4. Tahap pembuatan model dan pelatihan data dilakukan dengan menggunakan *platform* Google Colab dengan menggunakan *library* yang dibutuhkan.

### 1.4 Maksud dan Tujuan Penelitian

Maksud dari penelitian ini adalah menerapkan model *Convolutional Neural Network* untuk klasifikasi citra sampah dengan nilai *F1 Score* yang optimal serta diterapkan pada aplikasi berbasis *website*.



Tujuan yang ingin dicapai oleh penulis dari penelitian ini adalah :

1. Dapat menerapkan metode *Convolutional Neural Network* untuk klasifikasi citra sampah.
2. Dapat menemukan konfigurasi *hyperparameter* pada model *Convolutional Neural Network* dengan mendapatkan nilai *F1 Score* yang paling optimal untuk klasifikasi citra sampah.
3. Dapat menghasilkan sebuah aplikasi klasifikasi citra sampah berbasis *website*.

### **1.5 Manfaat Penelitian**

Manfaat yang diharapkan dari penelitian ini adalah:

1. Dapat memudahkan pengguna yang ingin mengklasifikasikan sampah.
2. Dapat menambah pengetahuan mengenai implementasi metode *Convolutional Neural Network* pada klasifikasi citra sampah.
3. Dapat mengembangkan penelitian yang sebelumnya telah dibuat.

### **1.6 Metodologi Penelitian**

Jenis penelitian yang akan digunakan ialah *Research and Development*, yaitu penelitian dengan menerapkan langkah-langkah yang ada untuk menghasilkan sebuah aplikasi klasifikasi citra sampah. Tahapan yang akan dilalui adalah sebagai berikut:

1. Studi literatur. Pada tahap ini penulis mencari referensi dan informasi dari buku, jurnal, maupun artikel yang mendukung penelitian.
2. Pengumpulan data. Pada tahap ini penulis mengumpulkan *dataset* citra sampah menggunakan *platform* Kaggle.

3. Perancangan Model. Pada tahap ini penulis melakukan perancangan model *Convolutional Neural Network* untuk klasifikasi citra sampah.
4. Implementasi dan Evaluasi Model. Pada tahap ini penulis melakukan implementasi hasil rancangan menjadi sebuah model *Convolutional Neural Network* untuk klasifikasi citra sampah menggunakan bahasa pemrograman Python dan mengevaluasi model tersebut serta diterapkan dalam bentuk aplikasi berbasis *website*.
5. Penulisan laporan. Pada tahap ini penulis menuliskan hasil penelitian berupa pengembangan aplikasi klasifikasi citra sampah menggunakan metode *Convolutional Neural Network* ke dalam laporan skripsi.

## **1.7 Sistematika Penulisan**

Untuk memberi gambaran yang jelas tentang penelitian ini, maka disusunlah sistematika penulisan yang berisi materi yang akan dibahas pada setiap bab. Sistematika dalam penulisan tugas akhir ini adalah sebagai berikut:

### **BAB I PENDAHULUAN**

Pada bab ini dijelaskan tentang latar belakang dari topik penulisan skripsi, pokok permasalahan berupa identifikasi dan batasan masalah, tujuan dan manfaat yang diharapkan dari penulisan skripsi, metodologi yang digunakan serta sistematika penulisan.

### **BAB II TINJAUAN PUSTAKA**

Pada bab ini dijelaskan seluruh landasan teori yang berhubungan dengan penelitian, yaitu tentang metode *Convolutional Neural Network*, penjelasan teoritis mengenai bahasa pemrograman dan *framework* serta *library* yang digunakan dalam proses

pengimplementasian aplikasi, serta teori lainnya guna memahami permasalahan yang dibahas.

### **BAB III ANALISIS DAN PERANCANGAN**

Pada bab ini dijelaskan tentang metode *Deep Learning* yang digunakan meliputi analisis kebutuhan aplikasi, perancangan aplikasi, dan diagram pemodelan aplikasi.

### **BAB IV HASIL DAN PEMBAHASAN**

Pada bab ini dijelaskan tentang implementasi model *Convolutional Neural Network* untuk klasifikasi citra sampah yang telah dirancang, evaluasi model, serta hasil dari penerapan aplikasi.

### **BAB V KESIMPULAN DAN SARAN**

Bab ini merupakan penutup yang berisi kesimpulan dan saran dari penelitian yang sudah dilakukan.

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1 Pengertian Sampah**

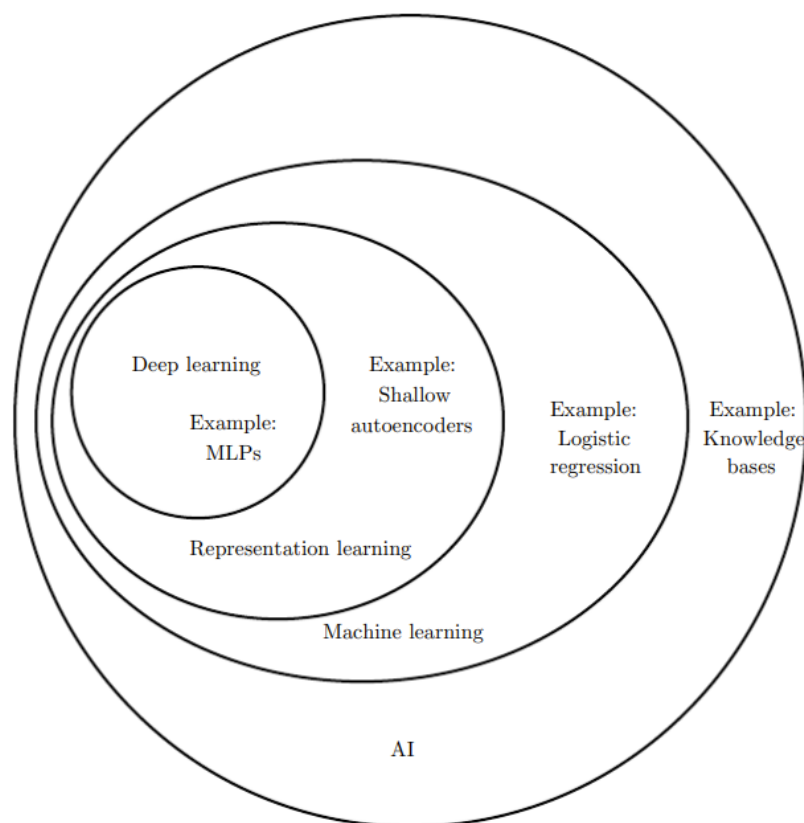
Sampah adalah sisa kegiatan sehari-hari manusia dan/atau proses alam yang berbentuk padat (Pemerintah Republik Indonesia, 2008). Istilah sampah umumnya digunakan untuk merujuk pada limbah padat. Sampah terdiri dari berbagai sisa bahan yang telah melalui berbagai proses, seperti bagian utamanya sudah diambil, melalui pengolahan, atau karena tidak memiliki manfaat lagi dari segi sosial ekonomi dan berpotensi menyebabkan pencemaran serta gangguan bagi lingkungan hidup.

Sampah dapat diklasifikasikan menjadi dua jenis yaitu sampah mudah membusuk atau sampah organik, dan sampah yang sulit membusuk atau sampah anorganik. Sampah juga dapat diklasifikasikan berdasarkan bahan dasarnya, seperti kardus, plastik, besi, kaca, dan lain-lain. Dengan klasifikasi tersebut, pengelolaan dan pemilahan sampah akan lebih mudah dilakukan.

#### **2.2 *Deep Learning***

*Deep Learning* adalah teknik pembelajaran dari kecerdasan buatan atau *Artificial Intelligence* dalam bidang *machine learning*. Teknik ini melibatkan komposisi yang lebih banyak dari fungsi atau konsep yang dipelajari dibandingkan *machine learning* tradisional. *Deep Learning* memungkinkan komputer untuk mempelajari program komputer dengan banyak instruksi yang dilakukan dengan menggunakan lapisan representasi sebagai memori yang akan mengeksekusi

instruksi secara berurutan. Instruksi secara berurutan menjadi sangat kuat karena instruksi selanjutnya dapat merujuk kembali ke hasil instruksi yang sebelumnya telah dilakukan. Lapisan representasi juga menyimpan informasi yang membantu menjalankan program yang dapat memahami *input* (Goodfellow et al., 2016).



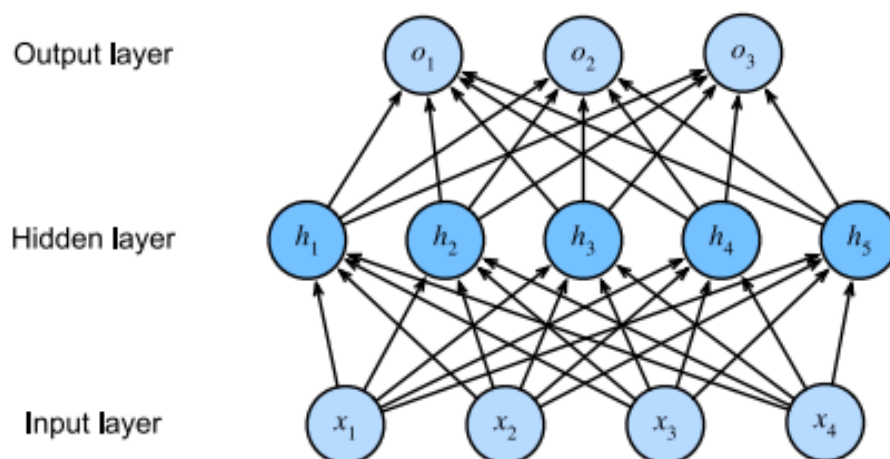
Gambar 2.1 Ilustrasi *Deep Learning* (Goodfellow et al., 2016)

Pada Gambar 2.1, *Deep Learning* dapat diilustrasikan sebagai bagian terdalam dari ilmu *Artificial Intelligence*. *Deep Learning* telah banyak diimplementasikan dalam berbagai macam bidang dengan skala data yang besar, seperti dalam bidang kesehatan, keuangan, dan bidang lainnya. *Framework* populer yang digunakan dalam pengembangan *Deep Learning* seperti TensorFlow dan PyTorch.

### 2.3 Convolutional Neural Network

*Convolutional Neural Network* atau disingkat CNN adalah jenis jaringan saraf khusus untuk memproses data yang memiliki topologi yang seperti bingkai. Nama *Convolutional Neural Network* menunjukkan bahwa jaringan menggunakan operasi matematika yang disebut *convolution*. *Convolution* adalah jenis operasi linier khusus. Jaringan *convolution* merupakan jaringan saraf yang menggunakan konvolusi sebagai pengganti matriks umum perkalian dalam setidaknya satu lapisannya (Goodfellow et al., 2016).

*Convolutional Neural Network* merupakan sebuah arsitektur model yang sangat populer dan efektif dalam dalam pengaplikasian pada bidang *Deep Learning*, namun memerlukan jumlah data yang besar dalam proses pelatihannya. Model ini telah banyak diaplikasikan dalam berbagai bidang pada *Deep Learning*, seperti *computer vision*, *speech recognition*, *natural language processing*, dan masih banyak pengaplikasian lainnya.



Gambar 2.2 Ilustrasi Arsitektur *Convolutional Neural Network* (Zhang et al., 2023)

Gambar 2.2 merupakan ilustrasi arsitektur *Convolutional Neural Network* dapat dibagi menjadi *input layer*, *hidden layer* dengan *layer* yang dikonfigurasi sesuai dengan kebutuhan, dan *output layer* yang disesuaikan dengan bentuk *output* yang diharapkan. Dalam konteks ini, *neural network* yang digunakan pada *hidden layer* dapat menghasilkan *output* yang diinginkan berdasarkan *input* yang diberikan oleh pengguna.

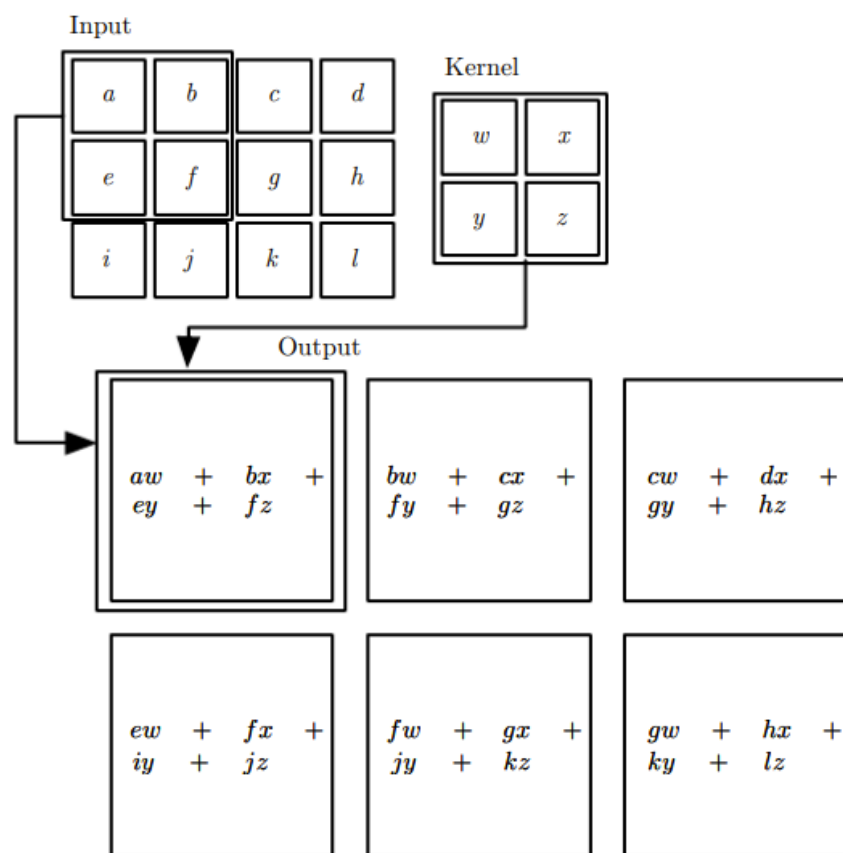
*Convolutional Neural Network* telah memainkan peran penting dalam sejarah *Machine Learning*. Mereka adalah contoh utama dari aplikasi yang berhasil menggunakan wawasan yang diperoleh dari mempelajari otak makhluk hidup untuk *Machine Learning*. *Convolutional Neural Network* juga merupakan beberapa jaringan saraf pertama yang berhasil memecahkan aplikasi komersial penting dan masih digunakan dalam *Deep Learning* yang komersial saat ini.

### 2.3.1 *Convolution Layer*

*Convolution Layer* atau lapisan konvolusi adalah elemen kunci dalam arsitektur *Convolutional Neural Networks*. *Convolution Layer* pada dasarnya adalah operasi pada dua fungsi dengan argumen bernilai ril. Dalam *Convolution Layer*, argumen pertama disebut *input* yang biasanya berupa *array* multidimensi dan argumen kedua disebut *kernel* yang biasanya berupa *array* multidimensi dari parameter yang disesuaikan oleh algoritma, sedangkan *outputnya* adalah fitur-fitur dari data input yang disebut *feature map* (Goodfellow et al., 2016).

*Convolution* diskrit dapat dianggap sebagai perkalian dengan matriks. Matriks tersebut memiliki beberapa *input* yang dibatasi untuk sama dengan *input* lainnya. Setiap algoritma *neural network* yang bekerja dengan perkalian matriks

tidak tergantung pada struktur matriks yang harus bekerja dengan *Convolution Layer*. *Convolution Layer* memiliki parameter bernama *stride*, yaitu jarak antara pergeseran *kernel*. *Stride* dapat digunakan untuk proses *downsampling feature map*, yaitu mengurangi dimensi dari *feature map* dengan tujuan memperkecil ukuran data yang diolah.



Gambar 2.3 Ilustrasi *Convolution Layer* (Goodfellow et al., 2016)

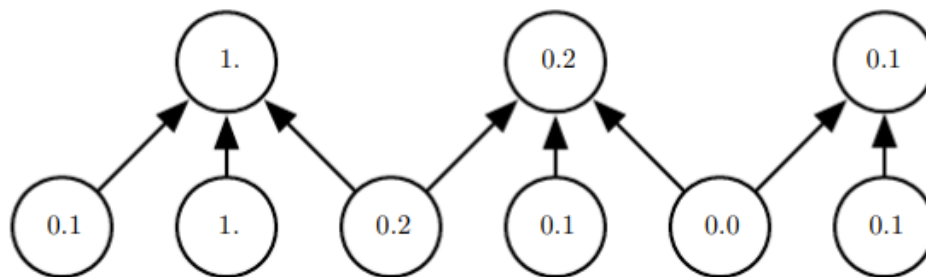
Pada Gambar 2.3, *Convolution Layer* dapat diilustrasikan dengan kotak dengan panah untuk menunjukkan bagaimana elemen pertama dari *tensor output* dibentuk dengan menerapkan *kernel* ke *tensor input*. *Stride* atau pergeseran sering digunakan bersama dengan *padding*, yaitu penambahan dimensi pada data *input*



apabila *stride* melebihi ukuran dimensi *input*. *Padding* berguna untuk memastikan bahwa informasi pada data *input* tidak hilang selama proses konvolusi.

### 2.3.2 Pooling Layer

*Pooling Layer* adalah teknik yang digunakan dalam *neural network* untuk membuat representasi data menjadi lebih *invariant* terhadap perubahan kecil dalam *input* dengan menambahkan prioritas untuk memastikan bahwa fungsi yang dipelajari lebih *invariant* terhadap perubahan kecil *input* yang dapat meningkatkan efisiensi jaringan secara statistik. *Pooling Layer* juga dapat mempercepat proses komputasi, karena lapisan berikutnya memiliki jumlah *input* yang lebih sedikit untuk diproses. *Pooling Layer* juga penting untuk mengatasi variasi ukuran *input* dalam banyak tugas, seperti klasifikasi gambar dengan ukuran yang berbeda-beda (Goodfellow et al., 2016). Salah satu jenis *Pooling Layer* yang paling populer adalah *Max Pooling* seperti pada Gambar 2.4.



Gambar 2.4 Ilustrasi *Max Pooling Layer* (Goodfellow et al., 2016)

Pada Gambar 2.4, *Max Pooling Layer* bekerja dengan memilih nilai maksimum dari setiap nilai dalam *feature map* untuk menghasilkan hasil *downsampling*. Hal ini mengurangi ukuran representasi yang mengurangi beban komputasi pada lapisan berikutnya. Semua nilai yang dimasukkan untuk *Max*

*Pooling* harus disertakan jika tidak ingin mengabaikan beberapa *unit* detektor. Selain itu, terdapat juga metode *Pooling* lain yang dikenal sebagai *Global Average Pooling* yaitu dengan melakukan *downsampling ekstrim* dengan mengubah seluruh *feature map* menjadi berukuran 1x1 dengan cara mengambil nilai rata-rata dari semua elemen dalam *feature map*.

### 2.3.3 Fully Connected Layer

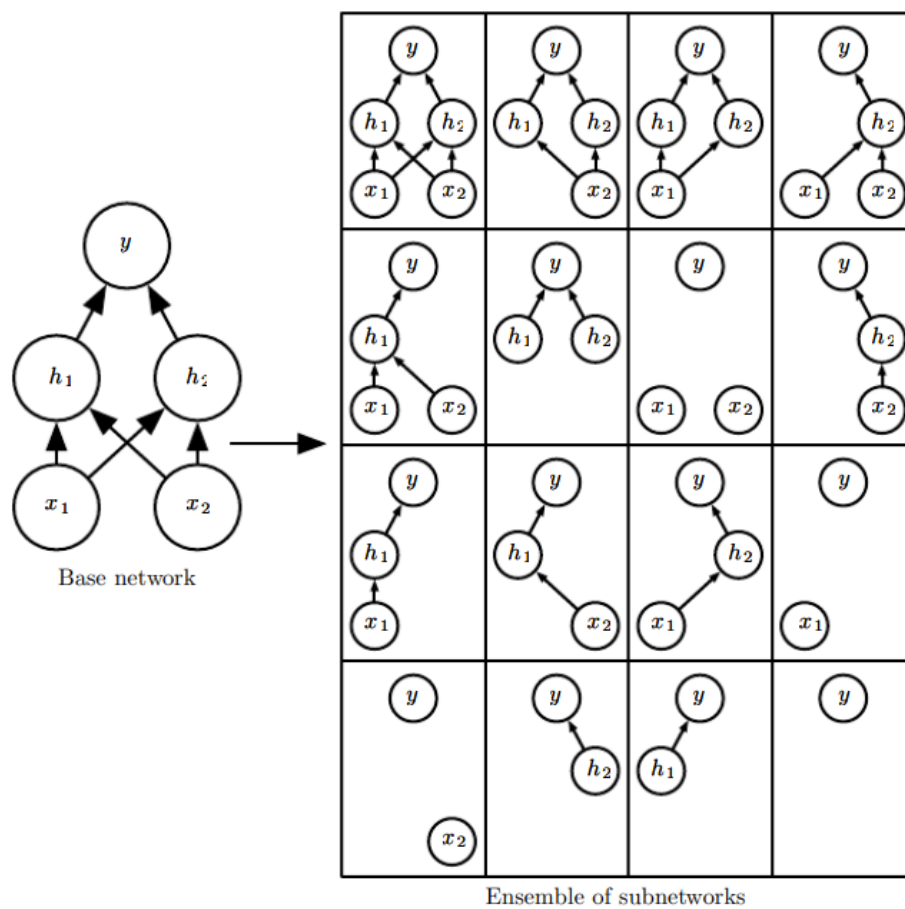
*Fully Connected Layer* atau *Dense Layer* adalah teknik untuk melakukan klasifikasi pada *output* yang sudah menjadi vektor. Setiap jaringan pada *layer* ini saling terhubung dengan jaringan pada *layer* sebelumnya. *Fully Connected Layer* dapat digunakan sebagai *output layer* dan digunakan sebagai *hidden units* (Goodfellow et al., 2016).

*Fully Connected Layer* yang digunakan sebagai *output layer* untuk klasifikasi menjadi beberapa kelas atau *multiclass classifier* menggunakan fungsi aktivasi khusus untuk lapisan terakhir seperti *softmax*, dengan jumlah *units* sebanyak jumlah kelas dalam *label* untuk menghasilkan nilai probabilitas diantara 0 hingga 1 untuk setiap kelasnya.

