



TUGAS AKHIR - EC4801

DETEKSI AIRGAP PADA BETON MENGGUNAKAN CNN DARI DATA GPRMAX DUA DIMENSI

Gilang Maulana

NRP 5024 20 1034

Dosen Pembimbing

Dion Hayu Ferdiantoro, S.T., M.T.

NIP 1994202011064

Dr. Arief Kurniawan, S.T., M.T.

NIP 19740907200212 1 001

Program Studi Strata 1 (S1) Teknik Komputer

Departemen Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2024

[Halaman ini sengaja dikosongkan]



TUGAS AKHIR - EC4801

DETEKSI AIRGAP PADA BETON MENGGUNAKAN CNN DARI DATA GPRMAX DUA DIMENSI

Gilang Maulana

NRP 5024 20 1034

Dosen Pembimbing

Dion Hayu Ferdiantoro, S.T., M.T.

NIP 1994202011064

Dr. Arief Kurniawan, S.T., M.T.

NIP 19740907200212 1 001

Program Studi Strata 1 (S1) Teknik Komputer

Departemen Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2024

[Halaman ini sengaja dikosongkan]



FINAL PROJECT - EC4801

AIRGAP DETECTION IN CONCRETE USING CNN FROM TWO-DIMENSIONAL GPRMAX DATA

Gilang Maulana

NRP 5024 20 1034

Advisor

Dion Hayu Ferdiantoro, S.T., M.T.

NIP 1994202011064

Dr. Arief Kurniawan, S.T., M.T.

NIP 19740907200212 1 001

Undergraduate Study Program of Computer Engineering

Department of Computer Engineering

Faculty of Intelligent Electrical and Informatics Technology

Sepuluh Nopember Institute of Technology

Surabaya

2024

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DETEKSI AIRGAP PADA BETON MENGGUNAKAN CNN DARI DATA GPRMAX DUA DIMENSI

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Teknik pada
Program Studi S-1 Teknik Komputer
Departemen Teknik Komputer
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh: **Gilang Maulana**
NRP. 5024 20 1034

Disetujui oleh Tim Penguji Tugas Akhir:

Dion Hayu Ferdiantoro, S.T., M.T.
NIP: 1994202011064

(Pembimbing I)

.....

Dr. Arief Kurniawan, S.T., M.T.
NIP: 19740907200212 1 001

(Pembimbing II)

.....

Dr. Diah Puspito Wulandari, S.T., M.Sc.
NIP: 19801219200501 2 001

(Penguji I)

.....

Dr. Eko Mulyanto Yuniarno, S.T., M.T.
NIP: 19680601199512 1 009

(Penguji II)

.....

Arta Kusuma Hernanda, S.T., M.T..
NIP: 1996202311024

(Penguji III)

.....

Mengetahui,
Kepala Departemen Teknik Komputer FTEIC - ITS

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T..
NIP. 19700313 199512 1 001

SURABAYA
Mei, 2024

[Halaman ini sengaja dikosongkan]

APPROVAL SHEET

AIRGAP DETECTION IN CONCRETE USING CNN FROM TWO-DIMENSIONAL GPRMAX DATA

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree Bachelor of Engineering at
Undergraduate Study Program of Computer Engineering
Department of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology

By: **Gilang Maulana**
NRP. 5024 20 1034

Approved by Final Project Examiner Team:

Dion Hayu Ferdiantoro, S.T., M.T. (Advisor I)
NIP: 1994202011064

Dr. Arief Kurniawan, S.T., M.T. (Co-Advisor II)
NIP: 19740907200212 1 001

Dr. Diah Puspito Wulandari, S.T., M.Sc. (Examiner I)
NIP: 19801219200501 2 001

Dr. Eko Mulyanto Yuniarno, S.T., M.T. (Examiner II)
NIP: 19680601199512 1 009

Arta Kusuma Hernanda, S.T., M.T.. (Examiner III)
NIP: 1996202311024

Acknowledged,
Head of Computer Engineering Department F-ELECTICS - ITS

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T..
NIP. 19700313 199512 1 001

SURABAYA
May, 2024

[Halaman ini sengaja dikosongkan]

PERNYATAAN ORISINALITAS

Yang bertanda tangan dibawah ini:

Nama Mahasiswa / NRP : Gilang Maulana / 5024 20 1034
Departemen : Teknik Komputer
Dosen Pembimbing / NIP : Dion Hayu Ferdiantoro, S.T., M.T. / 1994202011064

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "DETEKSI AIRGAP PADA BETON MENGGUNAKAN CNN DARI DATA GPRMAX DUA DIMENSI" adalah hasil karya sendiri, berfsifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

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

Surabaya, May 2024

Mengetahui
Dosen Pembimbing

Mahasiswa

Dion Hayu Ferdiantoro, S.T., M.T.
NIP. 1994202011064

Gilang Maulana
NRP. 5024 20 1034

[Halaman ini sengaja dikosongkan]

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP : Gilang Maulana / 5024 20 1034
Department : Computer Engineering
Advisor / NIP : Dion Hayu Ferdiantoro, S.T., M.T. / 1994202011064

Hereby declared that the Final Project with the title of "*AIRGAP DETECTION IN CONCRETE USING CNN FROM TWO-DIMENSIONAL GPRMAX DATA*" is the result of my own work, is original, and is written by following the rules of scientific writing.

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

Surabaya, May 2024

Acknowledged
Advisor

Student

Dion Hayu Ferdiantoro, S.T., M.T.
NIP. 1994202011064

Gilang Maulana
NRP. 5024 20 1034

[Halaman ini sengaja dikosongkan]

ABSTRAK

Nama Mahasiswa : Gilang Maulana
Judul Tugas Akhir : DETEKSI AIRGAP PADA BETON MENGGUNAKAN CNN DARI DATA GPRMAX DUA DIMENSI
Pembimbing : 1. Dion Hayu Ferdiantoro, S.T., M.T.
 2. Dr. Arief Kurniawan, S.T., M.T.