### 2.3.4 Dropout Layer

*Dropout Layer* merupakan teknik untuk mengurangi kesalahan generalisasi dengan menggabungkan beberapa model pada *set training* agar model tidak akan membuat kesalahan yang sama pada *set testing*. *Dropout* menyediakan metode dengan *cost* yang murah dari sisi komputasi, tetapi kuat dalam meregularisasi model yang luas. *Dropout* memberikan *cost* yang lebih murah untuk melatih dan

mengevaluasi sebuah kumpulan dari *neural network* secara eksponensial. *Dropout* melatih *ensemble* yang terdiri dari semua sub-jaringan yang dapat dibangun dengan menghapus *unit non-output* dari jaringan dasar yang mendasarinya. *Dropout* dapat secara efektif menghapus *unit* dari sebuah jaringan dengan mengalikan nilai *output* *unit* yang dipilih dengan nol. (Goodfellow et al., 2016)



Gambar 2.5 Ilustrasi *Dropout Layer* (Goodfellow et al., 2016)

Pada Gambar 2.5, sebagian besar jaringan yang dihasilkan tidak memiliki *unit input* atau tidak ada koneksi yang menghubungkan *input* ke *output*. Masalah ini menjadi tidak signifikan untuk arsitektur *neural network* yang lebih luas karena terdapat lebih banyak jalur yang mungkin dari *input* ke *output* sehingga probabilitas

untuk menghapus semua jalur yang mungkin dari *input* ke *output* menjadi lebih kecil.

*Dropout* dapat digunakan dengan memberikan variabel probabilitas untuk mengaktifkan atau menonaktifkan setiap *node* dalam suatu *layer*. *Dropout* bekerja dengan mengambil nilai sampel dari setiap *node*. Jika nilai sampel suatu *node* lebih besar dari satu dikurangi nilai probabilitas, maka *node* tersebut diaktifkan. Jika nilai sampel lebih kecil atau sama dengan satu dikurangi nilai probabilitas, maka *node* tersebut dinonaktifkan. Langkah ini mengatur kehadiran dari setiap *node* selama pelatihan model yang membantu mencegah *overfitting* dan meningkatkan generalisasi pada model (Zhang et al., 2023).

### 2.3.5 Optimizer

*Optimizer* atau *Optimization Algorithm* merupakan salah satu *hyperparameter* dalam model *Deep Learning* untuk mencari parameter terbaik untuk meminimalkan *loss function* pada tahap pelatihan. *Optimizer* yang paling populer untuk *Deep Learning* didasarkan pada pendekatan yang disebut *Gradient Descent*. Metode ini memeriksa pada setiap langkah pelatihan untuk melihat arah *loss* pada set pelatihan akan bergerak lalu memperbarui parameter pada model tersebut ke arah yang akan menurunkan *loss* (Zhang et al., 2023). Pada penelitian ini, *Optimizer* yang digunakan yaitu Adam, Nadam, dan AdamW.

## 5. Adam

*Adaptive Moment Estimation* atau disingkat Adam merupakan *Optimization Algorithm* yang sangat populer digunakan dalam *Deep Learning*. Adam

memanfaatkan perkiraan adaptif dari momen orde rendah untuk tujuan stokastik. Metode ini menghitung *learning rate* secara adaptif untuk setiap parameter berdasarkan perkiraan momen pertama dan kedua dari gradien. Metode ini memiliki keunggulan yaitu mudah diimplementasikan, efisien secara komputasional, memiliki kebutuhan memori yang sedikit, dan tidak dipengaruhi oleh penskalaan diagonal dari gradien (Kingma & Ba, 2014).

## 6. Nadam

*Nesterov-accelerated Adaptive Moment Estimation* atau disingkat Nadam merupakan modifikasi dari Adam dengan mengganti komponen momentum menggunakan ide dari algoritma *Nesterov's Accelerated Gradient* (NAG). Metode ini bertujuan untuk meningkatkan kualitas model dengan menyajikan algoritma pembelajaran yang lebih kuat. Metode ini menunjukkan penggantian komponen momentum dengan mengadopsi NAG yang dapat meningkatkan kecepatan konvergensi dengan mencegah *overshoot* yang sering terjadi pada *accelerated gradient* biasa (Dozat, 2016).

## 7. AdamW

*Adaptive Moment Estimation with Weight Decay* atau disingkat AdamW merupakan modifikasi dari Adam dengan mengintegrasikan *Weight Decay* atau dapat disebut juga dengan *L2 Regularization* untuk meningkatkan efektivitas regularisasi pada pelatihan model. Metode AdamW melibatkan *Weight Decay* pada pembaruan gradien dengan tujuan mencegah *overfitting* pada saat pelatihan model sehingga dapat meningkatkan generalisasi pada

model untuk data-data baru terutama dalam tugas *image recognition* (Loshchilov & Hutter, 2019).

### 2.3.6 *Learning Rate*

*Learning Rate* adalah salah satu parameter penting dalam model *Deep Learning*. *Learning Rate* merupakan sebuah parameter untuk mengukur seberapa cepat sebuah model *Deep Learning* dalam beradaptasi dengan perubahan data. Penggunaan *Learning Rate* yang tinggi dapat membuat model lebih cepat beradaptasi dengan data baru, tetapi cenderung dengan cepat melupakan data lama sedangkan dengan penggunaan *Learning Rate* yang lebih rendah dapat membuat model lebih lambat dalam beradaptasi, tetapi akan membuat model kurang sensitif pada data baru atau rangkaian *data points* yang tidak representatif atau *outliers* (Géron, 2019).

*Learning Rate* dapat dipilih dengan *trial* dan *error* dengan memantau performa pembelajaran dari model. *Learning Rate* awal yang terlalu rendah dapat membuat pembelajaran menjadi terhambat karena *cost* komputasi yang tinggi. Oleh karena itu, cara untuk memilih *Learning Rate* yaitu dapat dilakukan dengan memantau beberapa *epoch* pertama dengan *Learning Rate* yang lebih tinggi, tetapi tetap tidak terlalu tinggi karena dapat menyebabkan model mengalami ketidakstabilan yang parah (Goodfellow et al., 2016).

## 2.4 *Data Augmentation*

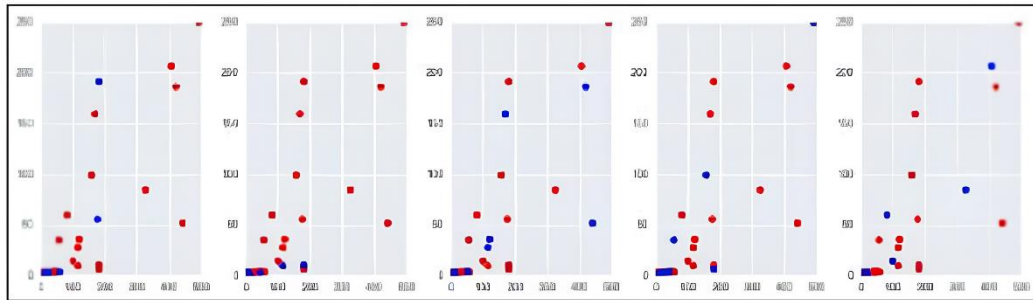
*Data Augmentation* adalah teknik yang efektif dalam *image classification* untuk meningkatkan kinerja model dengan menerapkan transformasi pada data citra

yang memiliki dimensi tinggi dan bervariasi sehingga dapat meningkatkan generalisasi dalam model (Goodfellow et al., 2016). *Data Augmentation* menghasilkan data untuk *set training* yang serupa tetapi berbeda setelah serangkaian transformasi pada citra dalam *set training*, sehingga memperluas ukuran dari *set training* yang memungkinkan model untuk tidak terlalu bergantung pada atribut tertentu (Zhang et al., 2023).

Metode transformasi yang diterapkan pada penelitian ini yaitu *flipping*, *rotating*, *shearing*, *zooming*, dan *shifting*. *Flipping* merupakan pembalikan citra yang dapat dilakukan secara horizontal dan vertikal. *Rotating* merupakan pemutaran citra maksimal sebanyak derajat pada *parameter* yang diberikan. *Shearing* merupakan perubahan kemiringan dan perspektif maksimal sebanyak persentase pada *parameter* yang diberikan. *Zooming* merupakan perbesaran (*zoom in*) atau pengecilan (*zoom out*) citra maksimal sebanyak persentase pada *parameter* yang diberikan. *Shifting* merupakan pergeseran citra secara horizontal dan vertikal maksimal sebanyak persentase pada *parameter* yang diberikan.

## 2.5 *K-Fold Cross Validation*

*K-Fold Cross Validation* merupakan salah satu teknik untuk *splitting dataset* dengan mengambil sejumlah irisan data dari *dataset* sebanyak K dan dilakukan sebanyak K kali atau *K fold*. Setiap *fold* memperlakukan jumlah irisan sebanyak K dikurang satu sebagai *set training* dan satu irisan sisa sebagai *set testing*. Untuk *fold* yang tersisa, susunan jumlah irisan K dikurang satu yang berbeda dengan *fold* sebelumnya akan menjadi *set training* dan satu irisan sisa sebagai *set testing* (Ozdemir, 2016). Ilustrasi *5-Fold Cross Validation* dapat dilihat pada Gambar 2.6.



Gambar 2.6 Ilustrasi 5-Fold Cross Validation (Ozdemir, 2016)

Pada Gambar 2.6, terdapat 5 bagian *fold* yang terdiri dari titik dengan warna merah sebagai *set training* dan titik dengan warna biru sebagai *set testing* yang berbeda-beda setiap pada setiap *fold*. Hal ini menunjukkan terdapat 5 kasus berbeda dalam model yang sama untuk dilihat apakah model memiliki kinerja yang konsisten pada setiap *fold*.

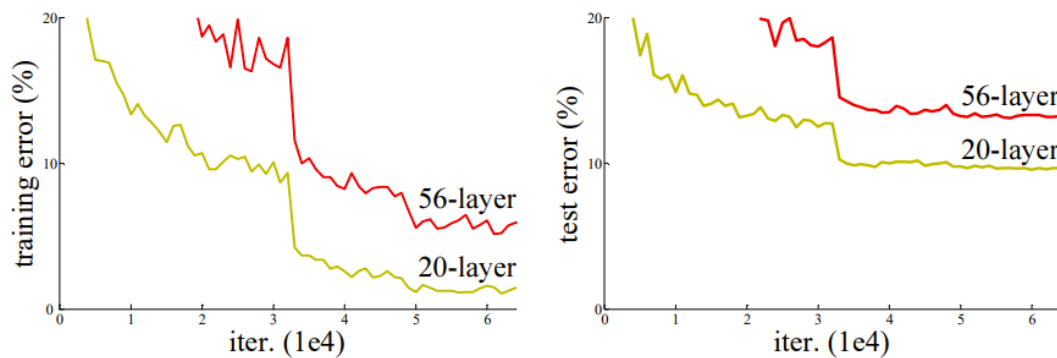
*K-Fold Cross Validation* menawarkan estimasi yang lebih akurat yang dilihat dari kesalahan prediksi daripada *single train-test split* karena mengambil *single train-test split* sebanyak K kali secara independen dan merata-ratakan hasilnya. *K-Fold Cross Validation* menggunakan data secara lebih efisien karena seluruh data pada *dataset* tidak hanya digunakan untuk *single train-test split*, tetapi digunakan untuk *multiple train-test split* sebanyak K kali.

## 2.6 Residual Network (ResNet)

*Residual Network* atau ResNet merupakan arsitektur dari *Convolutional Neural Network* yang dibuat untuk mengatasi permasalahan mengenai tingkat kedalaman pada *Convolutional Neural Network*. Banyak orang mengasumsikan bahwa dengan menambah *layer* pada model akan memberikan akurasi yang lebih optimal, tetapi di titik tertentu model akan mengalami permasalahan degradasi

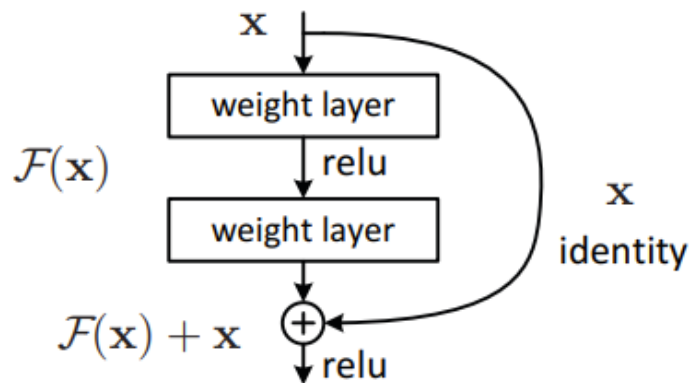


akurasi. Dengan bertambahnya kedalaman pada model, akurasi akan mengalami degradasi secara terus menerus. Akan tetapi, hal tersebut bukan disebabkan oleh *overfitting* dan dengan penambahan *layer* yang lebih banyak, model akan mengalami *training error* yang lebih tinggi (He et al., 2016). Gambar 2.7 merupakan perbandingan nilai *error* antara *Convolutional Neural Network* dengan 20 *layers* dan 56 *layers* yang dilatih pada *dataset* CIFAR-10.



Gambar 2.7 Perbandingan *error* pada *Convolutional Neural Network* 20 *layers* dan 56 *layers* (He et al., 2016)

Pada Gambar 2.7, dapat dilihat bahwa model *Convolutional Neural Network* dengan 20 *layers* memiliki tingkat *error* yang lebih rendah dibandingkan dengan model *Convolutional Neural Network* dengan 56 *layers*. ResNet dibuat untuk mengatasi permasalahan tersebut dengan *layer* yang lebih dalam. Pada arsitektur ResNet, dilakukan juga penambahan koneksi yang bernama *shortcut connection* untuk mempermudah melakukan *identity mapping* yang dilakukan dengan menambahkan *output* dari *layer* sebelumnya ke *output layer* selanjutnya. Kesatuan blok *layer* ini disebut sebagai *residual block*. Gambar 2.8 merupakan ilustrasi *residual block*.



Gambar 2.8 Ilustrasi *Residual Block* (He et al., 2016)

Pada Gambar 2.8, dapat dilihat terdapat *shortcut connection* antara *output* suatu blok *layer* sebelumnya menuju *output* dari blok tersebut. Hal ini memungkinkan ResNet untuk membangun jaringan yang lebih dalam dibandingkan pada arsitektur *Convolutional Neural Network* lainnya sehingga dapat menghasilkan akurasi yang lebih optimal.

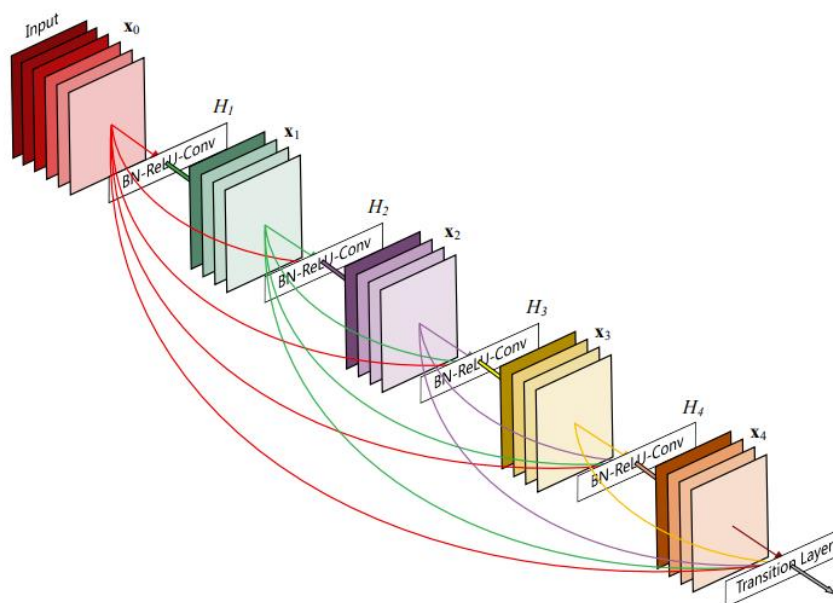
## 2.7 Visual Geometry Group (VGG)

*Visual Geometry Group* atau VGG merupakan arsitektur dari *Convolutional Neural Network* yang memiliki kemampuan kinerja yang baik dengan memanfaatkan kedalaman dari arsitekturnya. Arsitektur VGG diperkenalkan oleh Simonyan dan Zisserman pada kompetisi *ImageNet Challenge* pada tahun 2014 (Simonyan & Zisserman, 2015).

Arsitektur VGG menggunakan blok *layer* dengan *layer* konvolusi dengan *kernel* 3x3 dilanjutkan dengan *max pooling* untuk *downsampling feature map*nya. VGG19 merupakan salah satu versi dari arsitektur VGG dengan menggunakan dua blok dengan dua *layer* konvolusi dan *max pooling* dan tiga blok dengan empat *layer* konvolusi dan *max pooling*.

## 2.8 *Densely Connected Convolutional Networks (DenseNet)*

*Densely Connected Convolutional Networks* atau DenseNet merupakan arsitektur dari *Convolutional Neural Networks* yang merupakan pengembangan dari arsitektur ResNet. Arsitektur DenseNet memiliki koneksi yang menghubungkan setiap *layer* dengan *layer-layer* berikutnya dengan ukuran *feature map* yang sesuai secara langsung (Huang et al., 2017). Gambar 2.9 merupakan ilustrasi arsitektur DenseNet.



Gambar 2.9 Ilustrasi DenseNet (Huang et al., 2017)

Pada Gambar 2.9, dapat dilihat setiap *layer* dengan *layer* berikutnya pada arsitektur DenseNet saling terkoneksi yang menunjukkan setiap *layer* mendapatkan *input* tambahan yang terdiri dari *feature map* dari *layer* sebelumnya. *Feature map* yang digabungkan tidak dilakukan dengan metode penjumlahan biasa seperti pada

arsitektur ResNet, melainkan dengan menggabungkan *feature map* tersebut. Hal ini memungkinkan aliran informasi yang terjadi antar *layer* yang lebih maksimal.

## 2.9 Python

Python merupakan bahasa pemrograman tingkat tinggi yang sangat populer karena bersifat *open-source*, *user-friendly*, dan memiliki banyak *library* yang memudahkan pengguna dalam pengerjaan. Python dibuat oleh Guido van Rossum pada awal 1990. Python memiliki struktur data tingkat tinggi yang efisien dan efektif untuk pemrograman berorientasi objek (Rossum & Drake, 2003).

Dalam sejarahnya, berbagai bahasa pemrograman telah digunakan untuk proses *deep learning* dan pengembangan aplikasi. Namun, Python telah mengalami perkembangan yang pesat, menjadikannya salah satu bahasa pemrograman yang paling banyak digunakan dalam *deep learning* menggunakan *library* seperti SciPy, NumPy, Pandas, Flask, dan lain-lain.

Pada penelitian ini, *library* Flask digunakan untuk mengintegrasikan model *deep learning*. Flask adalah *micro-framework* yang dirancang untuk yang dapat diperluas dari bawah ke atas dengan menyediakan inti yang solid sebagai layanan dasar, sementara ekstensi menyediakan sisanya. Flask digunakan sebagai media untuk memenuhi kebutuhan sebagian aplikasi berbasis *website* seperti *image classification* melalui ekstensi yang terintegrasi dengan Flask (Grinberg, 2018).

## 2.10 TensorFlow

TensorFlow adalah sebuah perangkat lunak *open source* yang digunakan untuk implementasi *machine learning* dan dikembangkan oleh Google pada tahun

2015. TensorFlow memiliki fokus pada komputasi numerik menggunakan *data flow graphs* yang terdiri dari kumpulan simpul atau *nodes* yang mewakili operasi dan memiliki *input* dan *output* (Google Research, 2016).

TensorFlow memiliki elemen bernama *tensor* yang berbentuk *array* berdimensi yang mengalir di sepanjang sisi-sisi *graphs*. *Graphs* juga dapat memiliki sisi-sisi khusus yang disebut *control dependencies*, yang menunjukkan urutan eksekusi antara operasi yang independen. TensorFlow memiliki keunggulan karena merupakan *framework* yang kuat untuk menghasilkan komputasi yang efisien.

## 2.11 OpenCV

OpenCV merupakan suatu *library open source* yang dikembangkan oleh tim Intel Research untuk kebutuhan *computer vision*, berbasis bahasa pemrograman C dan C++. *Library* ini mendukung berbagai sistem operasi seperti Windows, Linux, dan MacOS X, sehingga memungkinkan penggunaannya dalam berbagai fungsi *programming* untuk memproses berbagai jenis tipe gambar (Bradski & Kaehler, 2008).

OpenCV dikembangkan dengan fokus pada pengaplikasian bidang *computer vision* secara *real time*. *Library* ini juga dikembangkan dengan *cost* komputasi yang efisien. *Library* ini memiliki berbagai *framework computer vision* yang dapat digunakan dengan menyediakan lebih dari 500 fungsi yang dapat digunakan secara *real time*. *Library* ini mendukung banyak bidang pengaplikasian seperti inspeksi produk, *medical imaging*, *user interface*, kalibrasi kamera, *stereo vision*, dan robotik.

## 2.12 Confusion Matrix

*Confusion Matrix* merupakan sebuah alat evaluasi yang sangat populer digunakan dalam *image classification*. *Confusion Matrix* berbentuk sebuah tabel yang digunakan untuk menggambarkan performa sebuah model dari data yang diekstrak dari *set testing*. *Confusion Matrix* membandingkan setiap kelas dengan setiap kelas lainnya dan melihat berapa banyak sampel yang benar dan salah dalam klasifikasi (Joshi, 2017). Dengan *Confusion Matrix*, kita dapat melihat seberapa baik model dapat mengklasifikasikan data ke dalam kelas yang benar dan salah. Pada Gambar 2.10 menunjukkan contoh bentuk dari *Confusion Matrix*.

Actual class	Assigned class	
	Positive	Negative
	TP	FN
	FP	TN

Gambar 2.10 Ilustrasi *Confusion Matrix* (Sammut & Webb, 2017)

Pada Gambar 2.10, terdapat empat variabel yang penting untuk dipahami yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Variabel *True Positive* mewakili jumlah prediksi yang benar dalam kelas positif, yaitu ketika model berhasil memprediksi sampel dengan nilai positif dan sesuai dengan nilai aktual yaitu positif. Variabel *True Negative* mewakili jumlah prediksi yang benar dalam kelas negatif, yaitu ketika model berhasil memprediksi sampel dengan nilai negatif dan sesuai dengan nilai aktual yaitu negatif. Variabel *False Positive* atau *Type I Error* mewakili jumlah prediksi yang salah dalam kelas negatif, yaitu ketika model memprediksi sampel dengan nilai

positif tetapi sebenarnya tidak sesuai dengan nilai aktual yaitu negatif. Variabel *False Negative* atau *Type II Error* mewakili jumlah prediksi yang salah dalam kelas positif, yaitu ketika model memprediksi sampel dengan nilai negatif tetapi sebenarnya tidak sesuai dengan nilai aktual yaitu positif.

Dengan menggunakan *Confusion Matrix*, kita dapat mengevaluasi performa model atau sistem secara lebih komprehensif. Dengan memahami nilai-nilai dalam *Confusion Matrix*, kita dapat menghitung berbagai parameter evaluasi seperti *Precision*, *Recall*, dan *F1 Score*. Parameter-parameter ini memberikan gambaran yang lebih ringkas tentang performa model tersebut dalam melakukan klasifikasi citra.

*Precision* adalah akurasi prediksi positif yang berarti performa model dalam mengklasifikasi kelas positif secara benar (Géron, 2019). *Precision* dapat dihitung dengan membagi nilai jumlah prediksi benar dalam kelas positif atau *True Positive* dengan jumlah keseluruhan sampel yang diprediksi sebagai kelas positif seperti pada Rumus 2.1.

$$Precision = \frac{TP}{TP + FP} \quad (2.1)$$

*Recall* atau sensitivitas adalah rasio kasus positif yang dideteksi dengan benar (Géron, 2019). *Recall* dapat dihitung dengan membagi nilai jumlah prediksi benar dalam kelas positif atau *True Positive* dengan jumlah keseluruhan sampel pada kelas positif seperti pada Rumus 2.2.

$$Recall = \frac{TP}{TP + FN} \quad (2.2)$$

*F1 Score* adalah rata-rata harmonik dari *Precision* dan *Recall*. *F1 Score* menjadi cara mudah dan sederhana untuk membandingkan *Precision* dan *Recall* (Géron, 2019). Rata-rata harmonik pada *F1 Score* memberikan bobot yang lebih besar pada nilai yang rendah. *F1 Score* dapat dihitung dengan cara seperti pada Rumus 2.3.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.3)$$

### 2.13 Unified Modelling Language (UML)

*Unified Modeling Language* (UML) merupakan bahasa *graphical* untuk visualisasi, spesifikasi, dan dokumentasi dari sebuah sistem. *Unified Modeling Language* diadopsi oleh *Object Management Group* (OMG) sebagai bahasa pemodelan standar pada tahun 1997. *Unified Modeling Language* mendefinisikan sembilan teknik diagram yaitu *Class Diagram*, *Object Diagram*, *Use Case Diagram*, *Sequence Diagram*, *Collaboration Diagram*, *Statechart Diagram*, *Activity Diagram*, *Component Diagram*, dan *Deployment Diagram* (Booch et al., 1998). Pada penelitian ini, *Unified Modeling Language* yang akan digunakan meliputi *Use Case Diagram*, *Activity Diagram*, *Sequence Diagram*, dan *Deployment Diagram*.




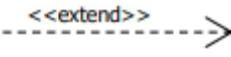

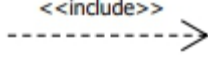
#### 2.13.1 Use Case Diagram

*Use Case Diagram* adalah jenis diagram yang menggambarkan hubungan interaksi antara sistem dan aktor (Booch et al., 1998). *Use Case Diagram* dapat menggambarkan penggunaan dari sebuah sistem. Dalam *Use Case Diagram*



terdapat simbol-simbol yang menjelaskan pada setiap bagian untuk membangun *use case* sesuai dengan sistem yang akan dibuat. Tabel 2.1 merupakan simbol-simbol yang digunakan pada *Use Case Diagram*.



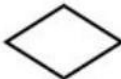



Tabel 2.1 Simbol-simbol pada *Use Case Diagram*

No	Simbol	Keterangan
1	<p><i>Use Case</i></p> 	Fungsionalitas yang disediakan sistem sebagai <i>unit-unit</i> yang saling bertukar pesan antar <i>unit</i> atau aktor.
2	<p><i>Actor</i></p> 	Orang yang berinteraksi dengan sistem yang akan dibuat. Biasanya dinyatakan menggunakan kata benda atau nama aktor.
3	<p><i>Association</i></p> 	Komunikasi antara aktor dengan <i>use case</i> yang berpartisipasi pada diagram.
4	<p><i>Extend</i></p> 	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> , dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri meski tanpa <i>use case</i> tambahan itu. Arah panah mengarah pada <i>use case</i> yang ditambahkan.
5	<p><i>Generalization</i></p> 	Hubungan antara generalisasi dan spesialisasi antar dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6	<p><i>Include</i></p> 	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> , dimana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya. Arah panah <i>include</i> mengarah pada <i>use case</i> yang dibutuhkan.

### 2.13.2 Activity Diagram

*Activity Diagram* adalah jenis diagram yang menggambarkan aliran dari aktivitas ke aktivitas dalam sebuah sistem. Sebuah aktivitas menunjukkan sekumpulan aktivitas, aliran berurutan atau bercabang dari aktivitas ke aktivitas, dan objek yang bertindak dan ditindaklanjuti (Booch et al., 1998). Tabel 2.2 merupakan simbol-simbol yang digunakan pada *Activity Diagram*.


Tabel 2.2 Simbol-simbol pada *Sequence Diagram*



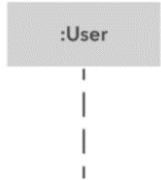


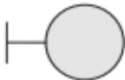


No	Simbol	Keterangan
1	Status awal 	Status awal sebagai penanda tindakan awal atau titik awal untuk sebuah <i>Activity Diagram</i> .
2	Aktivitas 	Aktivitas yang ada atau dilakukan suatu sistem.
3	Percabangan 	Percabangan diperlukan jika ada beberapa pilihan aktivitas atau jalan alternatif.
4	Penggabungan 	Penggabungan <i>flow</i> yang semula dipecah menjadi beberapa bagian.
5	Status akhir 	Status akhir sebagai penanda bahwa suatu proses sudah berakhir.
6	<i>Swimlane</i> 	<i>Swimlane</i> memisahkan <i>Activity Diagram</i> menjadi kolom dan baris untuk membagi tanggung jawab objek-objek yang melakukan suatu aktivitas.

### 2.13.3 *Sequence Diagram*

*Sequence Diagram* adalah jenis diagram yang menggambarkan interaksi yang menekankan urutan waktu dalam sebuah pesan. *Sequence Diagram* menunjukkan sekumpulan objek dan pesan yang dikirim dan diterima oleh objek-objek berupa *instance* kelas, kolaborasi, komponen, atau *node* (Booch et al., 1998). Tabel 2.3 merupakan simbol-simbol yang digunakan pada *Sequence Diagram*.

Tabel 2.3 Simbol-simbol pada *Sequence Diagram*

No	Simbol	Keterangan
1	Aktor 	Aktor menggambarkan seorang pengguna yang berada di luar sistem dan sedang berinteraksi dengan sistem.


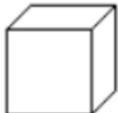
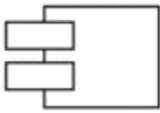



No	Simbol	Keterangan
2	<p><i>Activation Box</i></p> 	Komponen untuk merepresentasikan waktu yang dibutuhkan suatu objek untuk menyelesaikan tugasnya.
3	<p><i>Object</i></p> 	Komponen yang akan mendemonstrasikan bagaimana sebuah objek akan berperilaku dalam sebuah konteks sistem tertentu.
4	<p><i>Lifeline</i></p> 	Komponen untuk menunjukkan kejadian berurutan yang terjadi pada sebuah objek selama proses pembuatan grafik berlangsung.
5	<p><i>Synchronous Message</i></p> 	Simbol yang digunakan untuk memberi isyarat bila pengirim harus menunggu respon pesan sebelum melanjutkan.
6	<p><i>Reply Message</i></p> 	Simbol yang digunakan untuk memberi pesan yang merupakan balasan untuk sebuah panggilan.
7	<p><i>Boundary</i></p> 	<i>Boundary</i> umumnya merupakan tepi atau <i>edge</i> dari sebuah sistem yang biasanya berupa <i>user interface</i> untuk berinteraksi antar sistem.
8	<p><i>Entity</i></p> 	<i>Entity</i> merupakan elemen yang memiliki tugas menyimpan data atau informasi dan umumnya berupa <i>object model</i> .
9	<p><i>Control</i></p> 	<i>Control</i> merupakan elemen untuk mengatur arus informasi dalam sebuah skenario sistem. Elemen ini secara umum dapat mengatur perilaku bisnis dari suatu sistem teknis.

#### 2.13.4 Deployment Diagram

*Deployment Diagram* adalah jenis diagram yang menggambarkan konfigurasi *node* pemrosesan *runtime* beserta komponen pada sistem. *Deployment Diagram* menunjukkan penyebaran statis dari sebuah arsitektur dengan *node* yang

membungkus komponen-komponen (Booch et al., 1998). Tabel 2.4 merupakan simbol-simbol yang digunakan pada *Deployment Diagram*.

Tabel 2.4 Simbol-simbol pada *Deployment Diagram*

No	Simbol	Keterangan
1	<i>Package</i> 	Simbol untuk menggambarkan sebuah bungkusan dari satu atau lebih komponen.
2	<i>Node</i> 	Simbol untuk merepresentasikan <i>hardware</i> atau <i>execution environment</i> .
3	<i>Component</i> 	Simbol untuk menggambarkan komponen sistem.
4	<i>Dependency</i> 	Simbol untuk menggambarkan kebergantungan antar komponen.
5	<i>Link</i> 	Simbol untuk menggambarkan relasi antar koponen.
6	<i>Interface</i> 	Simbol yang memiliki fungsi sebagai antarmuka komponen agar tidak mengakses komponen secara langsung.

## 2.14 Penelitian Terdahulu

Penerapan *deep learning* dengan metode *Convolutional Neural Network* telah beberapa kali digunakan oleh beberapa peneliti sebelumnya. Terdapat penelitian mengenai konteks yang sama seperti penelitian ini yaitu klasifikasi citra sampah dan penerapan *deep learning* pada konteks klasifikasi citra. Maka dari itu, penulis melampirkan tabel yang berisi informasi terkait penelitian terdahulu.

Tabel 2.5 Penelitian Terdahulu

Judul Penelitian	Nama Penulis
Analisis Perbandingan Algoritma Convolutional Neural Network Dan Algoritma Multi-Layer Perceptron Neural Dalam Klasifikasi Citra Sampah	Kelvin Leonardi Kohsasih, Muhammad Dipo Agung Rizky. 2022
Deep Learning untuk Klasifikasi Glaukoma dengan menggunakan Arsitektur EfficientNet	Wahyuni Rizky Perdani, Rita Magdalena, Nor Kumalasari Caecar Pratiwi. 2022

Dari penelitian tersebut yang membedakan dengan penelitian yang akan dilakukan oleh penulis adalah disini metode yang digunakan untuk penelitian serta fokus penelitian ini yang berfokus pada perbandingan *hyperparameter* untuk menemukan konfigurasi *hyperparameter* yang menghasilkan *F1 Score* yang paling optimal.

## **BAB III**

### **ANALISIS DAN PERANCANGAN**

#### **3.1 Fase Analisis**

Analisis kebutuhan penelitian diperlukan agar penelitian dapat berjalan sesuai rencana dan dilaksanakan sebelum penelitian dimulai. Analisis dilakukan dengan mendeskripsikan spesifikasi dari alat penunjang yang digunakan pada penelitian ini. Alat penunjang penelitian ini terbagi menjadi perangkat lunak dan perangkat keras.

##### **3.1.1 Kebutuhan Perangkat Lunak**

Kebutuhan perangkat lunak atau *software* yang digunakan untuk penelitian ini adalah sebagai berikut :

1. Sistem Operasi Microsoft Windows 10
2. Python versi 3.8
3. Kaggle
4. Figma
5. Google Colab dengan spesifikasi *Compute Engine* :
  - a. Nama : Python 3 Google Compute Engine Backend
  - b. RAM : 12.7 GB
  - c. Disk : 78.2 GB
  - d. GPU Device : Tesla T4
  - e. GPU Name : NVIDIA-SMI
  - f. GPU RAM : 15 GB
  - g. CUDA Version : 12.0

### 3.1.2 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras atau *hardware* yang digunakan untuk penelitian ini adalah sebagai berikut :

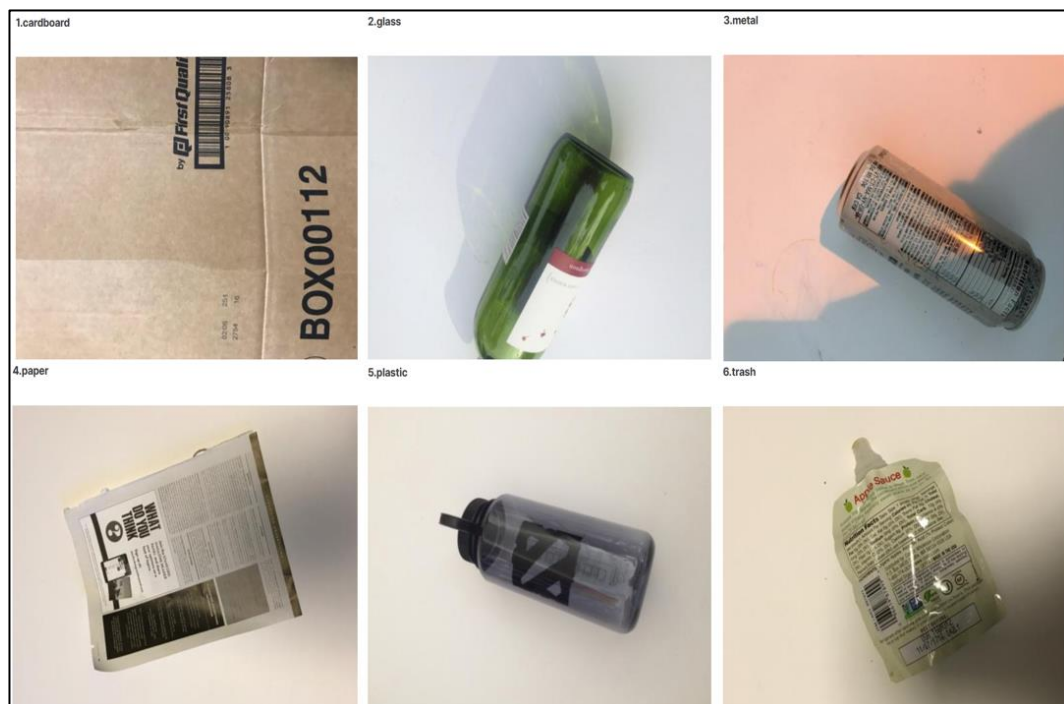
1. Processor : Intel Core i7 Gen 10
2. Harddisk : 1 TB
3. RAM : 16 GB
4. VGA : NVIDIA GeForce GTX 1660 Ti
5. SSD : 512 GB

## 3.2 Fase Penelitian

Penelitian ini dilakukan dengan pengumpulan *dataset* yang akan digunakan, *Data Preprocessing*, pembagian data menjadi data *training* dan *testing*, pembuatan model *Convolutional Neural Network*, pelatihan model, pengujian model, dan evaluasi hasil performa dari setiap model sesuai skenario. Fase penelitian ini yang akan digunakan sebagai acuan dalam penulisan skripsi dan tahapan yang akan dilakukan selanjutnya.

### 3.2.1 Pengumpulan Data

Pengumpulan data yang dilakukan penulis lakukan untuk menunjang proses penelitian didapatkan dari *platform* Kaggle yang bernama Garbage Classification yang diakses pada tanggal 28 September 2023 yang dapat diakses pada tautan <https://www.kaggle.com/datasets/asdasdasdasdas/garbage-classification>. *Dataset* ini terdiri dari 2527 data citra sampah yang terbagi ke 6 kelas yaitu *Cardboard*, *Glass*, *Metal*, *Paper*, *Plastic*, dan *Trash*. Sampel masing-masing kelas dari *dataset* dapat dilihat pada Gambar 3.1.



Gambar 3.1 Sampel Citra Sampah

*Dataset* ini terbagi menjadi 6 kelas, yaitu *Cardboard*, *Glass*, *Metal*, *Paper*, *Plastic*, dan *Trash*. Kelas *Cardboard* terdiri dari sampah-sampah berbahan dasar kardus yang berisi 403 citra. Kelas *Glass* terdiri dari sampah-sampah berbahan dasar kaca yang berisi 501 citra. Kelas *Metal* terdiri dari sampah-sampah berbahan dasar *metal* yang berisi 410 citra. Kelas *Paper* terdiri dari sampah-sampah berbahan dasar kertas yang berisi 594 citra. Kelas *Plastic* terdiri dari sampah-sampah berbahan dasar plastik yang berisi 482 citra. Kelas *Trash* mayoritas terdiri dari sampah-sampah bekas kemasan makanan yang berisi 137 citra.

### 3.2.2 Data Preprocessing

*Data Preprocessing* diawali dengan melakukan tahap *image preprocessing* pada citra yang akan digunakan. Tahap *image preprocessing* dilakukan untuk



mempermudah model untuk mengenali *feature* yang ada pada citra. Parameter dari *Data Preprocessing* dapat dilihat pada Tabel 3.1.

Tabel 3.1 Skenario *Data Preprocessing*

Tahap	Detail	Parameter
<i>Image Resize</i>	<i>Image Resize</i>	224x224
<i>Pretrained Model Preprocess Input</i>	<i>Pretrained Model Preprocess Input</i> sesuai dengan <i>Pretrained Model</i> yang digunakan	Resnet50.preprocess_input, vgg19.preprocess_input, atau densenet.preprocess_input
<i>Image Augmentation</i>	<i>Horizontal Flip</i>	TRUE
	<i>Vertical Flip</i>	TRUE
	<i>Rotation Range</i>	20
	<i>Shear Range</i>	0,2
	<i>Zoom Range</i>	0,2
	<i>Width Shift Range</i>	0,1
	<i>Height Shift Range</i>	0,1

Dari Tabel 3.1, tahap *image preprocessing* yang dilakukan adalah sebagai berikut :

1. *Image Resize*

Pada tahap ini dilakukan *resize* atau perubahan ukuran citra disamakan menjadi ukuran 224x224 *pixel* sesuai dengan kebutuhan *input* pada model ResNet, VGG, dan DenseNet.

2. *Pretrained Model Preprocessing Function*

Pada tahap ini dilakukan *Data Preprocessing* pada citra *input* dengan fungsi *preprocessing input* sesuai dengan *pretrained model* yang digunakan yaitu menggunakan *resnet50.preprocess\_input*, *vgg19.preprocess\_input*, atau *densenet.preprocess\_input* sesuai dengan skenario. Metode *preprocess\_input* ini berfungsi untuk menormalisasi data citra dan perubahan *channel* warna pada citra *input*.

### 3. *Image Augmentation*

Pada tahap ini dilakukan augmentasi citra untuk meningkatkan banyaknya variasi dengan mengubah bentuk citra. *Horizontal flip* dan *vertical flip* yang bernilai TRUE yang berarti citra dapat dibalik secara horizontal dan vertikal, *rotation range* sebesar 20 yang berarti citra dapat diputar maksimal sebesar 20 derajat searah jarum jam atau berlawanan arah jarum jam, *shear range* sebesar 0,2 yang berarti citra dapat dimiringkan maksimal sejauh 20 persen dari ukuran sebelumnya, *zoom range* sebesar 0,2 yang berarti citra dapat diperbesar atau diperkecil maksimal sebesar 20 persen dari ukuran sebelumnya, *width shift range* sebesar 0,1 yang berarti citra dapat digeser secara horizontal maksimal sebesar 10 persen dari lebar citra, dan *height shift range* sebesar 0,1 yang berarti citra dapat digeser secara vertikal maksimal sebesar 10 persen dari tinggi citra.

#### 3.2.3 *Data Splitting*

Pada tahap ini akan dilakukan pembagian data untuk *training* dan *testing* menggunakan metode *K-Fold Cross Validation*. Proses *K-Fold Cross Validation* menggunakan K sebesar 5 yang membagi data menjadi 5 *fold* yang terdiri dari jumlah citra yang seimbang untuk setiap *fold* pada setiap kelasnya dengan data yang diacak. Pengaturan pengacakan data akan diatur oleh *random state* yang ditentukan sehingga untuk komposisi data dari setiap kelas dalam pembagiannya akan sama untuk setiap skenario pelatihan. Pembagian *batch* data untuk *training* dan *testing* akan berbeda setiap *fold* sehingga proses pelatihan model akan menghasilkan performa yang lebih optimal.

### 3.2.4 Perancangan Model

Pada tahap ini akan dilakukan pembuatan model *Convolutional Neural Network* menggunakan metode *transfer learning* menggunakan *pretrained model* yaitu arsitektur ResNet50 dengan 50 *layer*, VGG19 dengan 21 *layer*, dan DenseNet121 dengan 124 *layer* dengan *layer classifier* yang ditentukan. *Pretrained model* tersebut diimpor dari *library* Tensorflow Keras dengan memasukkan ukuran *input* citra sebesar 224x224 *pixel*, mengambil bobot dari *pretrained model*, dan menonaktifkan *layer* pada *pretrained model*. *Layer classifier* pada *pretrained model* diubah sesuai dengan skenario percobaan yang akan dilakukan.

Penelitian dilakukan dengan beberapa skenario pelatihan model dengan percobaan *hyperparameter* yang berbeda-beda untuk mencari hasil *F1 Score* yang paling optimal dari model yang digunakan. *Hyperparameter* yang dibandingkan pada penelitian ini adalah penggunaan jenis *pretrained model* yang berbeda, penggunaan *Dropout Layer* dengan nilai yang berbeda, dan penggunaan jenis *Optimizer* yang berbeda. Nilai *hyperparameter* yang digunakan dapat dilihat pada Tabel 3.2.