Beton memegang peranan penting dalam berbagai proyek infrastruktur. Menurut data, sekitar 70% dari proyek-proyek konstruksi di dunia menggunakan beton sebagai material utamanya. Adanya rongga udara dapat mengurangi kekuatan beton hingga 30% dan mempengaruhi kinerjanya dalam jangka panjang. Pada penelitian kali ini, digunakan CNN 2D untuk mengklasifikasi data sinyal B-Scan hasil generate dari simulasi GPR menggunakan gprMax. File input untuk simulasi dipersiapkan secara otomatis menggunakan python yang nantinya akan digenerate menggunakan GPU. Data tersebut akan dilakukan preprocessing agar dapat dilakukan proses training. Digunakan 5 model CNN 2D pada percobaan kali ini dimana model pertama dilakukan percobaan sebanyak 3 kali untuk mengetahui pembagian data training yang optimal. Model hasil training nantinya akan diuji pada data yang telah disiapkan untuk testing. Digunakan pula Roboflow sebagai pembanding dari CNN 2D dan YOLOv9 sebagai pengujian tahap akhir pada implementasi deteksi berbentuk website. Hasil dari penelitian ini menunjukkan bahwa model CNN 2D yang digunakan mampu mengklasifikasi data sinyal B-Scan dengan akurasi sebesar 0.9964 pada model ke-1 dengan pembagian data 70/15/15. Pada data valiadasi, didapatkan inference time sebesar 115ms/step dengan F1 score sebesar 1.0. Hasil klasifikasi dengan Roboflow juga memiliki akurasi yang baik sebesar 0.986. Pada pengujian tahap akhir, YOLOv9 mampu mendeteksi rongga udara dengan baik dengan akurasi 0.979.

Kata Kunci: Beton, Rongga Udara, CNN, GPR, gprMax.

[Halaman ini sengaja dikosongkan]

ABSTRACT

*Name : Gilang Maulana
Title : AIRGAP DETECTION IN CONCRETE USING CNN FROM TWO-DIMENSIONAL GPRMAX DATA
Advisors : 1. Dion Hayu Ferdiantoro, S.T., M.T.
2. Dr. Arief Kurniawan, S.T., M.T.*

Concrete plays a crucial role in various infrastructure projects. According to data, about 70% of construction projects worldwide use concrete as their primary material. The presence of air voids can reduce the strength of concrete by up to 30% and affect its performance over the long term. In this study, a 2D CNN is used to classify B-Scan signal data generated from GPR simulations using gprMax. The input files for the simulations are automatically prepared using Python, and later generated using a GPU. The data undergoes preprocessing to facilitate the training process. Five 2D CNN models are tested in this experiment, with the first model undergoing three trials to determine the optimal training data split. The trained models are then tested on data prepared for testing. Roboflow is also used to compare with the 2D CNNs, and YOLOv9 is employed as the final stage tester in the website-based detection implementation. The results show that the 2D CNN models can classify B-Scan signal data with an accuracy of 0.9964 in the first model with a data split of 70/15/15. In the validation data, an inference time of 115ms/step and an F1 score of 1.0 are recorded. The classification results with Roboflow also show good accuracy at 0.986. In the final stage testing, YOLOv9 effectively detects air voids with an accuracy of 0.979.

Keywords: Concrete, Air Gap, CNN, GPR, gprMax.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur kehadirat Tuhan Yang Maha Esa, atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penelitian ini yang berjudul "DETEKSI AIRGAP PADA BETON MENGGUNAKAN CNN DARI DATA GPRMAX DUA DIMENSI".

Penelitian ini disusun dalam rangka pemenuhan Tugas Akhir sebagai syarat kelulusan Mahasiswa ITS. Oleh karena itu, penulis mengucapkan banyak terima kasih kepada

1. Bapak Dr.Supeno Mardi Susiko Nugroho, ST.,MT, selaku Kepala Departemen Teknik Komputer, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember
2. Bapak Dion Hayu Ferdiantoro, S.T., M.T. selaku Dosen Pembimbing I dan Bapak Dr. Arief Kurniawan, S.T., M.T. telah memberikan arahan selama pengerjaan tugas akhir ini
3. Bapak - selaku dosen penguji I, Bapak - selaku dosen penguji II dan Bapak - selaku dosen penguji III yang telah memberikan saran dan revisi agar pengerjaan Buku Tugas Akhir ini dapat menjadi lebih baik
4. Bapak-Ibu dosen pengajar Departemen Teknik Komputer, atas ilmu dan pengajaran yang telah diberikan kepada penulis selama ini
5. Keluarga yang senantiasa membantu serta membiayai pendidikan sedari kecil hingga sarjana
6. Teman - teman Departemen Teknik Komputer

Akhir kata, semoga penelitian ini dapat memberikan manfaat kepada banyak pihak, penulis menyadari jika skripsi ini masih belum sempurna, dikarenakan keterbatasan ilmu yang dimiliki. Untuk itu penulis mengharapkan saran dan kritik yang bersifat membangun kepada penulis untuk menuai hasil yang lebih baik lagi.

Surabaya, Mei 2024

Gilang Maulana

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

ABSTRAK	i
ABSTRACT	iii
KATA PENGANTAR	v
DAFTAR ISI	vii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xv
Program	xvii
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Permasalahan	1
1.3 Tujuan	1
1.4 Batasan Masalah	2
1.5 Manfaat	2
2 TINJAUAN PUSTAKA	3
2.1 Penelitian Terkait	3
2.1.1 Deteksi Rebar Untuk Mengukur Perbedaan Material Rebar	3
2.1.2 Deteksi Rongga Udara pada Beton Menggunakan Radar dengan Konfigurasi ZOP	3
2.1.3 Klasifikasi Jenis Tanah dari GPR B Scan Menggunakan Teknik Deep Learning	3
2.2 Teori/Konsep Dasar	4
2.2.1 Rongga Udara	4
2.2.2 Ground Penetrating Radar	4
2.2.3 gprMax	6
2.2.4 Python	7
2.2.5 Convolutional Neural Network	8

2.2.6	Roboflow	10
2.2.7	YOLO	10
2.2.8	Google Colab	11
2.2.9	Tensorflow	12
2.2.10	PyCUDA	13
2.2.11	NextJS	13
2.2.12	FastAPI	15
2.2.13	Redis	15
2.2.14	TypeScript	15
3	METODOLOGI	17
3.1	Deskripsi Sistem	17
3.1.1	<i>Generate</i> Data	17
3.1.2	<i>Preprocessing</i> Data	18
3.1.3	<i>Training</i> Data	19
3.1.4	Pengujian Klasifikasi	26
3.1.5	Deteksi YOLOv9	26
3.1.6	Hasil	27
3.2	Implementasi Deteksi	27
3.3	Kode Program	29
3.3.1	File Input Generate Data	30
3.3.2	Generate File Input gprMax dengan Rongga Udara	31
3.3.3	Setup GPU untuk Generate Data	32
3.3.4	Generate Data Sinyal GPR	32
3.3.5	<i>Preprocessing</i> Data	37
3.3.6	<i>Training</i> Data	40
3.3.7	Pengujian Klasifikasi Lanjut Revisi	42
3.3.8	Deteksi YOLOv9	42
4	PENGUJIAN DAN ANALISIS	45
4.1	Pengujian Klasifikasi Menggunakan CNN 2-Dimensi	45
4.1.1	Percobaan Pertama	45
4.1.2	Percobaan Kedua	47
4.1.3	Percobaan Ketiga	49
4.1.4	Percobaan Keempat	51