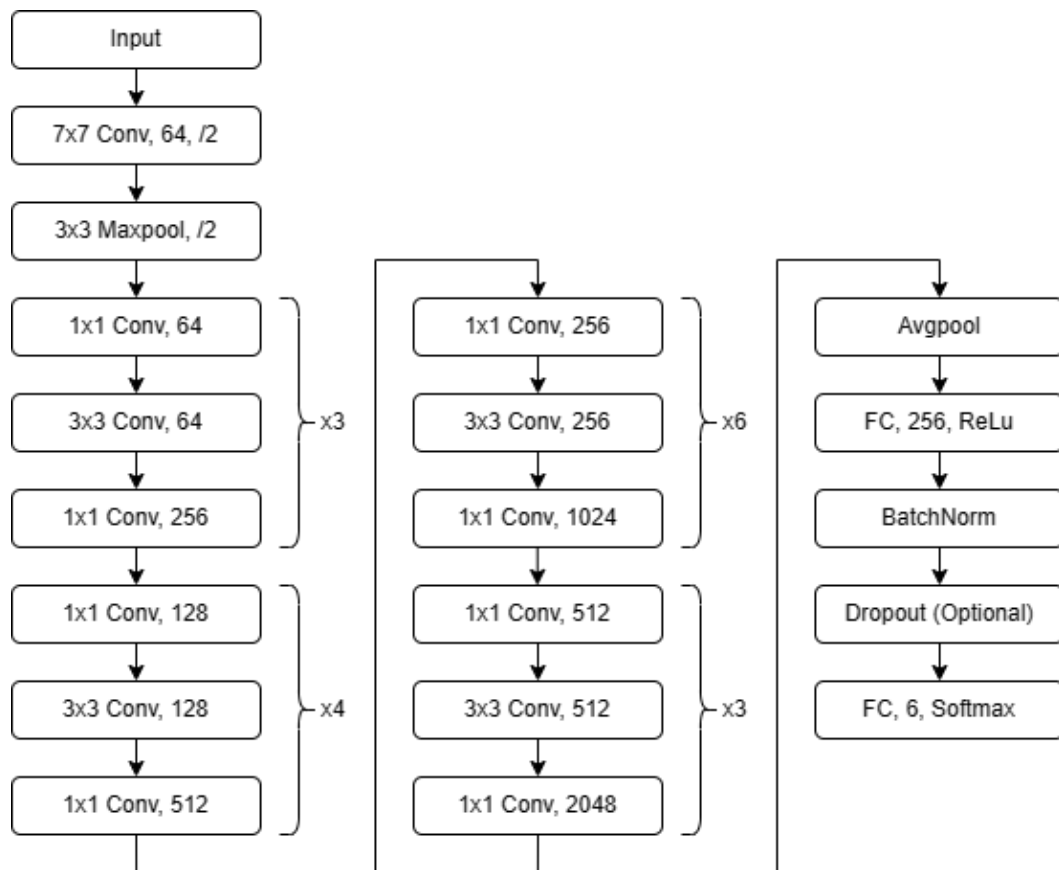
Tabel 3.2 Skenario Perbandingan *Hyperparameter*

Skenario	<i>Pretrained Model</i>	<i>Optimizer</i>	<i>Dropout Layer</i>
1	ResNet50	Adam	<i>No Dropout</i>
2	ResNet50	Adam	0,2
3	ResNet50	Adam	0,5
4	ResNet50	Nadam	<i>No Dropout</i>
5	ResNet50	Nadam	0,2
6	ResNet50	Nadam	0,5
7	ResNet50	AdamW	<i>No Dropout</i>
8	ResNet50	AdamW	0,2
9	ResNet50	AdamW	0,5
10	VGG19	Adam	<i>No Dropout</i>

Skenario	<i>Pretrained Model</i>	<i>Optimizer</i>	<i>Dropout Layer</i>
11	VGG19	Adam	0,2
12	VGG19	Adam	0,5
13	VGG19	Nadam	<i>No Dropout</i>
14	VGG19	Nadam	0,2
15	VGG19	Nadam	0,5
16	VGG19	AdamW	<i>No Dropout</i>
17	VGG19	AdamW	0,2
18	VGG19	AdamW	0,5
19	DenseNet121	Adam	<i>No Dropout</i>
20	DenseNet121	Adam	0,2
21	DenseNet121	Adam	0,5
22	DenseNet121	Nadam	<i>No Dropout</i>
23	DenseNet121	Nadam	0,2
24	DenseNet121	Nadam	0,5
25	DenseNet121	AdamW	<i>No Dropout</i>
26	DenseNet121	AdamW	0,2
27	DenseNet121	AdamW	0,5

Skenario pelatihan model ini dilakukan secara kombinasi untuk setiap *hyperparameter* dengan total sebanyak 27 skenario dengan 3 perbandingan *hyperparameter* yang masing-masing memiliki 3 skenario. Perbandingan *pretrained model* yang digunakan yaitu ResNet50, VGG19, dan DenseNet121 karena *model* tersebut memiliki karakteristik yang cocok dalam tugas untuk mengklasifikasikan citra. *Layer classifier* yang digunakan memiliki *layer* yaitu *Global Average Pooling*, *Dense layer* dengan *units* sebesar 256 dan menggunakan *activation ReLu*, *Batch Normalization*, *Dropout Layer* sesuai dengan skenario, dan *Dense layer* dengan *units* sebanyak jumlah kelas dari dataset yaitu 6 dan menggunakan *activation softmax*. Skenario penggunaan *Dropout Layer* yaitu tidak menggunakan *Dropout Layer*, menggunakan *Dropout Layer* sebesar 0,2, atau menggunakan *Dropout Layer* sebesar 0,5. Skenario penggunaan *Optimizer* yaitu

menggunakan Adam, Nadam, atau AdamW. Arsitektur model untuk skenario yang menggunakan *pretrained model* ResNet50 dapat dilihat pada Gambar 3.2.



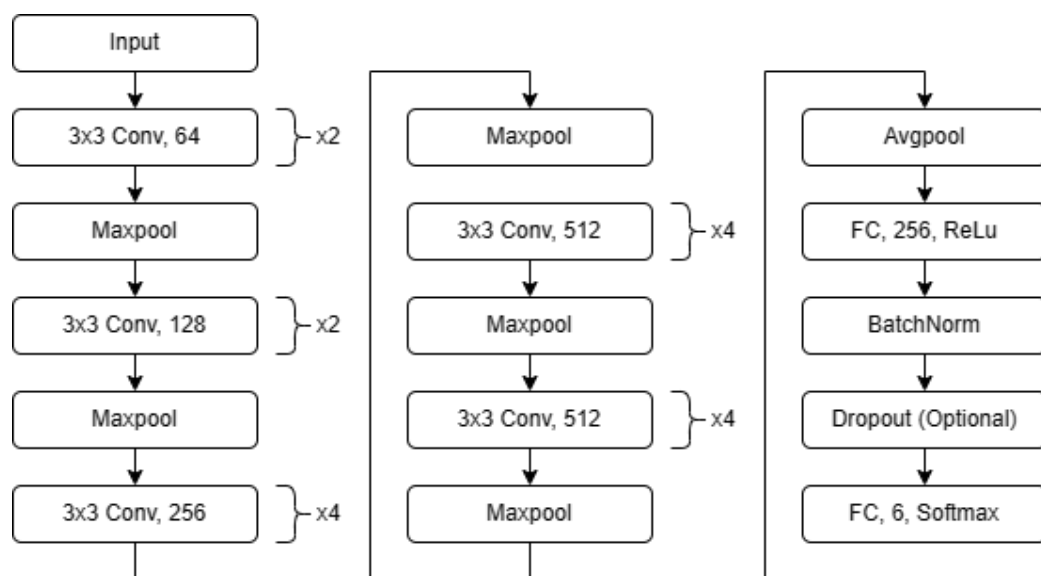
Gambar 3.2 Skenario Arsitektur Model ResNet50

Pada Gambar 3.2 dapat dilihat arsitektur ResNet50 dan *layer classifier* yang digunakan dengan penjelasan sebagai berikut :

1. Ukuran *input* yang digunakan sebesar 224x224 *pixel*.
2. *Model* akan mengambil bobot dari model ResNet50, tetapi citra tidak dilatih menggunakan *layer* pada model ResNet50.

3. Citra dilatih menggunakan *layer classifier* sesuai dengan skenario yaitu tidak menggunakan *Dropout Layer*, menggunakan *Dropout Layer* sebesar 0,2, atau menggunakan *Dropout Layer* sebesar 0,5.

Arsitektur model untuk skenario yang menggunakan *pretrained model* VGG19 dapat dilihat pada Gambar 3.3.

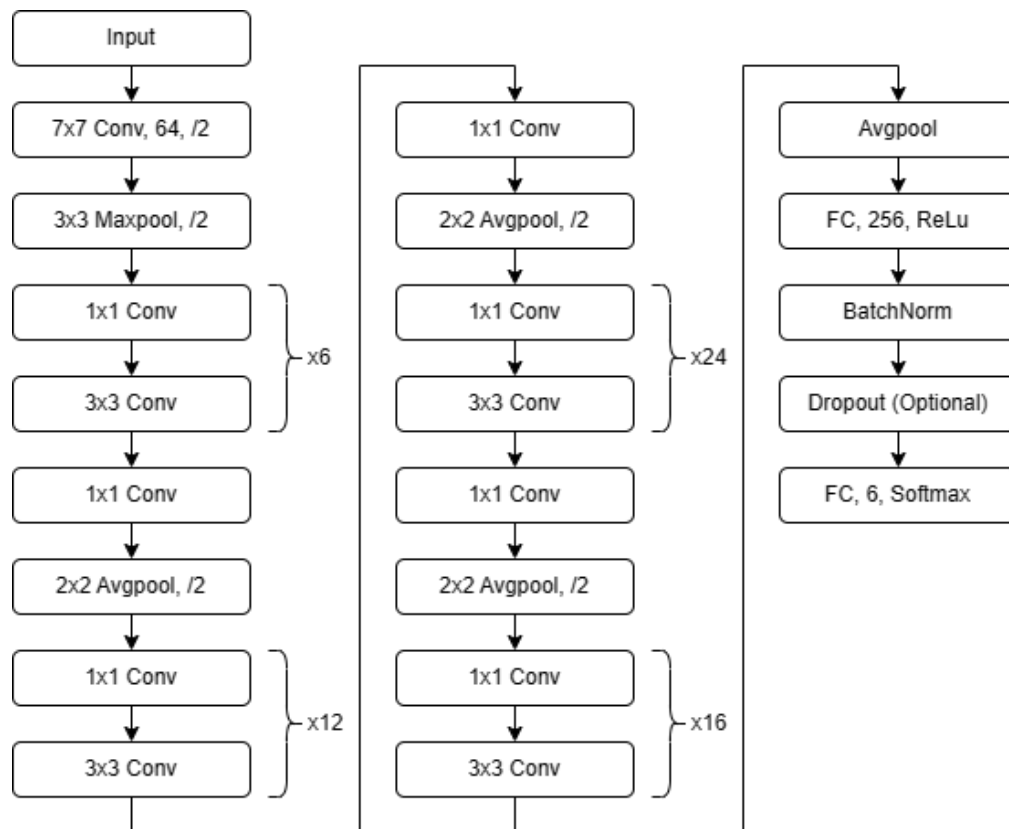


Gambar 3.3 Skenario Arsitektur Model VGG19

Pada Gambar 3.3 dapat dilihat arsitektur VGG19 dan *layer classifier* yang digunakan dengan penjelasan sebagai berikut :

1. Ukuran *input* yang digunakan sebesar 224x224 *pixel*.
2. *Model* akan mengambil bobot dari model VGG19, tetapi citra tidak dilatih menggunakan *layer* pada model VGG19.
3. Citra dilatih menggunakan *layer classifier* sesuai dengan skenario yaitu tidak menggunakan *Dropout Layer*, menggunakan *Dropout Layer* sebesar 0,2, atau menggunakan *Dropout Layer* sebesar 0,5.

Arsitektur model untuk skenario yang menggunakan *pretrained model* DenseNet121 dapat dilihat pada Gambar 3.4.



Gambar 3.4 Skenario Arsitektur Model DenseNet121

Pada Gambar 3.4 dapat dilihat arsitektur DenseNet121 dan *layer classifier* yang digunakan dengan penjelasan sebagai berikut :

1. Ukuran *input* yang digunakan sebesar 224x224 *pixel*.
2. *Model* akan mengambil bobot dari model DenseNet121, tetapi citra tidak dilatih menggunakan *layer* pada model DenseNet121.
3. Citra dilatih menggunakan *layer classifier* sesuai dengan skenario yaitu tidak menggunakan *Dropout Layer*, menggunakan *Dropout Layer* sebesar 0,2, atau menggunakan *Dropout Layer* sebesar 0,5.

### 3.2.5 Pelatihan Model

Pada tahap ini dilakukan pelatihan model sesuai dengan skenario pelatihan. Pelatihan model menggunakan beberapa *hyperparameter* sesuai skenario yaitu *Optimizer* sesuai dengan skenario yaitu Adam, Nadam, atau AdamW, *learning rate* sebesar 0,0001, dan *epochs* sebanyak 10 setiap *fold* dengan total 50 *epochs* secara keseluruhan untuk setiap skenario pelatihan. Matriks pengukuran yang digunakan dalam pelatihan model yaitu akurasi dan *loss* untuk setiap *epoch* serta waktu komputasi untuk setiap *epoch* dan *step* setiap *epoch*.

### 3.2.6 Evaluasi Model

Pada tahap ini dilakukan evaluasi pada model yang telah dilatih menggunakan data *testing* yang sudah dipisah pada proses *data splitting* menggunakan *K-Fold Cross Validation*. Evaluasi dilakukan menggunakan metode *Confusion Matrix* dari *library scikit-learn*. *Confusion Matrix* akan dibuat untuk setiap *fold* dan rata-rata akhir dari keseluruhan *fold* yang diarsipkan dan didokumentasikan dalam bentuk gambar. Berdasarkan *Confusion Matrix* keseluruhan, nilai *True Positive*, *True Negative*, *False Positive*, dan *False Negative* dari setiap kelasnya dapat dihitung untuk setiap kelasnya. Berdasarkan hasil tersebut, maka nilai *Precision*, *Recall*, dan *F1 Score* dapat dihitung menggunakan Rumus 2.1, Rumus 2.2, dan Rumus 2.3 tentang perhitungan *Precision*, *Recall*, dan *F1 Score* pada sub bab *Confusion Matrix*.

### 3.2.7 Konversi Model

Pada tahap ini dilakukan konversi model yang memiliki performa dan hasil yang paling optimal dari semua skenario yang telah dilakukan. Model dikonversi



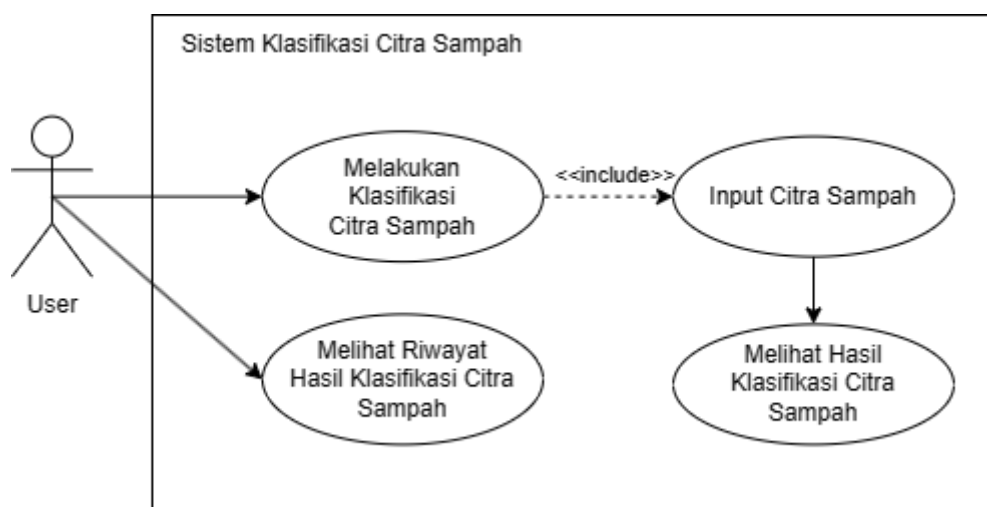
dalam format *file h5* yang dapat digunakan untuk memprediksi citra sesuai *input* dari pengguna. Proses konversi model ini dilakukan karena untuk aplikasi berbasis *website* dapat menggunakan model untuk memprediksi citra dalam bentuk *model* berformat *h5*. Model dalam format *file h5* dapat diintegrasikan dengan aplikasi berbasis *website* menggunakan Flask Application untuk proses *input* dan *output*.

### 3.3 Fase Perancangan Aplikasi

Pada tahapan ini dilakukan perancangan mengenai aplikasi berbasis *website* yang akan dibuat dengan menggunakan *framework* Laravel untuk memudahkan pembuatan aplikasi dengan fitur *Model-View-Controller*. Perancangan yang dibuat berupa pembuatan desain aplikasi dan UML yang terdiri dari *Use Case Diagram*, *Activity Diagram*, *Sequence Diagram*, dan *Deployment Diagram*.

#### 3.3.1 Use Case Diagram

Pada bagian ini berisi *Use Case Diagram* dari aplikasi klasifikasi citra sampah berbasis *website*.

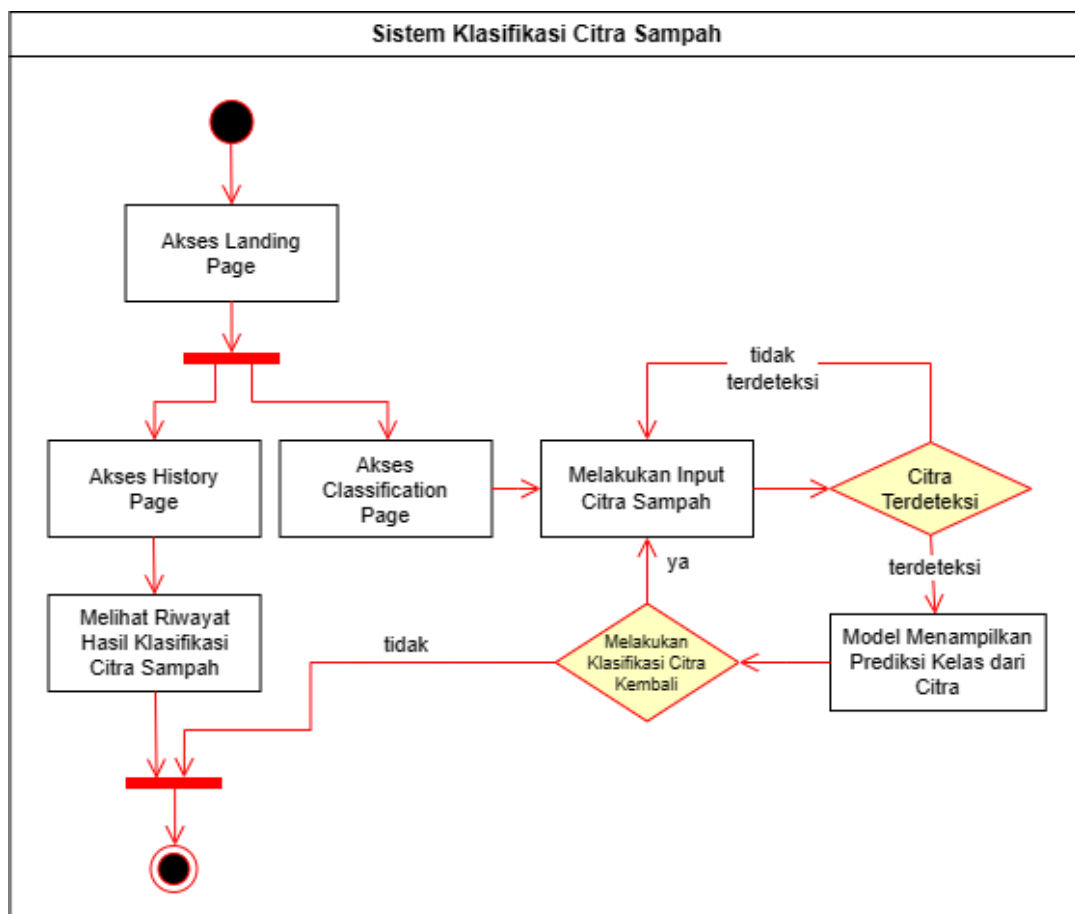


Gambar 3.5 Use Case Diagram

Pada Gambar 3.5, hanya terdapat satu aktor yaitu *user* atau pengguna. *User* dapat melakukan klasifikasi citra sampah dengan syarat *user* diperlukan untuk melakukan *input* citra sampah sebelumnya yang direpresentasikan dengan notasi *include*. Setelah *user* melakukan *input* citra, *user* dapat melihat hasil klasifikasi citra sampah berdasarkan citra yang *user input*. *User* juga dapat melihat riwayat hasil klasifikasi citra sampah yang pernah dilakukan pada aplikasi.

### 3.3.2 Activity Diagram

Pada bagian ini berisi *Activity Diagram* dari aplikasi klasifikasi citra sampah berbasis *website*.

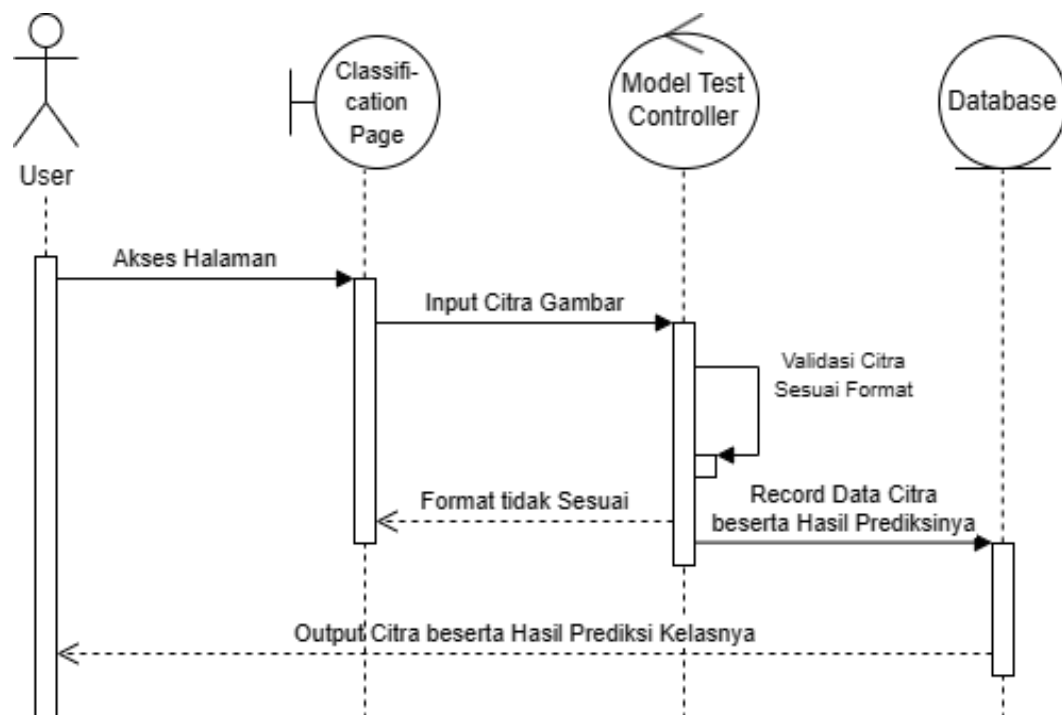


Gambar 3.6 Activity Diagram

Pada Gambar 3.6, terdapat 2 aktivitas utama yang dapat dilakukan yaitu melakukan klasifikasi citra sampah dan melihat riwayat hasil klasifikasi citra sampah. *User* dapat melakukan klasifikasi dengan mengakses halaman *classification page* dan melakukan *input* citra. Apabila citra *input* terdeteksi, maka model akan menampilkan citra *input* beserta prediksi kelas dari citra tersebut. *User* memiliki pilihan untuk melakukan klasifikasi citra sampah kembali atau tidak melakukannya kembali. *User* juga dapat melihat riwayat hasil klasifikasi yang pernah dilakukan pada aplikasi ini.

### 3.3.3 Sequence Diagram

Pada bagian ini berisi *Sequence Diagram* dari aplikasi klasifikasi citra sampah berbasis *website*.

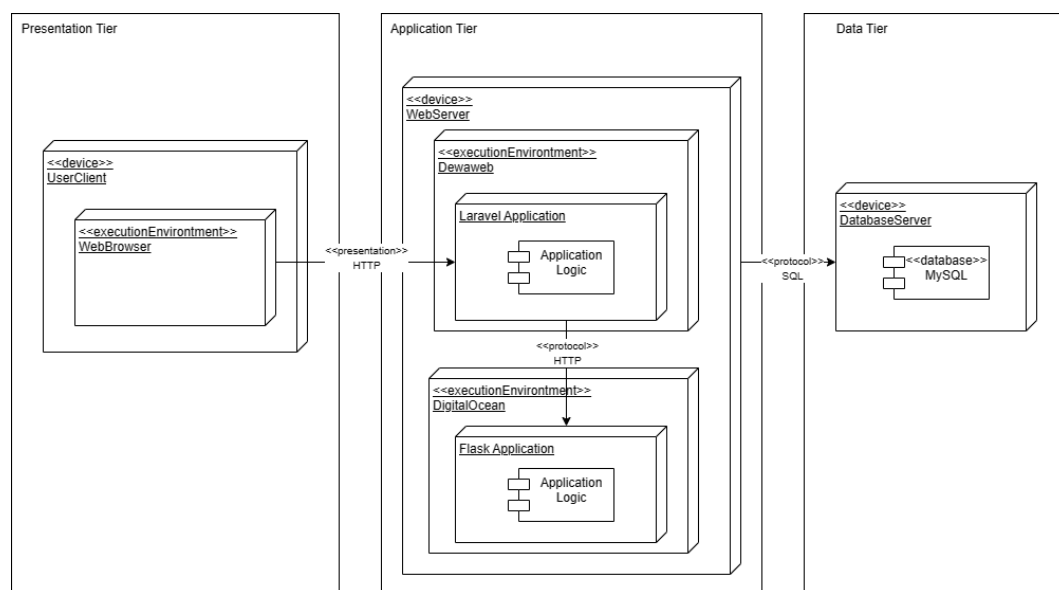


Gambar 3.7 *Sequence Diagram*

Pada Gambar 3.7, terdapat aktor yaitu *user* atau pengguna. *User* dapat melakukan *input* citra pada *classification page*. Setelah citra terinput, maka akan dilakukan validasi *format* citra oleh Model Test Controller. Apabila sesuai, maka citra akan diklasifikasi oleh model dan disimpan *record* citra beserta prediksi kelasnya pada database. Citra beserta prediksi kelasnya akan dikirimkan kembali kepada *user* pada halaman *prediction page*.

### 3.3.4 Deployment Diagram

Pada bagian ini berisi *Deployment Diagram* dari aplikasi klasifikasi citra sampah berbasis *website*.



Gambar 3.8 *Deployment Diagram*

Pada Gambar 3.8, terdapat 3 *tier* yaitu *presentation tier* yang berisi *device* dari *user* yaitu *web browser* sebagai *execution environment*, *application tier* yang berisi *device* pada aplikasi yaitu *web server* yang memiliki 2 komponen yaitu *web hosting* yang berisi *Laravel Application* dan *server* yang berisi *Flask Application*,

dan *data tier* yang berisi *device* yaitu *database server* berisi komponen MySQL. *Node* Laravel Application dan Flask Application masing-masing memiliki komponen *application logic* di dalamnya. *Web browser* pada *presentation tier* memiliki relasi ke *node* Laravel Application yang digunakan untuk *user* melakukan *request* ke Laravel Application agar dapat mengakses aplikasi. Pada *node* Laravel Application memiliki relasi ke *node* Flask Application yang digunakan untuk memprediksi citra *input* dari *user*. *Node web server* pada *application tier* memiliki relasi ke *node database server* untuk menyimpan *record* hasil *application logic* yang dilakukan pada Laravel Application.

### 3.3.5 Wireframe Aplikasi

Aplikasi terdiri dari empat halaman yang mencakup *landing page*, *classification page*, *prediction page*, dan *history page*. *Landing page* merupakan halaman awal yang akan dikunjungi pengguna pada saat pertama kali mengakses aplikasi.



Gambar 3.9 Wireframe Landing Page

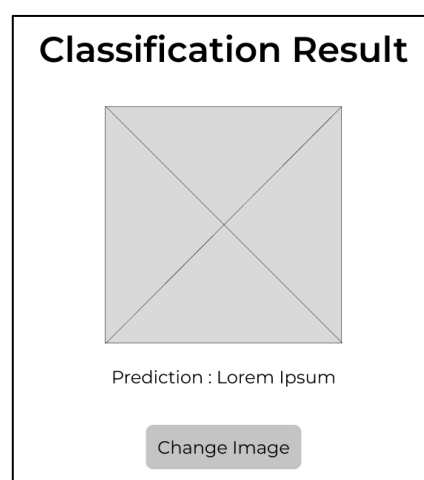
Pada Gambar 3.9, terdapat elemen tombol di tengah-tengah tampilan yang merujuk pada akses halaman *classification page*. Terdapat juga elemen untuk merujuk ke akses halaman *landing page*, *classification page*, dan *history page* pada *navigation bar*.



The wireframe for the 'Classify Garbage' page is enclosed in a rectangular border. At the top center, the title 'Classify Garbage' is displayed in a bold, black font. Below the title, there is a line of placeholder text: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore'. Underneath the text, there is a file selection interface consisting of a rectangular button labeled 'Choose File' and the text 'No file chosen' to its right. Below this, centered, is a gray rectangular button with the text 'Classify Image' in white.

Gambar 3.10 Wireframe Classification Page



Pada Gambar 3.10, terdapat *form* yang berisi *input* untuk tipe data *file* untuk pengguna mengupload gambar yang ingin diklasifikasikan. *Form input file* hanya dapat menerima satu gambar untuk satu kali prediksi. Pengguna dapat melakukan *submit form* dengan gambar yang sudah diupload dengan tombol di bawah *form*.



The wireframe for the 'Classification Result' page is enclosed in a rectangular border. At the top center, the title 'Classification Result' is displayed in a bold, black font. Below the title, there is a large square placeholder for an image, represented by a gray square with a black 'X' drawn across it from corner to corner. Underneath the image placeholder, the text 'Prediction : Lorem Ipsum' is displayed. At the bottom center, there is a gray rectangular button with the text 'Change Image' in white.

Gambar 3.11 Wireframe Prediction Page

Pada Gambar 3.11, hasil prediksi gambar yang *diupload* pengguna akan ditampilkan pada halaman ini beserta dengan kelas yang diprediksi oleh *model*. *User* dapat mengganti gambar dengan mengakses kembali halaman *classification page* dengan tombol di bawah hasil prediksi.

Classification History			
No	Image	Prediction	Timestamp
1		Lorem Ipsum	Lorem Ipsum
2		Lorem Ipsum	Lorem Ipsum

Gambar 3.12 Wireframe History Page

Pada Gambar 3.12, citra sampah beserta hasil prediksi yang *diupload* oleh pengguna dapat dilihat dalam bentuk tabel. Tabel ini berisi nomor, citra yang *diupload*, hasil prediksi, dan *timestamp* atau jam dan tanggal dari pengguna *mengupload* gambar pada *form*.

## BAB IV

### HASIL DAN PEMBAHASAN

#### 4.1 Implementasi Pembuatan Model

Implementasi pembuatan model sesuai dengan skenario melalui beberapa tahapan pada bab sebelumnya. Implementasi program dimulai dari tahap *Data Preprocessing* hingga tahap evaluasi dan konversi model. Pada tahapan-tahapan ini diperlukan pembuatan *function* dan metode untuk menunjang proses pembuatan model.

##### 4.1.1 *Import Library*

Tahap pertama pada implementasi program yaitu dengan memuat *library-library* yang digunakan pada program. Setelah itu dataset dimuat terlebih dahulu yang sebelumnya sudah diupload pada Google Drive yang sudah dipisahkan setiap kelas menjadi folder masing-masing. Pada Google Colab, Google Drive harus *dimounting* terlebih dahulu. Setelah berhasil *dimounting*, data akan dimuat dengan menentukan *directory path* sesuai dengan tempat penyimpanan data pada Google Drive.

Script/kode untuk *import library-library* yang digunakan :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import urllib
import itertools
import random, os, glob
import shutil
from imutils import paths
from sklearn.utils import shuffle
from urllib.request import urlopen
```



```

import warnings
warnings.filterwarnings("ignore")

from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import KFold
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization,
GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator

```

#### 4.1.2 Data Preprocessing

Pada tahap *Data Preprocessing*, *dataset* harus terlebih dahulu dimuat sebelum dilakukan proses-proses *Data Preprocessing*. Salah satu proses *Data Preprocessing* yaitu *image resize* dilakukan setelah data dimuat dan terpisah dengan tahap-tahap *Data Preprocessing* selanjutnya. Penentuan skala ukuran untuk *image resizing* ditentukan sebelum data dimuat.

Script/kode untuk *Loading* dataset :

```

target_size = (224, 224)
waste_labels = {"cardboard":0, "glass":1, "metal":2, "paper":3, "plastic":4, "trash":5}

def load_dataset(path, target_size):
    x = []
    labels = []
    image_paths = sorted(list(paths.list_images(path)))
    for image_path in image_paths:
        img = cv2.imread(image_path)
        img = cv2.resize(img, target_size)
        x.append(img)
        label = image_path.split(os.path.sep)[-2]
        labels.append(waste_labels[label])
    x, labels = shuffle(x, labels, random_state=42)
    input_shape = (target_size[0], target_size[1], 3)
    print("X shape:", np.array(x).shape)
    print("Total Class:", len(np.unique(labels)))
    print("Total Data:", len(labels))
    print("Input Shape:", input_shape)
    return x, labels, input_shape

```

```
x, labels, input_shape = load_dataset(dir_path, target_size)
```

*Dataset* dimuat dengan *function load\_dataset* dengan parameter *directory path dataset* dan *target size*. Data citra dibaca menggunakan *library* OpenCV dengan perintah *cv2.imread*. Setelah berhasil dibaca, data citra akan diubah ukurannya dengan perintah *cv2.resize* sesuai dengan ukuran yang telah ditentukan sebelumnya yaitu *224x224 pixel*. Data citra yang telah disesuaikan ukurannya disimpan di dalam variabel *x*. Label dari setiap data citra yang sudah dipisahkan terlebih dahulu disimpan dalam variabel *labels*. Data citra beserta *label* kemudian diacak dalam variabel yang sama agar komposisi dari citra dengan *label* dalam variabel teracak.

Data yang telah dimuat dapat divisualisasikan untuk mengecek data dan label sudah sesuai dengan fungsi *visualize\_img* dengan parameter *x*, *labels*, dan jumlah citra yang ingin ditampilkan. Untuk tahap ini, data citra yang akan dicek yaitu sebanyak 10 citra.

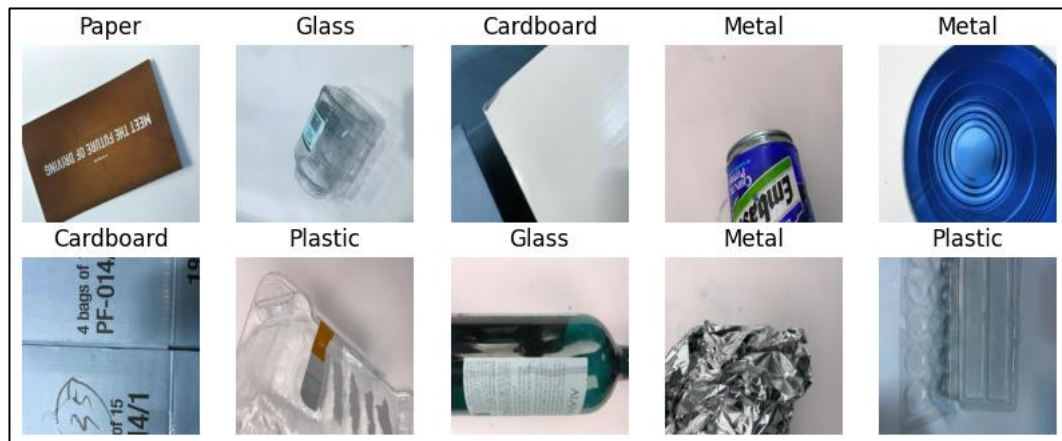
Script/kode untuk visualisasi data citra :

```
def visualize_img(image_batch, labels, num_of_img):
    plt.figure(figsize=(10, 10))

    for n in range(num_of_img):
        ax = plt.subplot(5, 5, n+1)
        plt.imshow(image_batch[n])
        plt.title(np.array(list(waste_labels.keys()))[to_categorical(labels,
num_classes=len(np.unique(labels)))[n] == 1][0].title())
        plt.axis("off")

visualize_img(x, labels, 10)
```

Fungsi *visualize\_img* berisi data citra dan label yang dikeluarkan dalam bentuk kanvas menggunakan *library matplotlib*. Hasil visualisasi data citra dapat dilihat pada gambar 4.1.



Gambar 4.1 Visualisasi Sampel Data

Pada Gambar 4.1, dapat terlihat 10 data citra acak beserta kelasnya sudah sesuai yang berarti data sudah dimuat secara benar. Tahap selanjutnya yang dilakukan yaitu tahap *Data Preprocessing* untuk *pretrained model preprocess input* dan *image augmentation*.

Script/kode untuk *pretrained model preprocess input* dan *image augmentation* :

```
def CNN_data_preparation():
    train = ImageDataGenerator(
        preprocessing_function=tf.keras.applications.resnet50.preprocess_input,
        horizontal_flip=True,
        vertical_flip=True,
        rotation_range=20,
        shear_range=0.2,
        zoom_range=0.2,
        width_shift_range=0.1,
        height_shift_range=0.1,
    )

    train_generator = train.flow_from_directory(
        directory=dir_path,
        target_size=(target_size),
        batch_size=32,
        class_mode="categorical",
        subset="training"
    )
    return train_generator

train_generator = CNN_data_preparation()
```

Fungsi ini berfungsi untuk melakukan *Data Preprocessing* yaitu *pretrained model preprocess input* dan *data augmentation* serta untuk mengenerate data citra untuk proses *training*. Proses ini menggunakan *function ImageDataGenerator* dari *library TensorFlow Keras Preprocessing Image* untuk mengenerate data citra dengan parameter yang dilakukan yaitu *preprocessing\_function* mengambil fungsi *preprocess\_input* dari setiap *pretrained model* yang diambil dari *library Keras Application*.

#### 4.1.3 Data Splitting

Tahap *data splitting* diawali dengan inisiasi metode *5-Fold Cross Validation* untuk pembagian data *training* dan *testing* dengan *n* sebesar 5 yang berarti *dataset* dibagi menjadi 5 *fold* dengan pembagian data citra untuk setiap *class* dilakukan secara acak dan memiliki komposisi setiap *class* yang seimbang.

Script/kode untuk inisiasi *5-Fold Cross Validation* :

```
n_splits = 5
kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)
```

Inisiasi metode *5-Fold Cross Validation* diawali dengan inisiasi *n\_splits* sebesar 5. Setelah itu *parameter* untuk metode *5-Fold Cross Validation* diinisiasikan dalam variabel *kf* dengan fungsi *KFold* yang memiliki parameter *n\_splits*, *shuffle* bernilai *True* yang berarti data citra yang digunakan akan diacak, dan *random\_state* yang merupakan kode untuk proses ini.

#### 4.1.4 Pembuatan Model

Tahap pembuatan model diawali dengan mengambil *pretrained model* sesuai dengan skenario yang digunakan dari *library TensorFlow Keras Application*.

*Pretrained model* yang diambil yaitu ResNet50, VGG19, atau DenseNet121 sesuai dengan skenario yang ingin diuji.

Script/kode untuk inisiasi *pretrained model* ResNet50 :

```
img_shape = (224, 224, 3)

base_model = tf.keras.applications.ResNet50(input_shape=img_shape,
                                             include_top=False,
                                             weights='imagenet')

for layer in base_model.layers:
    layer.trainable = False
```

Script/kode untuk inisiasi *pretrained model* VGG19 :

```
img_shape = (224, 224, 3)

base_model = tf.keras.applications.VGG19(input_shape=img_shape,
                                           include_top=False,
                                           weights='imagenet')

for layer in base_model.layers:
    layer.trainable = False
```

Script/kode untuk inisiasi *pretrained model* DenseNet121 :

```
img_shape = (224, 224, 3)

base_model = tf.keras.applications.DenseNet121(input_shape=img_shape,
                                                include_top=False,
                                                weights='imagenet')

for layer in base_model.layers:
    layer.trainable = False
```

Pada setiap inisiasi *pretrained model* memiliki *parameter* yang sama yaitu *input\_shape* yang diinisiasikan terlebih dahulu sebesar 224x224 pixel sesuai dengan ukuran input citra dari setiap *pretrained model*, *include\_top* dengan nilai *False* yang berarti *layer classifier* dari setiap *pretrained model* tidak dimasukkan ke dalam model, dan *weights* yang bernilai *imagenet* yang berarti model mengambil bobot dari hasil pelatihan *pretrained model* dengan *dataset* Image Net. Setelah

model diinisiasikan, setiap *layer* pada model dikonfigurasi dengan *layer.trainable* bernilai *False* yang berarti hanya mengambil bobot dari model tetapi tidak mengambil *layer* tersebut untuk digunakan dalam pelatihan model. Script/kode untuk inisiasi model keseluruhan :

```
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(units=256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(units=6, activation='softmax'))
model.compile(optimizer=Adam(learning_rate=1e-4), loss='categorical_crossentropy',
metrics=['accuracy'])
```

Setelah *pretrained model* diinisiasikan, model untuk pelatihan diinisiasikan dalam variabel baru dengan model *Sequential* dan diisi dengan *pretrained model* yang telah diinisiasikan sebelumnya. Setelah itu, *layer classifier* dibuat dengan *layer GlobalAveragePooling2D*, *Fully Connected* sebesar 256 *units* dengan fungsi aktivasi ReLu, *Batch Normalization*, *Dropout Layer* secara opsional sesuai dengan skenario, dan *Fully Connected* sebesar 6 *units* sesuai jumlah *class* dengan fungsi aktivasi *softmax*. Setelah seluruh *layer* dimasukkan, model kemudian *dcompile* dengan *hyperparameter Optimizer* Adam, Nadam, atau AdamW sesuai skenario dengan *learning rate* sebesar 0,0001, fungsi *loss* menggunakan *categorical crossentropy*, dan matriks pengukuran yaitu akurasi pelatihan.

#### 4.1.5 Pelatihan Model

Pelatihan model dilakukan dengan proses *looping* menggunakan variabel *kf* yang telah diinisiasi pada tahap inisiasi *5-Fold Cross Validation* dengan membagi *index* untuk proses *training* dan *testing*. Data yang akan digunakan untuk *set*

*training* dan *set testing* untuk setiap *fold* kemudian *digenerate* kembali di dalam *looping*.

Script/kode untuk pelatihan model :

```
def generate_data(data_generator, indices):
    while True:
        for i in indices:
            batch_data, batch_labels = data_generator[i]
            yield batch_data, batch_labels

for train_indices, val_indices in kf.split(train_generator):
    train_data_generator = generate_data(train_generator, train_indices)
    val_data_generator = generate_data(train_generator, val_indices)

    print('Training Per Fold')

    history = model.fit(train_data_generator,
                        validation_data=val_data_generator,
                        epochs=10,
                        verbose=2,
                        steps_per_epoch=len(train_indices),
                        validation_steps=len(val_indices),
                        )

    train_metrics.append(history.history['accuracy'])
    val_metrics.append(history.history['val_accuracy'])

    validation_labels = []
    validation_predictions = []
    for _ in range(len(val_indices)):
        batch_data, batch_labels = next(val_data_generator)
        validation_labels.extend(np.argmax(batch_labels, axis=1))
        validation_predictions.extend(np.argmax(model.predict(batch_data), axis=1))

    all_validation_labels.extend(validation_labels)
    all_validation_predictions.extend(validation_predictions)

    classification_report_str = classification_report(validation_labels, validation_predictions,
                                                    target_names=["cardboard", "glass", "metal", "paper", "plastic", "trash"])
    all_classification_reports.append(classification_report_str)

    confusion_mtx = confusion_matrix(validation_labels, validation_predictions)
    all_confusion_matrices.append(confusion_mtx)
```

Fungsi *generate\_data* berfungsi untuk *generate* data *training* dan *testing* untuk setiap *fold*. Proses *training* data menggunakan data yang telah *digenerate* dan

*hyperparameter epochs* sebesar 10 setiap *fold*, *verbose* bernilai 2 yang berarti informasi setiap *epoch* akan ditampilkan secara lengkap pada proses pelatihan, dan ukuran *batch* untuk setiap *epoch* untuk *training* dan *testing* yang ditentukan dari setiap *index training* dan *testing*. Hasil *testing* setiap *fold* akan disimpan dalam variabel *validation\_labels* dan *validation\_predictions* untuk kemudian dibuat *classification report* dan *confusion matrix* untuk setiap *fold* pada tahap selanjutnya.

#### 4.1.6 Evaluasi Model

Evaluasi model dilakukan dengan mengambil fungsi *confusion matrix* dari *library* Sklearn. Variabel yang menyimpan hasil pelatihan dari setiap *fold* dan keseluruhan *fold* digunakan untuk membuat laporan hasil pelatihan yang telah dilakukan untuk dianalisa kemudian.

Script/kode untuk membuat *confusion matrix* :

```
for fold_num, confusion_mtx in enumerate(all_confusion_matrices, 1):
    if False:
        confusion_mtx = confusion_mtx.astype("float") / confusion_mtx.sum(axis=1)[:,
np.newaxis]

    plt.figure(figsize=(8, 6))
    plt.imshow(confusion_mtx, interpolation="nearest", cmap=plt.cm.OrRd)
    plt.title(f"Confusion Matrix for Fold {fold_num}")
    plt.colorbar()
    tick_marks = np.arange(len(waste_labels.keys()))
    plt.xticks(tick_marks, waste_labels.keys(), rotation=45)
    plt.yticks(tick_marks, waste_labels.keys())
    fmt = ".2f" if False else "d"
    thresh = confusion_mtx.max() / 2.
    for i, j in itertools.product(range(confusion_mtx.shape[0]), range(confusion_mtx.shape[1])):
        plt.text(j, i, format(confusion_mtx[i, j], fmt), horizontalalignment="center",
            color="white" if confusion_mtx[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel("True Labels", fontweight="bold")
    plt.xlabel("Predicted Labels", fontweight="bold")
    plt.savefig(f'/content/1-confusion-matrix-fold-{fold_num}.png', bbox_inches='tight')
    plt.close()
    source_dir = '/content/'
    destination_dir = '/content/drive/My Drive/Skripsi/Evaluation'
```