4.1.5	Percobaan Kelima	54
4.1.6	Percobaan Keenam	56
4.1.7	Percobaan Ketujuh	58
4.2	Pengujian Klasifikasi Roboflow	61
4.3	Pengujian Deteksi YOLOv9	63
5	PENUTUP	65
5.1	Kesimpulan	65
5.2	Saran	65
DAFTAR PUSTAKA		67
BIOGRAFI PENULIS		151

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

2.1	Prinsip kerja GPR (Persico, 2014)	4
2.2	Sinyal A-scan (Daniels, 2004)	5
2.3	Sinyal B-scan (Daniels, 2004)	5
2.4	Sinyal C-scan (Daniels, 2004)	6
2.5	Contoh file input simulasi gprMax (Craig Warren, 2016)	6
2.6	Pemodelan sinyal 2D pada gprMax (Goiannopoulos, 2005)	7
2.7	Pemodelan sinyal 3D pada gprMax (Goiannopoulos, 2005)	7
2.8	Arsitektur CNN sederhana (Keiron O Shea, 2015)	9
2.9	Aktivasi dari convolutional layer pertama (Keiron O Shea, 2015)	9
2.10	Representasi visual convolution layer (Keiron O Shea, 2015)	9
2.11	Google Colab (Kimm et al., 2021)	12
2.12	Built-in CSS pada Next.js (Lazuardy & Anggraini, 2022)	13
2.13	Routing pada Next.js(Lazuardy & Anggraini, 2022)	14
2.14	Pre-rendering pada Next.js (Lazuardy & Anggraini, 2022)	14
3.1	Blok Diagram Penelitian	17
3.2	Data Sinyal gprMax Setelah Dibersihkan	18
3.3	Data Sinyal gprMax Setelah Dibersihkan	19
3.4	Data Sinyal gprMax Setelah Dibersihkan	19
3.5	Arsitektur Model CNN 2D Pertama	20
3.6	Arsitektur Model CNN 2D Kedua	22
3.7	Arsitektur Model CNN 2D Ketiga	23
3.8	Arsitektur Model CNN 2D Keempat	24
3.9	Arsitektur Model CNN 2D Kelima	25
3.10	Flowchart Penggunaan Website	27
3.11	Flowchart Website	28
3.12	Gambar Platform Website 1	29
3.13	Gambar Platform Website 2	29
3.14	Gambar Platform Website 3	29
3.15	Input File Generate Data	30
3.16	Flowchart Setup GPU	31

3.17 Flowchart Setup GPU	32
3.18 Kode Otomatisasi Generate Data	34
3.19 Input File Generate Data	36
3.20 Mengubah Format Gambar Sinyal	37
3.21 Proses Binarisasi	39
3.22 Training CNN 2D	40
3.23 Pengujian Klasifikasi	42
3.24 Deteksi YOLOv9	43
4.1 Grafik Akurasi dan Loss Model Percobaan Pertama	45
4.2 Confussion Matrix Training dan Validasi Percobaan Pertama	46
4.3 Confussion Matrix Testing Percobaan Pertama	46
4.4 Hasil Klasifikasi Percobaan Pertama 1	46
4.5 Hasil Klasifikasi Percobaan Pertama 2	47
4.6 Hasil Klasifikasi Percobaan Pertama 3	47
4.7 Hasil Klasifikasi Percobaan Pertama 4	47
4.8 Grafik Akurasi dan Loss Model Percobaan Kedua	47
4.9 Confussion Matrix Training dan Validasi Percobaan Kedua	48
4.10 Confussion Matrix Testing Percobaan Kedua	48
4.11 Hasil Klasifikasi Percobaan Kedua 1	48
4.12 Hasil Klasifikasi Percobaan Kedua 2	49
4.13 Hasil Klasifikasi Percobaan Kedua 3	49
4.14 Hasil Klasifikasi Percobaan Kedua 4	49
4.15 Grafik Akurasi dan Loss Model Percobaan Ketiga	49
4.16 Confussion Matrix Training dan Validasi Percobaan Ketiga	50
4.17 Confussion Matrix Testing Percobaan Ketiga	50
4.18 Hasil Klasifikasi Percobaan Ketiga 1	51
4.19 Hasil Klasifikasi Percobaan Ketiga 2	51
4.20 Hasil Klasifikasi Percobaan Ketiga 3	51
4.21 Hasil Klasifikasi Percobaan Ketiga 4	51
4.22 Grafik Akurasi dan Loss Model Percobaan Keempat	52
4.23 Confussion Matrix Training dan Validasi Percobaan Keempat	52
4.24 Confussion Matrix Testing Percobaan Keempat	53
4.25 Hasil Klasifikasi Percobaan Keempat 1	53
4.26 Hasil Klasifikasi Percobaan Keempat 2	53

4.27 Hasil Klasifikasi Percobaan Keempat 3	53
4.28 Hasil Klasifikasi Percobaan Keempat 4	53
4.29 Grafik Akurasi dan Loss Model Percobaan Kelima	54
4.30 Confussion Matrix Training dan Validasi Percobaan Kelima	55
4.31 Confussion Matrix Testing Percobaan Kelima	55
4.32 Hasil Klasifikasi Percobaan Kelima 1	55
4.33 Hasil Klasifikasi Percobaan Kelima 2	55
4.34 Hasil Klasifikasi Percobaan Kelima 3	56
4.35 Hasil Klasifikasi Percobaan Kelima 4	56
4.36 Grafik Akurasi dan Loss Model Percobaan Keenam	56
4.37 Confussion Matrix Training dan Validasi Percobaan Keenam	57
4.38 Confussion Matrix Testing Percobaan Keenam	57
4.39 Hasil Klasifikasi Percobaan Keenam 1	57
4.40 Hasil Klasifikasi Percobaan Keenam 2	58
4.41 Hasil Klasifikasi Percobaan Keenam 3	58
4.42 Hasil Klasifikasi Percobaan Keenam 4	58
4.43 Grafik Akurasi dan Loss Model Percobaan Ketujuh	59
4.44 Confussion Matrix Training dan Validasi Percobaan Ketujuh	59
4.45 Confussion Matrix Testing Percobaan Ketujuh	60
4.46 Hasil Klasifikasi Percobaan Ketujuh 1	60
4.47 Hasil Klasifikasi Percobaan Ketujuh 2	60
4.48 Hasil Klasifikasi Percobaan Ketujuh 3	60
4.49 Hasil Klasifikasi Percobaan Ketujuh 4	60
4.50 Hasil Training Loss dan Validation Accuracy Roboflow	61
4.51 Confussion Matrix Roboflow	62
4.52 Hasil Klasifikasi Roboflow 1	62
4.53 Hasil Klasifikasi Roboflow 2	62
4.54 Hasil Klasifikasi Roboflow 3	62
4.55 Hasil Klasifikasi Roboflow 4	63
4.56 Hasil Klasifikasi Roboflow 5	63
4.57 Metrics YOLOv9	63
4.58 Confussion Matrix YOLOv9	64
4.59 Hasil Deteksi YOLOv9 1	64
4.60 Hasil Deteksi YOLOv9 2	64