```

shutil.copy2(f'/content/1-confusion-matrix-fold-{fold_num}.png', destination_dir)

final_confusion_mtx = confusion_matrix(all_validation_labels, all_validation_predictions)

if False:
    final_confusion_mtx = final_confusion_mtx.astype("float") /
    final_confusion_mtx.sum(axis=1)[:, np.newaxis]

plt.figure(figsize=(8, 6))
plt.imshow(final_confusion_mtx, interpolation="nearest", cmap=plt.cm.OrRd)
plt.title(f"Final Confusion Matrix")
plt.colorbar()
tick_marks = np.arange(len(waste_labels.keys()))
plt.xticks(tick_marks, waste_labels.keys(), rotation=45)
plt.yticks(tick_marks, waste_labels.keys())
fmt = ".2f" if False else "d"
thresh = final_confusion_mtx.max() / 2.

for i, j in itertools.product(range(final_confusion_mtx.shape[0]),
                              range(final_confusion_mtx.shape[1])):
    plt.text(j, i, format(final_confusion_mtx[i, j], fmt), horizontalalignment="center",
            color="white" if final_confusion_mtx[i, j] > thresh else "black")
plt.tight_layout()
plt.ylabel("True Labels", fontweight="bold")
plt.xlabel("Predicted Labels", fontweight="bold")
plt.savefig(f'/content/1-confusion-matrix-final.png', bbox_inches='tight')
plt.close()
source_dir = '/content/'
destination_dir = '/content/drive/My Drive/Skripsi/Evaluation'
shutil.copy2(f'/content/1-confusion-matrix-final.png', destination_dir)

```

*Confusion matrix* dari setiap *fold* dioutputkan dengan *looping* dengan menggunakan *figure* dari *library matplotlib*. *Confusion matrix* untuk keseluruhan *fold* juga dibuat menggunakan metode yang sama. Semua *confusion matrix* kemudian disimpan pada Google Drive dengan *index* setiap skenarionya untuk dianalisis kemudian.

#### 4.1.7 Konversi Model

Tahap konversi model dilakukan pada tahap akhir setelah pelatihan dan evaluasi model dilakukan. Model dikonversi menjadi format h5 agar dapat digunakan pada aplikasi *website*. Konversi model menjadi format h5 tidak

memerlukan *library* khusus tetapi hanya menggunakan fungsi *model.save* dengan *directory* yang ditentukan.

Script/kode untuk konversi model :

```
model.save('/content/drive/My Drive/Skripsi/Program/GarbageClassification-1.h5')
```

## 4.2 Analisis dan Perhitungan Performa

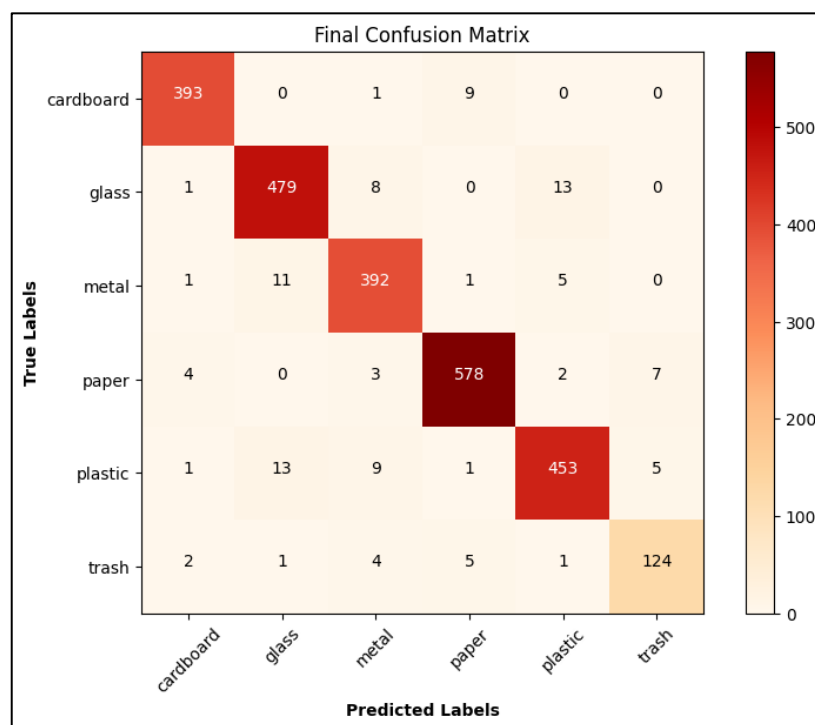
Pada tahapan ini akan dilakukan analisis dan perhitungan performa dari evaluasi keseluruhan skenario pelatihan yang telah dilakukan sebanyak 27 skenario. Analisis dan perhitungan performa dilakukan berdasarkan *Confusion Matrix* yang telah di dapat dari setiap skenario pelatihan. Matriks yang dijadikan pertimbangan adalah *F1 Score* dari hasil *testing* dari pelatihan model untuk mengukur tingkat keakuratan model. Nilai *Precision*, *Recall*, dan *F1 Score* dari hasil *testing* dari skenario 1 hingga 27 dapat dilihat pada Tabel 4.1.

Tabel 4.1 *Precision*, *Recall*, dan *F1 Score* Hasil Skenario

Skenario	Hyperparameters	Precision	Recall	F1 Score
1	ResNet50; Adam; No Dropout	0,9514	0,9509	0,9511
2	ResNet50; Adam; Dropout(0,2)	0,9470	0,9444	0,9456
3	ResNet50; Adam; Dropout(0,5)	0,9233	0,9170	0,9200
4	ResNet50; Nadam; No Dropout	0,9493	0,9468	0,9479
5	ResNet50; Nadam; Dropout(0,2)	0,9334	0,9339	0,9336
6	ResNet50; Nadam; Dropout(0,5)	0,9308	0,9235	0,9270
7	ResNet50; AdamW; No Dropout	0,9436	0,9372	0,9401
8	ResNet50; AdamW; Dropout(0,2)	0,9461	0,9443	0,9452
9	ResNet50; AdamW; Dropout(0,5)	0,9287	0,9204	0,9243
10	VGG19; Adam; No Dropout	0,8858	0,8684	0,8759
11	VGG19; Adam; Dropout(0,2)	0,8756	0,8549	0,8635
12	VGG19; Adam; Dropout(0,5)	0,8496	0,8208	0,8319
13	VGG19; Nadam; No Dropout	0,8681	0,8524	0,8592
14	VGG19; Nadam; Dropout(0,2)	0,8723	0,8505	0,8589
15	VGG19; Nadam; Dropout(0,5)	0,8529	0,8352	0,8427
16	VGG19; AdamW; No Dropout	0,8654	0,8561	0,8604
17	VGG19; AdamW; Dropout(0,2)	0,8787	0,8570	0,8658

Skenario	Hyperparameters	Precision	Recall	F1 Score
18	VGG19; AdamW; Dropout(0,5)	0,8497	0,8335	0,8401
19	DenseNet121; Adam; No Dropout	0,9055	0,9042	0,9048
20	DenseNet121; Adam; Dropout(0,2)	0,9063	0,9066	0,9061
21	DenseNet121; Adam; Dropout(0,5)	0,8797	0,8691	0,8736
22	DenseNet121; Nadam; No Dropout	0,9122	0,9062	0,9090
23	DenseNet121; Nadam; Dropout(0,2)	0,9067	0,8967	0,9011
24	DenseNet121; Nadam; Dropout(0,5)	0,8857	0,8696	0,8761
25	DenseNet121; AdamW; No Dropout	0,9109	0,9006	0,9053
26	DenseNet121; AdamW; Dropout(0,2)	0,9040	0,8906	0,8966
27	DenseNet121; AdamW; Dropout(0,5)	0,8929	0,8874	0,8897



Berdasarkan Tabel 4.1, hasil dengan nilai *F1 Score* yang paling optimal dari proses *testing* adalah skenario 1 dengan *hyperparameter pretrained model* ResNet50, *Optimizer* Adam, dan tidak menggunakan *Dropout Layer* yaitu dengan nilai *F1 Score* sebesar 0,9511. *Confusion Matrix* dari hasil paling optimal dari skenario 1 dapat dilihat pada Gambar 4.2.

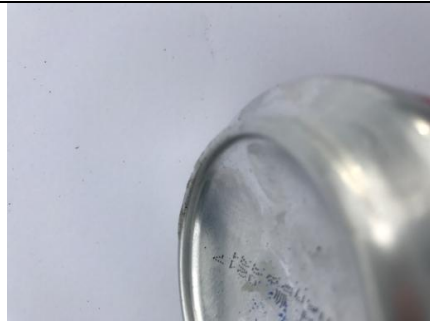


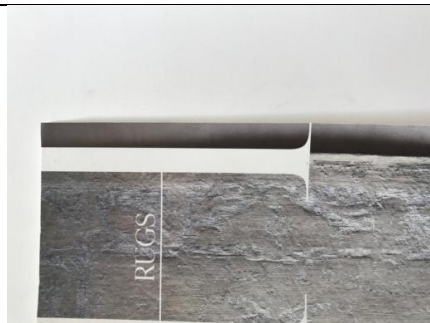



Gambar 4.2 *Confusion Matrix*

Gambar 4.2 merupakan *Confusion Matrix* dari skenario 1 dengan penggunaan perbandingan *hyperparameter pretrained model* ResNet50, *Optimizer* Adam, dan tidak menggunakan *Dropout Layer* yang memberikan *F1 Score* yang terbaik dari semua skenario pelatihan sebesar 0,9511. Pada Gambar 4.2, tugas klasifikasi citra sampah mayoritas sudah akurat untuk setiap kelasnya dibuktikan dengan nilai *True Positive* yang tinggi pada setiap kelasnya. Akan tetapi, terdapat kesalahan klasifikasi pada beberapa kelas seperti klasifikasi citra *Plastic* menjadi *Glass* dan sebaliknya sebanyak masing-masing 13 data, klasifikasi citra *Metal* menjadi *Glass* sebanyak 11 data, klasifikasi citra *Glass* menjadi *Metal* sebanyak 8 data, klasifikasi citra *Plastic* menjadi *Metal* sebanyak 9 data, dan klasifikasi citra *Cardboard* menjadi *Paper* sebanyak 9 data. Tabel 4.2 merupakan sampel dari hasil kesalahan prediksi pada model.

Tabel 4.2 Sampel Kesalahan Klasifikasi

Citra	Actual Class	Predicted Class
	<i>Plastic</i>	<i>Glass</i>
	<i>Glass</i>	<i>Plastic</i>

Citra	<i>Actual Class</i>	<i>Predicted Class</i>
	<i>Glass</i>	<i>Metal</i>
	<i>Metal</i>	<i>Glass</i>
	<i>Metal</i>	<i>Plastic</i>
	<i>Plastic</i>	<i>Metal</i>
	<i>Paper</i>	<i>Cardboard</i>

Citra	Actual Class	Predicted Class
	Cardboard	Paper

Berdasarkan Tabel 4.2, citra pertama dan kedua merupakan kesalahan klasifikasi dari kelas *Plastic* menjadi *Glass* dan *Glass* menjadi *Plastic*. Dapat dilihat pada kedua citra tersebut, terdapat kemiripan dari warna dan dimensi dari objek yang memungkinkan membuat model melakukan kesalahan klasifikasi. Pada citra ketiga merupakan kesalahan klasifikasi dari kelas *Glass* menjadi *Metal*. Hal ini kemungkinan dikarenakan objek pada citra diambil dari sudut pandang atas dari objek sehingga memungkinkan model sulit untuk mengklasifikasikan citra ini sebagai kelas *Glass* dan terdapat kemiripan warna pada objek dengan citra pada kelas *Metal* yaitu warna *silver* berkilau. Pada citra keempat merupakan kesalahan klasifikasi dari kelas *Metal* menjadi *Glass*. Hal ini kemungkinan dikarenakan sudut pandang pada citra diambil hanya dari sebagian kecil dari objek sehingga memungkinkan model sulit mengklasifikasikan citra ini sebagai *Metal*. Pada citra kelima merupakan kesalahan klasifikasi dari kelas *Metal* menjadi *Plastic*. Hal ini kemungkinan dikarenakan sudut pandang citra diambil dari hanya bagian atas objek dan hanya menunjukkan penutup dari objek tersebut yang memiliki warna dominan merah sedangkan pada mayoritas citra pada kelas *Metal* memiliki warna *silver* berkilau. Pada citra keenam merupakan kesalahan klasifikasi dari kelas *Plastic* menjadi *Metal*. Hal ini kemungkinan dikarenakan *label* dari objek lebih dominan

pada objek tersebut dan berwarna *silver* berkilau sehingga memungkinkan model untuk mengklasifikasikan citra tersebut sebagai kelas *Metal*. Pada citra ketujuh merupakan kesalahan klasifikasi dari kelas *Paper* menjadi *Cardboard*. Hal ini kemungkinan dikarenakan tekstur dari objek yang kasar sehingga memungkinkan model untuk mengklasifikasikan citra sebagai kelas *Cardboard* yang mayoritas objek pada kelas ini memiliki tekstur objek yang kasar. Pada citra terakhir merupakan kesalahan klasifikasi dari kelas *Cardboard* menjadi *Paper*. Hal ini kemungkinan dikarenakan objek memiliki *label* dengan konten yang cukup banyak sehingga memungkinkan model untuk mengklasifikasi citra sebagai kelas *Paper* yang mayoritas objek pada kelas ini memiliki konten yang cukup banyak.

Berdasarkan *Confusion Matrix* pada Gambar 4.2, nilai *True Positive*, *False Positive*, dan *False Negative* dari setiap kelas untuk digunakan pada perhitungan nilai *Precision*, *Recall* dan *F1 Score* dapat dilihat dari *Confusion Matrix*. Berdasarkan hasil tersebut, dapat dilakukan perhitungan untuk nilai *Precision* dari hasil skenario 1 menggunakan Rumus 2.1 pada sub bab *Confusion Matrix*.

Tabel 4.3 Perhitungan Nilai *Precision*

<i>Class</i>	<b>Perhitungan</b>
<i>Cardboard</i>	$\frac{393}{393 + (1 + 1 + 4 + 1 + 2)} = 0,9776$
<i>Glass</i>	$\frac{479}{479 + (11 + 13 + 1)} = 0,9504$
<i>Metal</i>	$\frac{392}{392 + (1 + 8 + 3 + 9 + 4)} = 0,9400$
<i>Paper</i>	$\frac{578}{578 + (9 + 1 + 1 + 5)} = 0,9731$
<i>Plastic</i>	$\frac{453}{453 + (13 + 5 + 2 + 1)} = 0,9557$

<i>Class</i>	<b>Perhitungan</b>
<i>Trash</i>	$\frac{124}{124 + (7 + 5)} = 0,9118$
	$\frac{0,9776 + 0,9504 + 0,9400 + 0,9731 + 0,9557 + 0,9118}{6} = 0,9514$

Berdasarkan Tabel 4.3, performa model untuk mengklasifikasikan menjadi kelas-kelas yang ada dapat diukur berdasarkan nilai *Precision* dari setiap kelas. Performa model untuk melakukan klasifikasi citra dengan kelas *Cardboard* sebagai kelas *Cardboard* diukur dari nilai *Precision* untuk kelas *Cardboard* yang bernilai sebesar 0,9776, klasifikasi citra dengan kelas *Glass* sebagai kelas *Glass* diukur dari nilai *Precision* untuk kelas *Glass* yang bernilai sebesar 0,9504, klasifikasi citra dengan kelas *Metal* sebagai kelas *Metal* diukur dari nilai *Precision* untuk kelas *Metal* yang bernilai sebesar 0,9400, klasifikasi citra dengan kelas *Paper* sebagai kelas *Paper* diukur dari nilai *Precision* untuk kelas *Paper* yang bernilai sebesar 0,9731, klasifikasi citra dengan kelas *Plastic* sebagai kelas *Plastic* diukur dari nilai *Precision* untuk kelas *Plastic* yang bernilai sebesar 0,9557, dan klasifikasi citra dengan kelas *Trash* sebagai kelas *Trash* diukur dari nilai *Precision* untuk kelas *Trash* yang bernilai sebesar 0,9118.

Berdasarkan nilai *Precision* model secara keseluruhan yang diambil dengan cara menghitung rata-rata dari setiap nilai *Precision* dari setiap kelas, model dapat mengklasifikasikan citra dengan kelas tertentu sebagai kelas yang sesuai dengan kelas aktualnya yang dibuktikan dengan nilai *Precision* sebesar 0,9514. Hal ini menunjukkan bahwa skenario pelatihan dengan penggunaan perbandingan *hyperparameter pretrained model* ResNet50, *Optimizer* Adam, dan tidak



menggunakan *Dropout Layer* dapat melakukan tugas untuk mengklasifikasikan citra dengan kelas tertentu sebagai kelas yang sesuai dengan baik yang dibuktikan dengan nilai *Precision* yang tinggi.

Berdasarkan nilai *True Positive*, *False Positive*, dan *False Negative* yang didapatkan dari *Confusion Matrix* pada Gambar 4.2, dapat dilakukan perhitungan untuk nilai *Recall* dari hasil skenario 1 menggunakan Rumus 2.2 pada sub bab *Confusion Matrix*.

Tabel 4.4 Perhitungan Nilai *Recall*

Class	Perhitungan
Cardboard	$\frac{393}{393 + (1 + 9)} = 0,9752$
Glass	$\frac{479}{479 + (1 + 8 + 13)} = 0,9561$
Metal	$\frac{392}{392 + (1 + 11 + 1 + 5)} = 0,9561$
Paper	$\frac{578}{578 + (4 + 3 + 2 + 7)} = 0,9731$
Plastic	$\frac{453}{453 + (1 + 13 + 9 + 1 + 5)} = 0,9398$
Trash	$\frac{124}{124 + (2 + 1 + 4 + 5 + 1)} = 0,9051$
	$\frac{0,9752 + 0,9561 + 0,9561 + 0,9731 + 0,9398 + 0,9051}{6} = 0,9509$

Berdasarkan Tabel 4.4, performa model untuk mendeteksi citra dengan sebagai suatu kelas dibandingkan dengan jumlah aktual dari citra dengan kelas tersebut dapat diukur berdasarkan nilai *Recall* dari setiap kelas. Performa model untuk mendeteksi citra dengan kelas *Cardboard* dari jumlah citra aktual pada kelas *Cardboard* diukur dari nilai *Recall* untuk kelas *Cardboard* yang bernilai sebesar 0,9752, deteksi citra dengan kelas *Glass* sebagai kelas *Glass* diukur dari nilai *Recall*

untuk kelas *Glass* yang bernilai sebesar 0,9504, deteksi citra dengan kelas *Metal* sebagai kelas *Metal* diukur dari nilai *Recall* untuk kelas *Metal* yang bernilai sebesar 0,9400, deteksi citra dengan kelas *Paper* sebagai kelas *Paper* diukur dari nilai *Recall* untuk kelas *Paper* yang bernilai sebesar 0,9731, deteksi citra dengan kelas *Plastic* sebagai kelas *Plastic* diukur dari nilai *Recall* untuk kelas *Plastic* yang bernilai sebesar 0,9557, dan deteksi citra dengan kelas *Trash* sebagai kelas *Trash* diukur dari nilai *Recall* untuk kelas *Trash* yang bernilai sebesar 0,9118.

Berdasarkan nilai *Recall* model secara keseluruhan yang diambil dengan cara menghitung rata-rata dari setiap nilai *Recall* dari setiap kelas, model dapat mendeteksi citra sebagai kelas tertentu dengan nilai *Recall* sebesar 0,9509. Hal ini menunjukkan bahwa skenario pelatihan dengan penggunaan perbandingan *hyperparameter pretrained model* ResNet50, *Optimizer* Adam, dan tidak menggunakan *Dropout Layer* dapat melakukan tugas untuk mendeteksi citra sebagai kelas tertentu dengan baik dan memiliki sensitivitas yang cukup baik yang dibuktikan dengan nilai *Recall* yang tinggi.

Berdasarkan nilai *True Positive*, *False Positive*, dan *False Negative* yang didapatkan pada *Confusion Matrix* pada Gambar 4.2, dapat dilakukan perhitungan untuk nilai *F1 Score* dari hasil skenario 1 menggunakan Rumus 2.3 pada sub bab *Confusion Matrix*.

Tabel 4.5 Perhitungan Nilai *F1 Score*

<i>Class</i>	<b>Perhitungan</b>
<i>Cardboard</i>	$2 \times \frac{0,9776 \times 0,9752}{0,9776 + 0,9752} = 0,9764$
<i>Glass</i>	$2 \times \frac{0,9504 \times 0,9561}{0,9504 + 0,9561} = 0,9532$

<i>Class</i>	<b>Perhitungan</b>
<i>Metal</i>	$2 \times \frac{0,9400 \times 0,9561}{0,9400 + 0,9561} = 0,9480$
<i>Paper</i>	$2 \times \frac{0,9731 \times 0,9731}{0,9731 + 0,9731} = 0,9731$
<i>Plastic</i>	$2 \times \frac{0,9557 \times 0,9398}{0,9557 + 0,9398} = 0,9477$
<i>Trash</i>	$2 \times \frac{0,9118 \times 0,9051}{0,9118 + 0,9051} = 0,9084$
	$\frac{0,9764 + 0,9532 + 0,9480 + 0,9731 + 0,9477 + 0,9084}{6} = 0,9511$

Berdasarkan Tabel 4.5, dapat dilihat nilai *F1 Score* dari setiap kelas yang masing-masing memiliki nilai yang cukup tinggi yang mengindikasikan model memiliki keseimbangan yang baik antara kemampuan model untuk mengklasifikasikan citra dengan kelas tertentu yang sesuai dengan kelas aktualnya dan kemampuan model untuk mendeteksi citra dengan kelas tertentu dari jumlah aktual untuk setiap kelasnya. Hal ini menunjukkan bahwa model dapat melaksanakan tugas untuk mengklasifikasikan citra sampah menjadi kelas *Cardboard*, *Glass*, *Metal*, *Paper*, *Plastic*, dan *Trash* dengan baik.

Berdasarkan evaluasi dari model pelatihan sebanyak 27 skenario, analisis yang didapat berdasarkan *hyperparameter* yang dibandingkan adalah sebagai berikut:

1. Penggunaan arsitektur *pretrained model* yang berbeda sangat berpengaruh pada performa sebuah model. Dapat dilihat *F1 Score* dari skenario pelatihan yang menggunakan arsitektur DenseNet121 dengan koneksi antar *layer*, tidak lebih baik dibandingkan dengan skenario pelatihan yang menggunakan arsitektur ResNet50 dengan *shortcut connection*. Di sisi lain, skenario

pelatihan yang menggunakan arsitektur VGG19 tanpa ada koneksi antar *layer* menunjukkan *F1 Score* sebesar 0,8759 yang lebih kecil dibandingkan kedua arsitektur tersebut. Hal ini menunjukkan model dengan arsitektur DenseNet121 dengan koneksi antar *layer* yang lebih padat mengalami kesulitan dalam generalisasi saat melakukan klasifikasi citra pada *dataset* Garbage Classification. Di sisi lain, model dengan arsitektur tradisional seperti VGG19 sudah jelas menghasilkan performa yang tidak lebih baik dibandingkan dengan kedua arsitektur yang lebih kompleks dalam melakukan klasifikasi citra pada *dataset* Garbage Classification.

2. Penggunaan *Optimizer* cukup berpengaruh pada *F1 Score* dari pelatihan sebuah model. Penggunaan *Optimizer* Adam menunjukkan performa yang lebih baik pada arsitektur Resnet50 dan VGG19, sedangkan Nadam menunjukkan performa yang lebih baik pada arsitektur DenseNet121. Hal ini menunjukkan bahwa *Optimizer* yang lebih kompleks seperti Nadam dengan penggunaan *Nesterov's Accelerated Gradient* dan AdamW dengan penggunaan *Weight Decay* tidak selalu menghasilkan performa yang lebih baik dibandingkan *Optimizer* yang lebih sederhana seperti Adam yang hanya memanfaatkan momentum order rendah dalam melakukan klasifikasi citra pada *dataset* Garbage Classification.
3. Penggunaan *Dropout layer* cukup berpengaruh pada *F1 Score* dari pelatihan sebuah model. Penggunaan *Dropout layer* sebesar 0,2 mayoritas menghasilkan performa yang tidak lebih baik dibandingkan tidak menggunakan *Dropout layer*. Di sisi lain, penggunaan *Dropout layer* sebesar

0,5 sudah jelas tidak menghasilkan performa yang lebih baik. Hal ini menunjukkan untuk *dataset* Garbage Classification, tidak memerlukan pembuangan *node* pada *layer* untuk mendapatkan hasil *F1 Score* yang optimal. Hal ini juga menunjukkan *feature map* dari data pada *dataset* ini mudah diekstrak sehingga tidak memerlukan *Dropout layer* untuk membuang *node* yang tidak efektif.

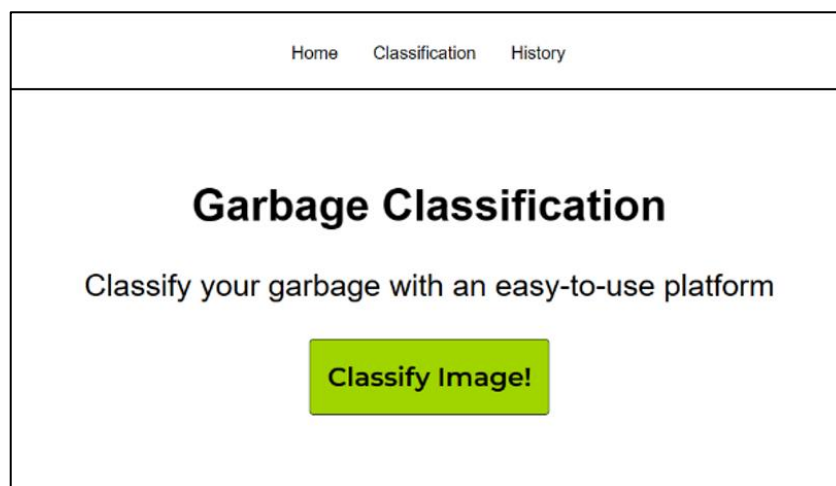
4. Kombinasi arsitektur ResNet50 dengan *Optimizer* Adam dan tidak menggunakan *Dropout layer* lebih cocok dalam klasifikasi citra pada *dataset* Garbage Classification dibuktikan memberikan performa yang lebih optimal dibandingkan dengan skenario lainnya. Penggunaan arsitektur dengan *layer* tidak terlalu banyak dengan koneksi yang sederhana serta penggunaan *optimization algorithm* seperti Adam cukup untuk tugas klasifikasi citra dengan karakteristik citra seperti pada *dataset* Garbage Classification. Terlepas dari *hyperparameter* arsitektur model, *Optimizer*, dan *Dropout layer*, performa model juga memiliki ketergantungan dengan *hyperparameter* lain yang digunakan pada model.

### 4.3 Implementasi Pembuatan Aplikasi

Pada tahapan ini dilakukan pembuatan aplikasi berbasis *website* untuk melakukan tugas klasifikasi citra sampah dengan menggunakan *framework* Laravel. Pada tahapan ini dijelaskan implementasi pembuatan aplikasi yang terdiri dari 4 halaman, yaitu *landing page*, *classification page*, *prediction page*, dan *history page* serta penjelasan integrasi antara model *deep learning* dan aplikasi dengan menggunakan Flask Application.

#### 4.3.1 *Landing Page*

*Landing page* merupakan halaman utama saat pengguna mengakses aplikasi. Pada halaman ini terdapat judul aplikasi dan *tagline* serta navigasi untuk akses menuju ke halaman *classification page* dan *history page*.



Gambar 4.3 Tampilan *Landing Page*

Pada Gambar 4.3, terdapat *navigation bar* untuk mengakses *classification page* dengan tombol bertuliskan dengan tombol *Classification*, mengakses *history page* dengan tombol bertuliskan *History*, dan mengakses *landing page* kembali dengan tombol bertuliskan *Home*. Pada halaman ini, terdapat juga judul dari aplikasi beserta *tag line* dari aplikasi ini dan tombol untuk mengakses *classification page* dibawah judul dan *tag line* aplikasi.

#### 4.3.2 *Classification Page*

*Classification page* merupakan halaman untuk melakukan klasifikasi citra sampah dengan *form input* berbentuk *file* untuk *upload* citra yang ingin diklasifikasi serta tombol untuk *submit form*.

Gambar 4.4 Tampilan *Classification Page*

Pada Gambar 4.4, terdapat penjelasan untuk klasifikasi citra sampah menjadi kelas apa saja yang tersedia pada prediksi citra. Pada halaman ini terdapat *form input file* yang menerima *file image*. Setelah gambar berhasil diupload, maka untuk mensubmit gambar dapat dilakukan dengan menekan tombol *Classify Image*.

Script/kode *form input* gambar :

```
<form method="POST" id="form" action="{{ route('make-prediction') }}"
  enctype="multipart/form-data" class="text-center align-items-center justify-content-center">
  @csrf
  <div class="text-center justify-content-center align-items-center mb-4">
    <input type="text" name="text1" id="text1" style="color:black; display:none;
border:1px solid black" class="ml-20">
    <input type="text" name="text2" id="text2" style="color:black; display:none;
border:1px solid black" class="ml-20">
    <input type="text" name="text3" id="text3" style="color:black; display:none;
border:1px solid black" class="ml-20">
    <input type="file" name="image" id="image" accept="image/*" style="color:black;"
class="ml-20" required>
  </div>
</form>

<div class="text-center align-items-center justify-content-center">
  <button id="predictBtn" class=" btn bg-[#97D729] w-max p-3 text-md sm:text-xl
md:text-2xl lg:text-2xl xl:text-2xl 2xl:text-2xl fw-bold btn-outline-dark hover:bg-[#97D729]"
style="color:black; font-family: 'Montserrat', sans-serif; transition: opacity 0.2s ease;"
onmouseover="this.style.opacity='0.8'" onmouseout="this.style.opacity='1'">Classify
Image</button>
</div>
```

*Form* menggunakan metode *POST* dengan elemen *input* pada *form* hanya menerima satu *input* berupa satu *file image*. Gambar yang *disubmit* pada *form* kemudian dilakukan *handling* dengan melakukan *fetch* kepada *endpoint* API dari Flask pada bagian *JavaScript*.

Script/kode pada *JavaScript Classification Page* :

```
<script>
document.getElementById('predictBtn').addEventListener('click', function() {
  var input = document.getElementById('image');
  var imageFile = input.files[0];

  var formData = new FormData();
  formData.append('image', imageFile);

  fetch('https://143.198.204.170:5000/predict', {
    method: 'POST',
    body: formData
  })
  .then(response => response.json())
  .then(data => {
    var form = document.getElementById('form');
    var responseInput = document.createElement('input');
    responseInput.type = 'hidden';
    responseInput.name = 'responseData';
    responseInput.value = JSON.stringify(data);
    form.appendChild(responseInput);

    document.getElementById('text1').value = data.Class;
    document.getElementById('text2').value = data.Prediction;
    document.getElementById('text3').value = data.ProbabilityPercentage;

    form.submit();
  })
  .catch(error => {
    console.error('Error:', error);
  });
});
</script>
```

Elemen *button* setelah *form input* dijadikan sebagai tombol *submit*. *Input* citra akan didapatkan pada fungsi ini. Apabila citra terdeteksi, maka akan dilakukan fungsi *fetch* kepada *endpoint* API dari Flask dengan metode *POST* dengan data citra



dimasukkan ke dalam *FormData*. Setelah itu, hasil prediksi beserta probabilitas akan didapatkan dan akan disimpan ke dalam elemen *input hidden* pada *form* dan kemudian *form* *submit* untuk mengirimkan hasil prediksi beserta probabilitasnya kepada *controller*.

Script/kode pada *ModelTestController* :

```
public function makePrediction(Request $request)
{
    $request->validate([
        'image' => 'required|image|mimes:jpeg,png,jpg,jfif|max:2048',
    ]);

    $predictions = $request->text1;
    $class = $request->text2;
    $prob_percentage = $request->text3;

    $imagePath = $request->file('image')->store('public/images');
    $relativeImagePath = str_replace('public/', '', $imagePath);

    $image = new Image();
    $image->path = $relativeImagePath;
    $image->prediction = $predictions;
    if ($predictions !== 'none') {
        $image->probability = $prob_percentage;
    } else if ($predictions === 'none') {
        $image->probability = 'none';
    }
    $image->save();

    return view('predict', [
        'predictions' => $predictions,
        'class' => $class,
        'prob_percentage' => $prob_percentage,
        'imagePath' => $imagePath
    ]);
}
```

*Request* yang dikirim oleh *form* berisi citra, hasil prediksi, dan persentase probabilitas akan disimpan dalam variabel pada *controller*. Citra akan disimpan dalam folder *public* dan direcord dalam *database* beserta hasil prediksi dan persentase probabilitas hasil prediksinya. Data tersebut akan dikembalikan pada *prediction page* untuk ditampilkan.

#### 4.3.3 Prediction Page

*Prediction page* merupakan halaman yang berisi citra yang diupload pada *form* sebelumnya beserta hasil prediksi kelas untuk citra tersebut.





Gambar 4.5 Tampilan *Prediction Page*

Pada Gambar 4.5, dapat dilihat citra yang diupload pengguna adalah citra dari sampah plastik dan model memprediksi citra tersebut sebagai kelas *Plastic* beserta dengan probabilitas prediksinya sebesar 99.43 persen. Apabila pengguna ingin mengganti gambar dan melakukan klasifikasi citra lain, maka pengguna dapat menekan tombol *Change Image* dan akan diarahkan kembali menuju halaman *Classification Page*.

#### 4.3.4 History Page

*History page* merupakan halaman yang berisi riwayat dari klasifikasi citra sampah yang sebelumnya telah dilakukan pada aplikasi.

Classification History				
No	Image	Prediction	Probability	Timestamp
1		Metal	99.58%	2023-11-15 19:25:08
2		Plastic	99.43%	2023-11-15 19:25:18

Gambar 4.6 Tampilan *History Page*

Pada Gambar 4.6, dapat dilihat sebuah tabel yang berisi nomor, gambar, hasil prediksi dari gambar, dan *timestamp* yang berisi tanggal serta jam dari prediksi citra. Pengguna dapat melihat citra apa saja yang telah diklasifikasi pada aplikasi ini beserta dengan kelas apa yang berhasil diprediksi model untuk citra tersebut.

#### 4.3.5 Flask Application

Pada integrasi antara model *Deep Learning* dan aplikasi *website*, Flask Application digunakan untuk mengambil model yang sudah dikonversi menjadi *format .h5* dan memasukkan gambar yang telah *disubmit* dari *form input* kedalam model untuk diprediksi. Model akan mengeluarkan hasil prediksi dari gambar yang *diinput*.

Script/kode pada Flask Application :

```
app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
model = load_model('GarbageClassification-1.h5')
```

```

@app.route('/predict', methods=['POST'])
@cross_origin
def predict():
    try:
        image_file = request.files['image']

        if 'image' not in request.files:
            return jsonify({'error': 'No image in the request'}), 400

        temp_image = tempfile.NamedTemporaryFile(delete=False)
        image_file.save(temp_image)
        temp_image.close()

        img = load_img(temp_image.name, target_size=(
            224, 224))
        img_array = img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)

        class_probabilities = model.predict(img_array)
        waste_labels = {0: "cardboard", 1: "glass",
            2: "metal", 3: "paper", 4: "plastic", 5: "trash"}

        response = ""

        predicted_class_index = np.argmax(class_probabilities)
        predicted_class_label = (
            waste_labels[predicted_class_index]).capitalize()
        predicted_class_probability = float((class_probabilities[0,
            predicted_class_index]))
        predicted_class_probability_percentage = "{:.2f}%".format(
            predicted_class_probability * 100)

        response = {'Prediction': predicted_class_label,
            'Class': predicted_class_label,
            'ProbabilityPercentage': predicted_class_probability_percentage}

        print(response)
        return jsonify(response)

    except Exception as e:
        print('An error occurred:', str(e))
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True, ssl_context='adhoc', host='0.0.0.0', port=5000)

```

Flask Application dibuat dengan menginisiasi terlebih dahulu Flask *web application* dengan mengambil dari *library* Flask. Model *Deep Learning* yang sudah dikonversi menjadi *format* h5 dimuat dan disimpan dalam variabel model.

*Endpoint* API dibuat dengan method *POST* dengan *error handling* untuk dipanggil pada *JavaScript*. Konfigurasi *host* dan *port* juga dilakukan untuk kebutuhan pemanggilan API pada *JavaScript*.

Gambar yang masuk pada *POST request* dapat dimuat pada variabel dan dilakukan *error handling* untuk mencegah *error* pada fungsi. Setelah gambar berhasil dimuat, gambar diduplikat pada variabel sementara untuk dilakukan *resize* sesuai dengan *requirement input* pada model yaitu 224x224 dan kemudian gambar dibuat menjadi bentuk *numpy array*. Gambar kemudian akan diprediksi oleh model yang akan menghasilkan probabilitas dari setiap kelas dalam bentuk angka. Hasil prediksi, probabilitas, beserta gambarnya akan dikembalikan pada *JavaScript*.

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Berdasarkan berbagai hal yang telah disampaikan pada bab sebelumnya serta pembuatan dan implementasi sistem klasifikasi citra sampah yang sudah dilakukan penulis, maka dapat diambil simpulan sebagai berikut:

5. Penerapan model *Convolutional Neural Network* untuk klasifikasi citra sampah dapat dilakukan dengan menggunakan *framework* TensorFlow dan *library* Keras dengan bahasa pemrograman Python. *Dataset* yang digunakan diambil melalui *platform* Kaggle dengan nama Garbage Classification yang berisikan data dengan 6 kelas yaitu *Cardboard*, *Glass*, *Metal*, *Plastic*, *Paper*, dan *Trash*.
6. Cara menemukan konfigurasi *hyperparameter* pada model *Convolutional Neural Network* untuk klasifikasi citra sampah dilakukan dengan percobaan dengan mengkombinasikan *hyperparameter* yang difokuskan pada tiga *hyperparameter* yaitu penggunaan arsitektur *Pretrained Model*, penggunaan *Optimizer*, dan penggunaan *Dropout layer* dengan masing-masing *hyperparameter* sejumlah tiga skenario dengan total skenario pelatihan sebanyak 27 skenario. Konfigurasi *hyperparameter* model *Convolutional Neural Network* yang menghasilkan performa paling optimal yaitu menggunakan arsitektur ResNet50, menggunakan *Optimizer* Adam, dan tidak menggunakan *Dropout layer* yang menghasilkan *F1 Score* sebesar 0,9511.

7. Model yang memperoleh hasil *F1 Score* yang paling optimal adalah skenario pelatihan dengan menggunakan arsitektur ResNet50, menggunakan *Optimizer* Adam, dan tidak menggunakan *Dropout layer* yang menghasilkan *F1 Score* sebesar 0,9511 dengan nilai *Precision* sebesar 0,9514 dan nilai *Recall* sebesar 0,9509. Nilai tersebut menunjukkan model memiliki keseimbangan antara kemampuan model untuk mendeteksi dan melakukan klasifikasi citra menjadi kelas *Cardboard*, *Glass*, *Metal*, *Plastic*, *Paper*, atau *Trash*.
8. Penerapan model *Convolutional Neural Network* pada aplikasi klasifikasi citra sampah berbasis *website* dapat dilakukan dengan menggunakan Flask Application untuk melakukan prediksi pada citra yang diinputkan pada aplikasi. Model *Convolutional Neural Network* yang menghasilkan performa paling optimal dikonversi menjadi format h5 untuk digunakan pada Flask Application dan dapat menerima *input* berupa citra untuk dilakukan prediksi.

## 5.2 Saran

Berdasarkan penelitian yang dilakukan, penulis memberikan saran yang dapat diimplementasikan pada penelitian selanjutnya, yaitu sebagai berikut:

1. *Dataset* yang digunakan memiliki jumlah data yang *imbalance* untuk kelas *Trash* hanya sebanyak 137 citra. Oleh karena itu, diperlukan jumlah data yang seimbang untuk setiap kelasnya sehingga model *Convolutional Neural Network* dapat menghasilkan performa yang lebih optimal.
2. Diperlukan percobaan implementasi model dengan metode seperti *oversampling* untuk mengatasi kelas yang *imbalance*.

3. Diperlukan perangkat keras yang memiliki spesifikasi lebih baik untuk mempercepat proses *training* model sehingga penelitian dapat berjalan dengan lebih efektif.



## DAFTAR PUSTAKA

- Booch, G., Rumbaugh, J., & Jacobson, I. (1998). *The Unified Modeling Language User Guide* (1st ed.). Addison Wesley Longman, Inc.
- Bradski, G., & Kaehler, A. (2008). *Learning OpenCV* (1st ed.). O'Reilly Media, Inc.
- Direktorat Penanganan Sampah. (2022). *Sistem Informasi Pengelolaan Sampah Nasional*. <https://sipsn.menlhk.go.id/sipsn/>
- Dozat, T. (2016). Incorporating Nesterov Momentum into Adam. *4th International Conference on Learning Representations*.
- Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media, Inc.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>
- Google Research. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *ArXiv*.
- Grinberg, M. (2018). *Flask Web Development* (2nd ed.). O'Reilly Media, Inc.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Huang, G., Liu, Z., Maaten, L. van der, & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Joshi, P. (2017). *Artificial Intelligence with Python*. Packt Publishing Ltd.
- Kaza, S., Yao, L. C., Bhada-Tata, P., & Van Woerden, F. (2018). *What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050*. World Bank. <https://doi.org/10.1596/978-1-4648-1329-0>
- Kingma, D. P., & Ba, J. (2014). Adam : A Method for Stochastic Optimization. *3rd International Conference on Learning Representations*.
- Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. *7th International Conference on Learning Representations*.
- Ozdemir, S. (2016). *Principles of Data Science*. Packt Publishing Ltd.
- Pemerintah Republik Indonesia. (2008). *Undang-undang (UU) Nomor 18 Tahun 2008 tentang Pengelolaan Sampah*. Pemerintah Pusat.

- Rossum, G. van, & Drake, F. L. (2003). *An Introduction to Python: Vol. 2.2.2*. Network Theory Limited.
- Sammut, C., & Webb, G. I. (2017). *Encyclopedia of Machine Learning and Data Mining*. Springer Nature.
- Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *3rd International Conference on Learning Representations*.
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press. <https://d2l.ai/>

## LAMPIRAN

### Lampiran 1 Kode Implementasi Pembuatan Model

#### *Garbage Classification Model :*

```
# Import Library
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import cv2
import urllib
import itertools
import random, os, glob
import zipfile
import shutil
import h5py
from imutils import paths
from sklearn.utils import shuffle
from urllib.request import urlopen

import warnings
warnings.filterwarnings("ignore")

from sklearn.metrics import confusion_matrix,
classification_report
from sklearn.model_selection import KFold
import tensorflow as tf
from keras.models import load_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing import image
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.layers import Conv2D, Flatten,
MaxPooling2D, Dense, Dropout, SpatialDropout2D,
BatchNormalization, GlobalAveragePooling2D, Lambda
from tensorflow.keras.preprocessing.image import
ImageDataGenerator, img_to_array, load_img, array_to_img

# Mounting Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Load Dataset Path
dir_path = "/content/drive/My Drive/Skripsi/Program/Garbage
classification/Garbage classification"

print(os.listdir(dir_path))

# Load Dataset
target_size = (224, 224)
```

```

waste_labels = {"cardboard":0, "glass":1, "metal":2, "paper":3,
"plastic":4, "trash":5}

def load_dataset(path, target_size):
    x = []
    labels = []
    image_paths = sorted(list(paths.list_images(path)))
    for image_path in image_paths:
        img = cv2.imread(image_path)
        img = cv2.resize(img, target_size)
        x.append(img)
        label = image_path.split(os.path.sep)[-2]
        labels.append(waste_labels[label])
    x, labels = shuffle(x, labels, random_state=42)
    input_shape = (target_size[0], target_size[1], 3)
    print("X shape:", np.array(x).shape)
    print("Total Class:", len(np.unique(labels)))
    print("Total Data:", len(labels))
    print("Input Shape:", input_shape)
    return x, labels, input_shape

x, labels, input_shape = load_dataset(dir_path, target_size)

# Visualize Dataset
def visualize_img(image_batch, labels, num_of_img):
    plt.figure(figsize=(10, 10))
    for n in range(num_of_img):
        ax = plt.subplot(5, 5, n+1)
        plt.imshow(image_batch[n])

plt.title(np.array(list(waste_labels.keys()))[to_categorical(labels, num_classes=len(np.unique(labels)))[n] == 1][0].title())
plt.axis("off")

visualize_img(x, labels, 10)

# Data Preprocessing
def CNN_data_preparation():
    train = ImageDataGenerator(

# ResNet : tf.keras.applications.resnet50.preprocess_input
# VGG : tf.keras.applications.vgg19.preprocess_input
# DenseNet : tf.keras.applications.densenet.preprocess_input
preprocessing_function=tf.keras.applications.resnet50.preprocess_input,
        horizontal_flip=True,
        vertical_flip=True,
        rotation_range=20,
        shear_range=0.2,
        zoom_range=0.2,
        width_shift_range=0.1,
        height_shift_range=0.1,
    )
    train_generator = train.flow_from_directory(
        directory=dir_path,
        target_size=(target_size),

```

```

        batch_size=32,
        class_mode="categorical",
        subset="training"
    )

    return train_generator

train_generator = CNN_data_preparation()

# K-Fold Cross Validation
n_splits = 5

train_metrics = []
val_metrics = []
all_classification_reports = []
all_confusion_matrices = []
all_validation_labels = []
all_validation_predictions = []

kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)