4.61 Hasil Deteksi YOLOv9 3	64
4.62 Hasil Deteksi YOLOv9 4	64

DAFTAR TABEL

3.1	spesifikasi GPU	17
3.2	Konfigurasi Jaringan YOLOv9	26
3.3	Pengaturan Hyperparameter YOLOv9	27
4.1	Tabel Perbandingan Hasil F1 Score dan Inference Time	61

[Halaman ini sengaja dikosongkan]

DAFTAR PROGRAM

5.1	Contoh File Input gprMax	71
5.2	Program untuk Menghasilkan File Input gprMax	71
5.3	Program Setup GPU	74
5.4	Program Otomatisasi Generate Data gprMax dengan GPU	75
5.5	Program Generate Data Melalui Augmentasi	78
5.6	Program untuk Memformat Ulang Tipe File	80
5.7	Program untuk Binarization Gambar	81
5.8	Program CNN 2-Dimensi Percobaan Pertama	82
5.9	Program CNN 2-Dimensi Percobaan Kedua	91
5.10	Program CNN 2-Dimensi Percobaan Ketiga	100
5.11	Program CNN 2-Dimensi Percobaan Keempat	109
5.12	Program CNN 2-Dimensi Percobaan Kelima	118
5.13	Program CNN 2-Dimensi Percobaan Keenam	127
5.14	Program CNN 2-Dimensi Percobaan Ketujuh	136
5.15	Program Training YOLOv9	146
5.16	Pengujian Model CNN 2-Dimensi	147
5.17	Program Pengujian Model YOLOv9	148

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

1.1 Latar Belakang

Beton, sebagai salah satu material konstruksi yang paling sering digunakan, memegang peranan penting dalam berbagai proyek infrastruktur. Menurut data, sekitar 70% dari proyek-proyek konstruksi di dunia menggunakan beton sebagai material utamanya (Peng et al., 2022). Kesehatan dan integritas beton sangat penting untuk menjamin keamanan dan durabilitas struktur yang dibangun. Salah satu masalah yang sering dihadapi dalam konstruksi beton adalah adanya air gap atau rongga udara. Rongga udara ini terbentuk akibat proses pencampuran dan pengecoran beton yang kurang sempurna (Karisma, 2022). Adanya rongga udara dapat mengurangi kekuatan beton hingga 30% (Goncharov, 2021) dan mempengaruhi kinerjanya dalam jangka panjang. Selain itu, kerusakan lain seperti retak, amorfisasi, dan delaminasi juga dapat terjadi pada struktur beton (F. M. Rodrigues, 2022).

Untuk memastikan kualitas beton, diperlukan metode pengujian. Ada dua jenis metode pengujian, yaitu destructive dan non-destructive testing (NDT). Metode destructive melibatkan pengambilan sampel beton dan pengujian di laboratorium, namun metode ini dapat merusak struktur dan memerlukan biaya yang lebih besar (Karisma, 2022). Sebaliknya, NDT memungkinkan pengujian tanpa merusak struktur. Salah satu alat NDT yang populer adalah Ground Penetrating Radar (GPR). GPR memungkinkan deteksi objek tertanam di dalam beton, seperti rebar atau tulangan besi, serta adanya rongga udara atau air gap (F. M. Rodrigues, 2022). Namun, interpretasi data dari GPR memerlukan keahlian khusus dan seringkali sulit dilakukan secara manual.

Dalam dekade terakhir, perkembangan teknologi deep learning, khususnya Convolutional Neural Network (CNN), telah memberikan kemajuan signifikan dalam berbagai aplikasi, termasuk dalam bidang pengenalan pola dan analisis citra. CNN adalah salah satu metode yang dapat digunakan dalam pemanfaatan data dari GPR untuk mendeteksi objek tertanam di dalam beton dengan akurasi yang lebih tinggi (Peng et al., 2022). Oleh karena itu, penelitian ini akan diajukan dengan judul "Deteksi Airgap di Dalam Beton Menggunakan CNN Multi-label dari Data gprMax 2 Dimensi".

1.2 Permasalahan

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, maka dapat ditarik beberapa poin permasalahan yaitu adanya airgap di dalam beton yang dapat mempengaruhi kesehatan dari beton itu sendiri dan metode pendekripsi airgap yang baik.

1.3 Tujuan

Tujuan dari penelitian ini adalah mendekripsi adanya airgap di dalam beton dan menentukan model CNN yang baik untuk deteksi airgap.

1.4 Batasan Masalah

Batasan masalah pada penelitian kali ini mengikuti referensi dari penelitian yang berjudul "Perencanaan Bangunan Gedung Tahan Gempa 11 Lantai dengan Sistem Ganda" (Jeply Murdiaman Guci, 2021) antara lain:

1. Objek yang akan dimasukkan ke dalam deteksi adalah airgap dan rebar
2. Simulasi model yang dibentuk dengan gprMax akan berfokus pada sintesis sinyal B-Scan berbentuk parabolik dengan center frequency 4.5 GHz dan bentuk gelombang ricker
3. Jumlah airgap yang dideteksi berjumlah 1 hingga 2 buah objek irregular yang terdiri dari beberapa objek cavities yang saling menempel.
4. Digunakan ukuran beton sebagai media simulasi yakni beton untuk lantai dengan ketebalan 20 cm dengan error untuk posisi ketinggian rebar +- 1 cmn
5. Airgap yang dideteksi akan direpresentasikan dalam objek berbentuk irregular dengan ukuran airgap berdiameter 2 cm hingga 7 cm
6. Rebar yang digunakan di dalam beton untuk lantai berdiameter 14 mm

1.5 Manfaat

Manfaat dari penelitian ini adalah menghailkan sebuah metode yang dapat melakukan proses deteksi objek berdasarkan sinyal Ground Penetrating Radar (GPR) yang didapatkan dari hasil simulasi gprMax dengan menggunakan CNN. Hasil deteksi yang optimal diharapkan dapat membantu untuk pengaplikasian deteksi ini pada kebutuhan di dunia nyata, contohnya pada perawatan bangunan, konstruksi, dan lain-lain. Data simulasi digunakan karena data simulasi menggunakan gprMax memiliki hasil yang serupa dengan data asli menggunakan alat. Selain itu, penggunaan data simulasi didasari atas tidak tersedianya alat ground penetrating radar.