# Initiate Model
img_shape = (224, 224, 3)

# ResNet : tf.keras.applications.ResNet50
# VGG : tf.keras.applications.VGG19
# DenseNet : tf.keras.applications.DenseNet121
base_model =
tf.keras.applications.ResNet50(input_shape=img_shape,
                               include_top=False,
                               weights='imagenet')

for layer in base_model.layers:
    layer.trainable = False

# base_model.summary()

# Model Training
def generate_data(data_generator, indices):
    while True:
        for i in indices:
            batch_data, batch_labels = data_generator[i]
            yield batch_data, batch_labels

model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(units=256, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(units=6, activation='softmax'))
model.compile(optimizer=Adam(learning_rate=1e-4),
              loss='categorical_crossentropy', metrics=['accuracy'])

for train_indices, val_indices in kf.split(train_generator):

```

```

    train_data_generator = generate_data(train_generator,
train_indices)
    val_data_generator = generate_data(train_generator,
val_indices)

    print('Training Per Fold')

    history = model.fit(train_data_generator,
                        validation_data=val_data_generator,
                        epochs=10,
                        verbose=2,
                        steps_per_epoch=len(train_indices),
                        validation_steps=len(val_indices),
                        )

    train_metrics.append(history.history['accuracy'])
    val_metrics.append(history.history['val_accuracy'])

    validation_labels = []
    validation_predictions = []
    for _ in range(len(val_indices)):
        batch_data, batch_labels = next(val_data_generator)
        validation_labels.extend(np.argmax(batch_labels,
axis=1))

validation_predictions.extend(np.argmax(model.predict(batch_data
), axis=1))

    all_validation_labels.extend(validation_labels)
    all_validation_predictions.extend(validation_predictions)

    classification_report_str =
classification_report(validation_labels, validation_predictions,
target_names=["cardboard", "glass", "metal", "paper", "plastic",
"trash"])
    all_classification_reports.append(classification_report_str)

    confusion_mtx = confusion_matrix(validation_labels,
validation_predictions)
    all_confusion_matrices.append(confusion_mtx)

    print('\n')

# Classification Report
for fold_num, classification_report_str in
enumerate(all_classification_reports, 1):
    print(f"Classification Report for Fold {fold_num}\n")
    print(classification_report_str, '\n')

    plt.figure(figsize=(8, 6))
    plt.text(0.2, 0.7, classification_report_str, fontsize=12,
verticalalignment='center')
    plt.axis('off')
    plt.tight_layout()
    plt.savefig(f'/content/1-class-report-fold-{fold_num}.png',
bbox_inches='tight')

```

```

plt.close()

source_dir = '/content/'
destination_dir = '/content/drive/My
Drive/Skripsi/Evaluation'
shutil.copy2(f'/content/1-class-report-fold-{fold_num}.png',
destination_dir)

final_classification_report_str =
classification_report(all_validation_labels,
all_validation_predictions, target_names=["cardboard", "glass",
"metal", "paper", "plastic", "trash"])
print(f"Final Classification Report\n")
print(final_classification_report_str)

plt.figure(figsize=(8, 6))
plt.text(0.3, 0.0, final_classification_report_str, fontsize=12,
verticalalignment='center')
plt.axis('off')
plt.tight_layout()
plt.savefig('/content/1-class-report-final.png',
bbox_inches='tight')
plt.close()
source_dir = '/content/'
destination_dir = '/content/drive/My Drive/Skripsi/Evaluation'
shutil.copy2(source_dir + '1-class-report-final.png',
destination_dir)

# Confusion Matrix
for fold_num, confusion_mtx in enumerate(all_confusion_matrices,
1):
    if False:
        confusion_mtx = confusion_mtx.astype("float") /
confusion_mtx.sum(axis=1)[:, np.newaxis]

    plt.figure(figsize=(8, 6))
    plt.imshow(confusion_mtx, interpolation="nearest",
cmap=plt.cm.OrRd)
    plt.title(f"Confusion Matrix for Fold {fold_num}")
    plt.colorbar()
    tick_marks = np.arange(len(waste_labels.keys()))
    plt.xticks(tick_marks, waste_labels.keys(), rotation=45)
    plt.yticks(tick_marks, waste_labels.keys())
    fmt = ".2f" if False else "d"
    thresh = confusion_mtx.max() / 2.
    for i, j in itertools.product(range(confusion_mtx.shape[0]),
range(confusion_mtx.shape[1])):
        plt.text(j, i, format(confusion_mtx[i, j], fmt),
horizontalalignment="center",
color="white" if confusion_mtx[i, j] > thresh
else "black")
    plt.tight_layout()
    plt.ylabel("True Labels", fontweight="bold")
    plt.xlabel("Predicted Labels", fontweight="bold")
    plt.savefig(f'/content/1-confusion-matrix-fold-
{fold_num}.png', bbox_inches='tight')

```

```

plt.close()
source_dir = '/content/'
destination_dir = '/content/drive/My
Drive/Skripsi/Evaluation'
shutil.copy2(f'/content/1-confusion-matrix-fold-
{fold_num}.png', destination_dir)

final_confusion_mtx = confusion_matrix(all_validation_labels,
all_validation_predictions)

if False:
    final_confusion_mtx = final_confusion_mtx.astype("float") /
    final_confusion_mtx.sum(axis=1)[:, np.newaxis]

plt.figure(figsize=(8, 6))
plt.imshow(final_confusion_mtx, interpolation="nearest",
cmap=plt.cm.OrRd)
plt.title(f"Final Confusion Matrix")
plt.colorbar()
tick_marks = np.arange(len(waste_labels.keys()))
plt.xticks(tick_marks, waste_labels.keys(), rotation=45)
plt.yticks(tick_marks, waste_labels.keys())
fmt = ".2f" if False else "d"
thresh = final_confusion_mtx.max() / 2.
for i, j in
itertools.product(range(final_confusion_mtx.shape[0]),
range(final_confusion_mtx.shape[1])):
    plt.text(j, i, format(final_confusion_mtx[i, j], fmt),
horizontalalignment="center",
            color="white" if final_confusion_mtx[i, j] >
thresh else "black")
plt.tight_layout()
plt.ylabel("True Labels", fontweight="bold")
plt.xlabel("Predicted Labels", fontweight="bold")
plt.savefig(f'/content/1-confusion-matrix-final.png',
bbox_inches='tight')
plt.close()
source_dir = '/content/'
destination_dir = '/content/drive/My Drive/Skripsi/Evaluation'
shutil.copy2(f'/content/1-confusion-matrix-final.png',
destination_dir)

# Model Convert to H5
model.save('/content/drive/My
Drive/Skripsi/Program/GarbageClassification-1.h5')

```



## Lampiran 2 Kode Implementasi Pembuatan Aplikasi Website

### Flask Application :

```

from flask import Flask, request, jsonify
import numpy as np
import tempfile
import tensorflow as tf
from flask_cors import CORS, cross_origin
from keras.models import load_model
from keras.preprocessing.image import load_img, img_to_array

app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
model = load_model('GarbageClassification-1.h5')

@app.route('/test')
@cross_origin
def test():
    return 'FLASK RUNNING'

@app.route('/predict', methods=['POST'])
@cross_origin()
def predict():
    try:
        image_file = request.files['image']

        if 'image' not in request.files:
            return jsonify({'error': 'No image in the request'}), 400

        temp_image = tempfile.NamedTemporaryFile(delete=False)
        image_file.save(temp_image)
        temp_image.close()

        img = load_img(temp_image.name, target_size=(
            224, 224))
        img_array = img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)

        class_probabilities = model.predict(img_array)
        waste_labels = {0: "cardboard", 1: "glass",
            2: "metal", 3: "paper", 4: "plastic", 5:
"trash"}

        response = ''

        predicted_class_index = np.argmax(class_probabilities)
        predicted_class_label = (
            waste_labels[predicted_class_index]).capitalize()
        predicted_class_probability =
float((class_probabilities[0,
predicted_class_index]))

```

```

        predicted_class_probability_percentage =
"{:.2f}%".format(
    predicted_class_probability * 100)

        response = {'Prediction': predicted_class_label,
                    'Class': predicted_class_label,
                    'ProbabilityPercentage':
predicted_class_probability_percentage}

        print(response)
        return jsonify(response)

    except Exception as e:
        print('An error occurred:', str(e))
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True, ssl_context='adhoc', host='0.0.0.0',
port=5000)

```

### *Migration Table Image :*

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('images', function (Blueprint $table) {
            $table->id();
            $table->string('path');
            $table->string('prediction');
            $table->string('probability');
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('images');
    }
};

```

### *Model Test Controller :*

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;
use App\Models\Image;

```

```

class ModelTestController extends Controller
{
    public function index()
    {
        return view('landing');
    }

    public function history()
    {
        $images = Image::all();

        if ($images->isEmpty()) {
            return view('history', ['images' => null]);
        } else {
            return view('history', ['images' => $images]);
        }
    }

    public function history_admin()
    {
        $images = Image::all();

        if ($images->isEmpty()) {
            return view('history-admin', ['images' => null]);
        } else {
            return view('history-admin', ['images' => $images]);
        }
    }

    public function history_delete(Request $request)
    {
        $id = $request->route('id');

        if ($id == null) {
            return redirect('/history/admin');
        } else {
            $delete_image = Image::where('id', $id)->delete();
            return redirect('/history/admin');
        }
    }

    public function makePrediction(Request $request)
    {
        $request->validate([
            'image' =>
            'required|image|mimes:jpeg,png,jpg,jfif|max:2048',
        ]);

        $predictions = $request->text1;
        $class = $request->text2;
        $prob_percentage = $request->text3;

        $imagePath = $request->file('image')->store('public/images');
    }
}

```

```

        $relativeImagePath = str_replace('public/', '',
$imagePath);

        $image = new Image();
        $image->path = $relativeImagePath;
        $image->prediction = $predictions;
        if ($predictions !== 'none') {
            $image->probability = $prob_percentage;
        } else if ($predictions === 'none') {
            $image->probability = 'none';
        }
        $image->save();

        return view('predict', [
            'predictions' => $predictions,
            'class' => $class,
            'prob_percentage' => $prob_percentage,
            'imagePath' => $imagePath
        ]);
    }
}

```

#### *Route Web :*

```

<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ModelTestController;

Route::get('/', function () {
    return view('welcome');
});

Route::get('/classification', [ModelTestController::class,
'index'])->name('classification');

Route::get('/history', [ModelTestController::class, 'history'])->
name('history');

Route::get('/history/admin', [ModelTestController::class,
'history_admin'])->name('history_admin');

Route::get('/history/admin/delete/{id}',
[ModelTestController::class, 'history_delete'])->
name('history_delete');

Route::post('/classification/predict',
[ModelTestController::class, 'makePrediction'])->name('make-
prediction');

```

#### *Base Layout :*

```

<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">

<head>
    <meta charset="utf-8">

```

```

    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <title>Garbage Classification @yield('title')</title>
    <link rel="icon" type="image/x-icon" href="{{
asset('/logo.png') }}" />
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;
600&display=swap" rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@4
00;600&display=swap" rel="stylesheet">
    <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.0.2/dist/css/boot
strap.min.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDg9JLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpUuCOMLASjC
" crossorigin="anonymous">
</head>

<body>
    @extends('layout.navbar')

    @yield('content')

    @extends('layout.footer')

    <script
src="https://cdn.jsdelivrivr.net/npm/bootstrap@5.0.2/dist/js/bootst
rap.bundle.min.js" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsPlUyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM
" crossorigin="anonymous"></script>
    <script src="https://cdn.tailwindcss.com"></script>
</body>

</html>

```

### Navbar :

```

<nav class="navbar navbar-light w-100 align-items-center
justify-content-center border border-dark p-0 m-0 pt-2 pb-2
fixed-top bg-white">
    <div class="container-fluid m-0 p-0 justify-content-center">
        <div class="row align-items-center w-100 justify-
content-center">
            <div class="col-4 align-items-center justify-
content-center p-0 m-0">
                <a class="navbar-brand d-flex p-0 m-0" href="/"
style="height: max-content; margin-left: 1rem !important;">
                    <div class="d-flex align-items-center">
                        
                        <h1 class="font-bold" style="margin-
left: 1rem; font-family: 'Poppins', sans-serif;">
                            Garbage<br>Classification
                        </h1>
                    </div>
                </a>
            </div>
        </div>
    </div>

```

```

        </div>
    </a>
</div>

<?php
function isActive($page)
{
    return $_SERVER['REQUEST_URI'] === $page;
}
?>
<div class="col-4 text-center align-items-center
justify-content-center">
    <ul class="nav align-items-center justify-
content-center">
        <li class="nav-item">
            <a class="nav-link hover:text-[#97D729]
<?php echo isActive('/') ? 'text-[#97D729] font-bold' : 'text-
black font-bolder'; ?>" style="font-family: 'Montserrat', sans-
serif;" href="/">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link hover:text-[#97D729]
<?php echo isActive('/classification') ? 'text-[#97D729] font-
bold' : 'text-black font-bolder'; ?>" style="font-family:
'Montserrat', sans-serif;"
href="/classification">Classification</a>
        </li>
        <li class="nav-item">
            <a class="nav-link hover:text-[#97D729]
<?php echo isActive('/history') ? 'text-[#97D729] font-bold' :
'text-black font-bolder'; ?>" style="font-family: 'Montserrat',
sans-serif;" href="/history">History</a>
        </li>
    </ul>
</div>

    <div class="col-4 align-items-center justify-
content-center"><br></div>
</div>
</nav>

```

#### Footer :

```

<footer class="text-center text-lg-start border border-dark
text-muted fixed-bottom bg-white">
    <div class="text-center p-2">
        <span class="fw-light" style="font-size: smaller; font-family:
'Monsterrat', sans-serif;"> Classification Garbage © 2023</span>
    </div>
</footer>

```

#### Landing Page :

```

@extends('layout.base-layout')

@section('title', '- Home')

```

```

@section('content')
<div class="relative sm:flex sm:justify-center sm:items-center
min-h-screen bg-dots-darker bg-center bg-white-100 dark:bg-dots-
lighter dark:bg-white-900">
  <div class="max-w-7xl mx-auto p-6 lg:p-8">
    <div class="grid gap-6 lg:gap-8">
      <h1 class="fw-bold fs-1 text-center"
style="color:black; font-family: 'Monsterrat', sans-
serif;">Garbage Classification</h1>
      <h2 class="fw-light fs-3 text-center"
style="color:black; font-family: 'Monsterrat', sans-
serif;">Classify your garbage with an easy-to-use platform</h2>
      <div class="text-center justify-content-center
align-items-center">
        <a class="btn bg-[#97D729] w-max p-3 fs-4 fw-
bold btn-outline-dark hover:bg-[#97D729]" style="color:black;
font-family: 'Montserrat', sans-serif; transition: opacity 0.2s
ease;" onmouseover="this.style.opacity='0.8'"
onmouseout="this.style.opacity='1'"
href="/classification">Classify Image!</a>
      </div>
    </div>
  </div>
</div>
@endsection

```

### Classification Page :

```

@extends('layout.base-layout')

@section('title', '- Classification')

@section('content')
<div class="relative sm:flex sm:justify-center sm:items-center
min-h-screen bg-dots-darker bg-center bg-white-100 dark:bg-dots-
lighter dark:bg-white-900">
  <div class="max-w-7xl mx-auto p-6 lg:p-8">
    <div class="grid gap-6 lg:gap-8">
      <h1 class="fw-bold fs-1 text-center"
style="color:black; font-family: 'Monsterrat', sans-
serif;">Classify Garbage</h1>
      <h2 class="fw-light fs-3 text-center"
style="color:black; font-family: 'Monsterrat', sans-
serif;">Classify your garbage's image to these class.<br>
Cardboard, Glass, Metal, Paper, Plastic, or
Trash </h2>
      <form method="POST" id="form" action="{{
route('make-prediction') }}" enctype="multipart/form-data"
class="text-center align-items-center justify-content-center">
        @csrf
        <div class="text-center justify-content-center
align-items-center mb-4">
          <input type="text" name="text1" id="text1"
style="color:black; display:none; border:1px solid black"
class="ml-20">

```

```

        <input type="text" name="text2" id="text2"
style="color:black; display:none; border:1px solid black"
class="ml-20">
        <input type="text" name="text3" id="text3"
style="color:black; display:none; border:1px solid black"
class="ml-20">
        <input type="file" name="image" id="image"
accept="image/*" style="color:black;" class="ml-20" required>
    </div>
</form>
<div class="text-center align-items-center justify-
content-center">
    <button id="predictBtn" class=" btn bg-[#97D729]
w-max p-3 text-md sm:text-xl md:text-2xl lg:text-2xl xl:text-2xl
2xl:text-2xl fw-bold btn-outline-dark hover:bg-[#97D729]"
style="color:black; font-family: 'Montserrat', sans-serif;
transition: opacity 0.2s ease;"
onmouseover="this.style.opacity='0.8'"
onmouseout="this.style.opacity='1'">Classify Image</button>
</div>
</div>
</div>
</div>
<script>

document.getElementById('predictBtn').addEventListener('click',
function() {
    var input = document.getElementById('image');
    var imageFile = input.files[0];

    if (imageFile) {
        var formData = new FormData();
        formData.append('image', imageFile);

        fetch('https://143.198.204.170:5000/predict', {
            method: 'POST',
            body: formData
        })
        .then(response => response.json())
        .then(data => {
            var form = document.getElementById('form');
            var responseInput =
document.createElement('input');
            responseInput.type = 'hidden';
            responseInput.name = 'responseData';
            responseInput.value = JSON.stringify(data);
            form.appendChild(responseInput);

            document.getElementById('text1').value =
data.Class;
            document.getElementById('text2').value =
data.Prediction;
            document.getElementById('text3').value =
data.ProbabilityPercentage;

```



```

        form.submit();
    })
    .catch(error => {
        console.error('Error:', error);
    });
} else {
    console.error('No image selected');
}
});
</script>
@endsection

```

### *Prediction Page :*

```

@extends('layout.base-layout')

@section('title', '- Prediction')

@section('content')
<div class="relative sm:flex sm:justify-center sm:items-center
min-h-screen bg-dots-darker bg-center bg-white-100 dark:bg-dots-
lighter dark:bg-white-900">
    <div class="max-w-7xl mx-auto p-6 lg:p-8">
        <div class="grid gap-4 lg:gap-6">
            <h1 class="fw-bold fs-1 text-center"
style="color:black; font-family: 'Monsterrat', sans-
serif;">Classification Result</h1>

            @if(isset($imagePath))
            <div class="justify-content-center align-items-
center d-flex">
                
            </div>
            @else
            <p>Image Corrupted.</p>
            @endif

            @if(isset($predictions))
            <div class="justify-content-center align-items-
center d-flex">

                @if($predictions === 'none')
                <div class="text-center">
                    <p class="fs-5 fw-bolder"><span class="fw-
normal">Model is not confident in any class</span></p>
                    <p class="fs-5 fw-bolder">Highest
Probability : <span class="fw-normal">{{ $class }} ({{
$prob_percentage }})</span></p>
                </div>
                @else
                <div>
                    <p class="fs-5 fw-bolder">Prediction : <span
class="fw-normal">{{ $predictions }}</span></p>

```



```

        <td class="align-middle">{{ $data->id
    }}</td>
        <td class="align-middle">
            
        </td>
        <td class="align-middle">{{ $data-
>prediction }}</td>
        <td class="align-middle">{{ $data-
>probability }}</td>
        <td class="align-middle">{{ $data-
>created_at }}</td>
    </tr>
    @endforeach
</tbody>
</table>
@else
<div class="text-center mt-2">
    <p>No images found</p>
</div>
@endif
</div>
</div>
</div>
@endsection

```

### History Admin Page :

```

@extends('layout.base-layout')

@section('title', '- History Admin')

@section('content')
<div class="relative sm:flex sm:justify-center sm:items-center
min-h-screen bg-dots-darker bg-center bg-white-100 dark:bg-dots-
lighter dark:bg-white-900">
    <div class="max-w-7xl mx-auto p-6 lg:p-8">
        <div class="grid mt-16 mb-16">
<h1 class="fw-bold fs-3 text-center">Classification History -
Admin</h1>
            @if ($images)
                <table class="table table-hover table-bordered mt-
4">
                    <thead class="bg-[#f6f6f6]">
                        <tr class="text-center">
                            <th scope="col">No</th>
                            <th scope="col">Image</th>
                            <th scope="col">Prediction</th>
                            <th scope="col">Probability</th>
                            <th scope="col">Timestamp</th>
                            <th scope="col">Action</th>
                        </tr>
                    </thead>
                    <tbody>

```

```

                                @foreach ($images as $data)
                                <tr class="text-center">
                                    <td class="align-middle">{{ $data->id
}}</td>
                                    <td class="align-middle">
                                        
                                        </td>
                                    <td class="align-middle">{{ $data-
>prediction }}</td>
                                    <td class="align-middle">{{ $data-
>probability }}</td>
                                    <td class="align-middle">{{ $data-
>created_at }}</td>
                                    <td class="align-middle"><a class="btn
btn-danger" href="/history/admin/delete/{{ $data->id
}}">Delete</a></td>
                                </tr>
                                @endforeach
                            </tbody>
                        </table>
                        @else
                        <div class="text-center mt-2">
                            <p>No images found</p>
                        </div>
                        @endif
                    </div>
                </div>
            </div>
        @endsection

```

## RIWAYAT HIDUP

### DATA PRIBADI

Nama Lengkap : Rheza Pandya Andhikaputra  
NPM : 140810200023  
Tempat, Tanggal Lahir : Jakarta, 20 Maret 2002  
Jenis Kelamin : Laki-laki  
Agama : Islam  
Alamat : Permata Puri Blok D3 No 1, Cisalak Pasar,  
Cimanggis, Depok, Jawa Barat  
Telepon : 087720201166  
Email : [rheza20001@mail.unpad.ac.id](mailto:rheza20001@mail.unpad.ac.id)

### RIWAYAT PENDIDIKAN

Tahun	Pendidikan
2020 - 2024	Program Studi S-1 Teknik Informatika, Fakultas Matematika dan Pengetahuan Alam, Universitas Padjadjaran
2017 - 2020	SMAN 48 Jakarta
2014 - 2017	SMP Labschool Cibubur
2008 - 2014	SD Islam Al-Azhar 20 Cibubur

**PENGALAMAN KERJA**

Tahun	Jabatan	Instansi
2023	Back End Developer	PT Intelix Global Crossing
2023	Machine Learning Cohort	Bangkit Academy led by Google, Tokopedia, Gojek, & Traveloka

**PENGALAMAN ORGANISASI**

Tahun	Jabatan	Instansi
2021-2022	Staff Hubungan Internal	Himatif FMIPA Unpad
2021	Ketua Divisi IT Comp	Informatics Festival
2022	Staff Hubungan Eksternal	BEM FMIPA Unpad
2022	Ketua Bidang Acara	Informatics Festival

**DATA PENCAPAIAN**

Tahun	Pencapaian
2023	Certified TensorFlow Developer