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Pada sub bab ini akan disampaikan penelitian terdahulu yang berkaitan dengan penelitian ini.

2.1.1 Deteksi Rebar Untuk Mengukur Perbedaan Material Rebar

Dalam studi yang berjudul "Rebar Detection - POD AApproach to Determine the Reliability of GPR Systems and to Quantify the Influence of Different Material Parameters" (Feistkorn, 2016) dipaparkan keefektivitasan dari GPR untuk mengukur pengaruh dari perbedaan parameter material. Pada penelitian ini dijelaskan bahwa pantulan sinyal GPR memiliki hasil keakuratan yang berbeda bergantung dari penyusun beton itu sendiri, termasuk diamter dari rebar dan kondisi dari beton itu sendiri. Penelitian ini memiliki hasil yang baik dengan tingkat akurasi 90%. Meskipun objek yang diuji adalah rebar, namun pada penelitian ini tidak berfokus pada deteksi rebar itu sendiri.

2.1.2 Deteksi Rongga Udara pada Beton Menggunakan Radar dengan Konfigurasi ZOP

Selain itu terdapat penelitian yang berjudul "Detection of air voids in concrete by radar in transmission mode" (Trela et al., 2015) disampaikan tentang pendekstrian rongga udara yang sangat berbahaya bagi kesehatan beton dan bangunan dalam jangka panjang. Pendekstrian rongga udara dilakukan pada beton buatan dengan konfigurasi ZOP. Penelitian ini memberikan hasil yang cukup baik dimana metode ZOP memiliki potensi dalam deteksi air void.

2.1.3 Klasifikasi Jenis Tanah dari GPR B Scan Menggunakan Teknik Deep Learning

Selain itu terdapat penelitian yang berjudul "Classification of soil types from GPR B Scans using deep learning techniques" (Barkataki et al., 2021) disampaikan tentang pengklasifikasian jenis tanah dimana gelombang sinyal B-Scan didapatkan dari simulasi gprMax. Penelitian ini memberikan hasil yang cukup baik dimana klasifikasi jenis tanah menghasilkan akurasi hingga 97%.

Dari ketiga penelitian tersebut, terdapat beberapa research gap antara lain: deteksi hanya berfokus ke satu objek saja, deteksi rongga udara dilakukan pada satu variasi bentuk media, dan menggunakan konfigurasi ZOP dari penggunaan radar dan penggunaan gprMax baru untuk sinyal tanah. Hal ini menunjukkan adanya peluang penelitian yang signifikan untuk mengembangkan model CNN yang dapat bekerja dengan data sinyal GPR untuk deteksi airgap atau rongga udara yang lebih akurat dan efisien dalam konteks struktur beton.

2.2 Teori/Konsep Dasar

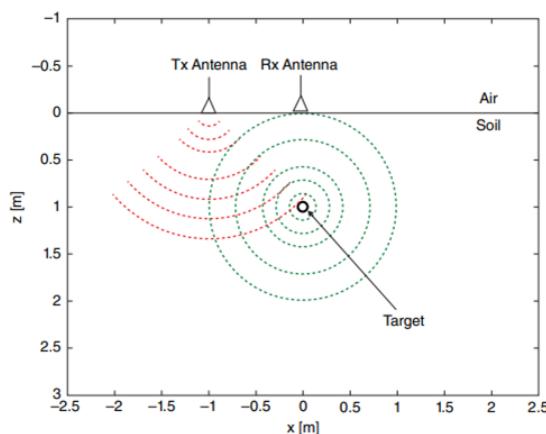
Pada sub bab ini akan dijelaskan teori yang berkaitan dengan penelitian ini.

2.2.1 Rongga Udara

Rongga udara di beton dapat membahayakan integritas dan daya tahan material. Karena pemasatan yang tidak tepat atau ketidakkonsistenan campuran, rongga ini sering terbentuk pada beton selama proses pengecoran. Hal ini mengganggu konsistensi matriks beton, yang mengurangi kapasitasnya untuk menahan beban dan meningkatkan kerentanannya terhadap retak tekanan. Rongga udara juga dapat memungkinkan uap air dan bahan kimia berbahaya masuk, yang mempercepat korosi tulangan baja yang tertanam dan kerusakan beton. Penetrasi kelembaban ini dapat menyebabkan kerusakan beku-cair di iklim dingin, yang membuat struktur lebih lemah. Jika beton diharapkan dapat menahan beban dinamis atau beban tumbukan, rongga udara mengurangi kemampuan material untuk menyerap dan mendistribusikan gaya(Zhu & Popovics, 2007).

2.2.2 Ground Penetrating Radar

Ground Penetrating Radar (GPR) atau juga disebut Surface Penetrating Radar (SPR) adalah radar untuk melihat kondisi di bawah tanah. GPR sendiri terdiri dari beberapa bagian inti seperti unit pusat, antena pemancar, antena penerima, dan komputer. Unit pusat memancarkan sinyal elektromagnetik ke tanah oleh antena pemancar. Sinyal dipancarkan ke semua arah, tetapi sebagian besar energi dipancarkan dalam volume kerucut di bawah antena, seperti ditunjukkan pada gambar berikut (Daniels, 2004):

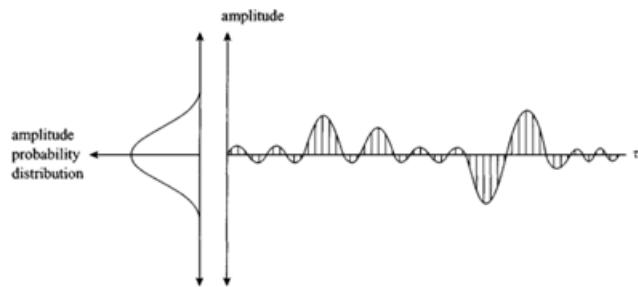


Gambar 2.1: Prinsip kerja GPR (Persico, 2014)

Ketika gelombang elektromagnetik bertemu dengan objek yang terkubur di dalam tanah, gelombang tersebar ke semua arah sesuai dengan pola tertentu. Pada penerapannya, antena pemancar dan penerima dipasang pada struktur yang kuat menjadi satu. Sinyal yang dikumpulkan biasanya ditampilkan secara real-time di layar komputer3 dan disimpan di hard disk computer. Prinsip kerja GPR tidak jauh berbeda dengan radar pada umumnya. Akan tetapi, terdapat perbedaan dalam hal teknologi, kebutuhan, aplikasi, dan pita frekuensi. GPR harus mengidentifikasi target statis, dan interpretasi data tidak diminta secara real-time. Di sisi lain, dalam prospeksi GPR gelombang elektromagnetik tidak merambat di udara tetapi malah merambat di media inang yang lebih rumit, biasanya berkerugian dan tidak homogen, mungkin dispersif, dan dalam beberapa kasus anisotropik dan/atau magnetik. GPR terbagi menjadi 2 sistem yakni pulsed

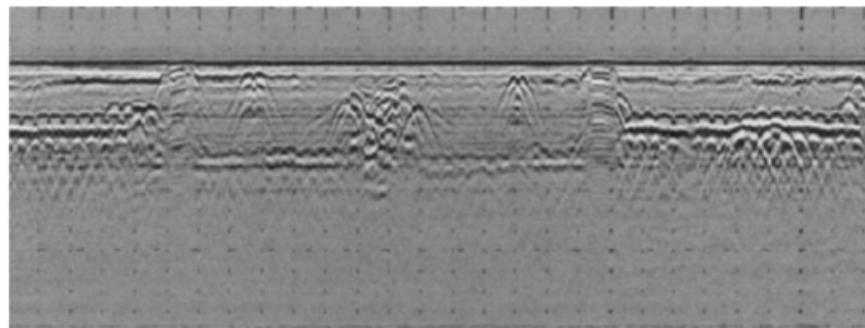
dan stepped-frequency. Pulsed memancarkan dan menerima gema ke pulsa elektromagnetik. Stepped-frequency mendekomposisi pulsa elektromagnetik menjadi komponen spektralnya dan memancarkannya secara berurutan. Akibatnya, memancarkan dan menerima kereta sinyal sinusoidal(Persico, 2014).

Mengacu dari terminology data GPR untuk titik ruang disebut A-scan atau jejak GPR.



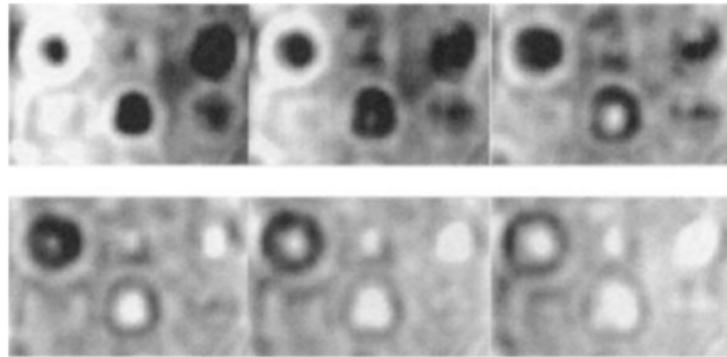
Gambar 2.2: Sinyal A-scan (Daniels, 2004)

Kumpulan jejak GPR untuk garis yang dipindai disebut B-scan. Meskipun GPR biasanya mengumpulkan data saat bergerak, model "stop-gather-and-go-on" dianggap cukup akurat dikarenakan perbedaan kecepatan antara sinyal elektromagnetik dan operator manusia. Penyimpanan A-scan bisa memakan waktu lebih lama karena alasan teknis.



Gambar 2.3: Sinyal B-scan (Daniels, 2004)

Namun, menjaga kecepatan konstan saat mengumpulkan data adalah praktik terbaik. Set data GPR dari B-scan paralel disebut C-scan. Yang biasa dilihat di lapangan adalah B-scan. Data awal, atau data mentah, bisa lebih jelas setelah pemrosesan(Daniels, 2004).



Gambar 2.4: Sinyal C-scan (Daniels, 2004)

2.2.3 gprMax

gprMax dikembangkan sebagai perangkat lunak lintas platform untuk Linux, Microsoft Windows, dan kemudian Mac OS X. GprMax ditulis dengan bahasa pemrograman C, dengan bagian yang memerlukan perhitungan intensif - loop solver FDTD - diparalelkan menggunakan OpenMP (Craig Warren, 2016). Pengembangan gprMax dalam implementasi fitur-fitur lanjut baru dan untuk meletakkan dasar untuk pengembangan masa depan, diputuskan bahwa kode harus ditulis ulang menggunakan bahasa yang berorientasi objek. Python adalah bahasa yang berorientasi objek dan menampilkan pengetikan dinamis dan manajemen memori otomatis. Ini juga dimaksudkan untuk sangat mudah dibaca dan dapat diperluas. Namun, kemudahan dan fleksibilitas Python memiliki beberapa kekurangan salah satunya adalah kecepatan dalam penulisan. Kehilangan kecepatan ini dapat diatasi dengan menggunakan Cython - sebuah superset dari Python yang menghasilkan kode C yang efisien yang dapat dikompilasi menjadi modul ekstensi. Versi baru dari gprMax telah dikembangkan ulang dengan kombinasi Python, NumPy, dan Cython dengan OpenMP, yang mempertahankan manfaat Python dengan kecepatan C yang paling (Warren et al., 2015).

gprMax menggunakan file input berupa teks di mana pengguna menentukan semua parameter untuk simulasi, misalnya, ukuran model, diskretisasi, jendela waktu, geometri, bahan, dan eksitasi. Penggunaan software gprMax sendiri tidak disertai dengan GUI. GUI tidak dikembangkan karena dinilai terlalu rumit bagi pengguna apabila berurusan dengan banyak nilai parameter yang harus diinputkan dalam satu waktu. Berikut adalah contoh file input untuk simulasi GPR 2D sederhana dari silinder logam yang dikubur di setengah ruang dielektrik tanpa kerugian(Craig Warren, 2016).

```

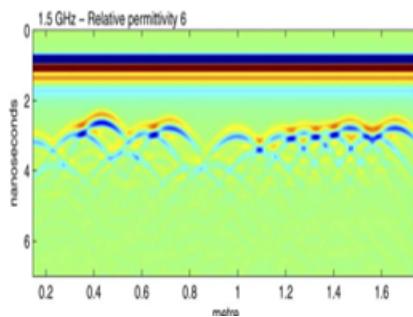
1 #domain: 0.240 0.210 0.002
2 #dx_dy_dz: 0.002 0.002 0.002
3 #time_window: 3e-9
4 #material: 6 0 1 0 half_space
5 #waveform: ricker 1 1.5e9 my_ricker
6 #hertzian_dipole: z 0.100 0.170 0
    ↵ my_ricker
7 #rx: 0.140 0.170 0
8 #box: 0 0 0 0.240 0.170 0.002
    ↵ half_space
9 #cylinder: 0.120 0.080 0 0.120 0.080
    ↵ 0.002 0.010 pec

```

Gambar 2.5: Contoh file input simulasi gprMax (Craig Warren, 2016)

Disamping itu, gprMax mengandung banyak fitur yang kuat dan dapat disesuaikan untuk pemodelan GPR. Selama 20 tahun terakhir, model antena telah dimasukkan dalam simulasi numerik GPR. Model antena actual telah dimasukkan terutama dari antena yang digunakan di dunia akademik atau untuk tujuan penelitian. Telah ada sedikit pekerjaan yang diterbitkan dari simulasi GPR dengan model antena komersial. Namun, kemajuan dalam kekuatan komputasi, ditambah dengan keinginan untuk menyelidiki informasi amplitudo kuantitatif dari GPR, terdapat kebutuhan untuk mengembangkan dan menggunakan model FDTD 3D rinci dari antena GPR realistik dalam simulas(Craig Warren, 2016).

GprMax2D dan GprMax3D merupakan dua program yang mengimplementasikan metode FDTD untuk pemodelan GPR di 2D dan 3D. Beberapa fiturnya adalah: antarmuka perintah yang mudah digunakan, kemampuan untuk memodelkan bahan dispersif, pemodelan target berbentuk kompleks serta simulasi ruang tak terbatas menggunakan kondisi batas penyerapan yang kuat. GprMax3D mampu mensimulasi antena GPR dan bahkan pengenalan garis transmisi pemberian mereka ke dalam model. GprMax2D digunakan untuk simulasi "signature" GPR sedangkan GprMax3D digunakan untuk simulasi yang lebih detail dan realistik terutama ketika perbandingan dengan data GPR nyata penting. Baik program GprMax2D dan 3D menggunakan file ASCII (teks) sederhana untuk mendefinisikan parameter model (Goiannopoulos, 2005).



Gambar 2.6: Pemodelan sinyal 2D pada gprMax (Goiannopoulos, 2005)

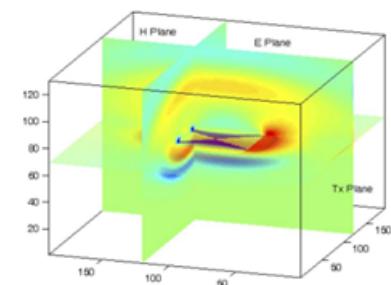


Fig. 7. Snapshots of three principal planes through a bow-tie antenna (units are in FDTD cells).

Gambar 2.7: Pemodelan sinyal 3D pada gprMax (Goiannopoulos, 2005)

2.2.4 Python

Python merupakan salah satu bahasa pemrograman serbaguna dan tingkat tinggi yang terkenal karena kepopuleritasnya. Python berkembang pertama kali pada akhir tahun 1980-an di Belanda, oleh Guido van Rossum. Versi pertama Python yakni 0.9.0, diluncurkan pada tahun 1991, diikuti oleh Python 2.0 pada tahun 2000. Python baru berkembang secara sig-

nifikan setelah berjalan 9 tahun lagi. Periode antara tahun 2005 dan 2013 menandai era penting dalam pengembangan Python, peningkatan ini tak lepas dari upaya yang membuat bahasa python itu sendiri. Python membedakan dirinya dengan menjadi bahasa yang diinterpretasikan, tidak memerlukan programnya untuk dikompilasi menjadi kode mesin. Sebagai gantinya, kode Python diubah menjadi bytecodes, yang kemudian dieksekusi oleh Mesin Virtual Python (PVM) yang terintegrasi dalam interpreter Python(H. Wang, 2023).

Menekankan aksesibilitas, sintaksnya yang sederhana memudahkan pembacaan dan penggunaan. Meskipun sederhana, Python tetap menjadi bahasa serbaguna yang kuat, menawarkan pengalaman pemrograman yang lebih efisien, terutama bila dibandingkan dengan Java. Kelebihan Python terletak pada dukungannya untuk berbagai pemrograman, termasuk pemrograman terstruktur, imperatif, berorientasi objek, fungsional, dan prosedural. Akan tetapi, ia tidak mendukung pemrograman logika. Python memiliki 2 kelebihan antara lain: pertama, Python memiliki berbagai konstruksi dan pernyataan bahasa yang kuat untuk representasi dan manipulasi data. Kedua, Python memiliki ekosistem yang luas dari paket, pustaka, dan modul pihak ketiga untuk berbagai kebutuhan pemrograman, ditambah dengan perpustakaan standar yang luas yang disertakan dalam setiap rilis Python. Keberagaman ini membuat Python menjadi pilihan populer di bidang seperti analisis data dan pembelajaran mesin, didukung oleh pustaka seperti NumPy, SciPy, Pandas, Matplotlib, dan TensorFlow(H. Wang, 2023).

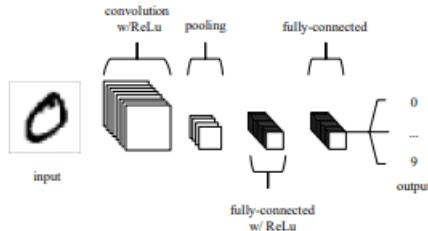
Kelebihan Python bukan tanpa kekurangan, mengingat interpreter Python membentuk lapisan dasar perangkat lunak, mereka tidak terhindar dari bug perangkat lunak. Masalah dalam interpreter Python cenderung lebih luas dan berpotensi lebih merusak dibandingkan dengan bug aplikasi biasa. Penanganan dan penyelesaian dari bug-bug ini dalam interpreter Python sangat penting untuk memastikan aplikasi Python dapat berjalan seperti yang diharapkan(Z. Wang et al., 2022).

2.2.5 Convolutional Neural Network

Artificial Neural Network (ANN) merupakan sistem pemrosesan komputasi yang terinspirasi oleh cara sistem saraf biologis otak beroperasi. ANNs terutama terdiri dari sejumlah besar node komputasi yang saling terhubung yang biasa disebut neuron dimana neuron yang bekerja bersama-sama secara terdistribusi untuk belajar secara kolektif dari input agar dapat mengoptimalkan outputnya. Convolutional Neural Network (CNN) memiliki kesamaan dengan ANN yang mana terdiri dari neuron yang mengoptimalkan diri melalui pembelajaran. Setiap neuron akan tetap menerima input dan melakukan operasi dasar dari banyak ANN. Dari vektor gambar mentah input hingga output akhir skor kelas, seluruh jaringan akan tetap mengekspresikan satu fungsi skor perceptif (bobot). Lapisan terakhir akan berisi fungsi kerugian yang terkait dengan kelas, dan semua tips dan trik reguler yang dikembangkan untuk ANNs tradisional masih berlaku(Keiron O Shea, 2015).

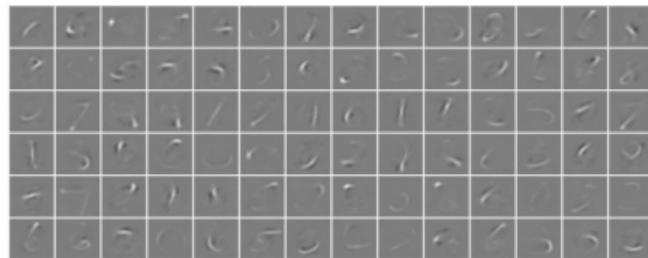
Bisa dikatakan bahwa CNN adalah model yang cukup baik dalam bidang pemrosesan gambar. CNN dapat memberikan hasil yang baik dalam klasifikasi gambar, pengenalan, segmentasi semantik, dan terjemahan mesin, dan dapat belajar serta mengekstrak fitur gambar secara mandiri. Namun, CNN hanya bisa dioperasikan pada data reguler, seperti gambar dengan ukuran tetap(Ruyue Xin, 2020). Perbedaan yang cukup terlihat antara CNN dan ANN tradisional CNN banyak digunakan dalam bidang pengenalan pola dalam gambar. Hal ini memungkinkan pengguna dalam mengkodekan fitur khusus gambar ke dalam arsitektur, membuat jaringan lebih cocok untuk tugas yang berfokus pada gambar - sekaligus mengurangi parameter yang dibutuhkan dalam mengatur model(Keiron O Shea, 2015).

CNN memiliki banyak variasi arsitektur. Namun, komponen dasar CNN bisa dikatakan mirip dimana terdiri dari tiga jenis lapisan, yaitu lapisan konvolusional, pooling, dan fully-connected. Lapisan konvolusional bertujuan untuk belajar representasi fitur dari input(Gu et al., 2018).



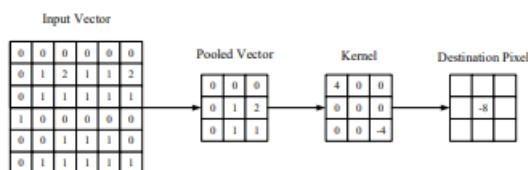
Gambar 2.8: Arsitektur CNN sederhana (Keiron O Shea, 2015)

Seperti pada ANN, input layer di CNN akan membawa nilai pixel dari gambar(Keiron O Shea, 2015). Lapisan konvolusi terbentuk oleh beberapa kernel konvolusi yang digunakan untuk menghitung peta fitur yang berbeda. Setiap neuron dari sebuah peta fitur terhubung dengan sebuah wilayah dari neuron tetangga di lapisan sebelumnya(Gu et al., 2018).



Gambar 2.9: Aktivasi dari convolutional layer pertama (Keiron O Shea, 2015)

Seperti namanya, lapisan konvolusi memainkan peran penting dalam cara kerja CNN. Parameter lapisan ini berfokus pada penggunaan kernel yang dapat dipelajari. Kernel ini memiliki ukuran kecil dalam dimensi spasial, tetapi menyebar sepanjang seluruh kedalaman input. Saat data mencapai lapisan konvolusi, dilakukan konvolusi setiap filter di seluruh dimensi spasial input untuk menghasilkan peta aktivasi 2D. Peta aktivasi ini dapat divisualisasikan, seperti yang terlihat pada Gambar 2.9. Saat melalui input, hasil kali skalar dihitung untuk setiap nilai dalam kernel. (Gambar 2.10) Jaringan akan belajar kernel yang 'aktif' ketika mereka melihat fitur tertentu pada posisi spasial tertentu dari input. Ini umumnya dikenal sebagai aktivasi(Keiron O Shea, 2015).



Gambar 2.10: Representasi visual convolution layer (Keiron O Shea, 2015)

Peta aktivasi dimiliki oleh setiap layer konvolusi yang akan ditumpuk sepanjang dimensi kedalaman membentuk volume output penuh dari lapisan konvolusi(Keiron O Shea, 2015).

Lapisan pooling memiliki tujuan untuk mencapai ketidakberubahan pergeseran dengan melakukan pengurangan resolusi fitur. Lapisan ini berada di antara dua lapisan konvolusi. Setiap peta fitur dari lapisan pooling tersambung dengan peta fitur yang sesuai dari lapisan konvolusi sebelumnya. Setelah beberapa lapisan konvolusi dan pooling, akan terdapat lapisan fully-connected yang bertujuan untuk menalar dalam tingkat tinggi. Lapisan ini mengambil semua neuron di lapisan sebelumnya dan menghubungkannya ke setiap neuron di lapisan saat ini untuk menghasilkan informasi semantik global(Gu et al., 2018).

Fully-connected layer berisi neuron yang terhubung langsung ke neuron di dua lapisan yang berdekatan(Keiron O Shea, 2015). Lapisan terakhir dari CNNs adalah lapisan output(Gu et al., 2018). Lapisan yang tidak kalah penting adalah lapisan ReLU. Tujuan dari ReLU adalah untuk melakukan peningkatan non-linearitas CNN. Lapisan ReLU sendiri tidak mengubah ukuran dari input dan lapisan ini tidak memiliki parameter. Pada penerapannya, lapisan ini aman untuk digunakan karena lapisan ReLU akan mengatur semua nilai negatif menjadi nol(Wu, 2017).

2.2.6 Roboflow

Roboflow adalah situs web yang memiliki banyak koleksi terhadap berbagai jenis dataset. Website ini juga menyediakan data dalam berbagai pilihan format untuk berbagai model machine learning(Deepa et al., 2023). Setidaknya, Roboflow memiliki lebih dari 100.000 pengguna yang telah mengumpulkan dan melabeli dataset mereka sendiri untuk kebutuhan kustom mereka. Hal tersebut menyebabkan Roboflow Universe telah banyak digunakan oleh para peneliti untuk digunakan dalam berbagai projek mereka khususnya dalam proyek deteksi objek(Ciaglia et al., 2022).

Sebagai platform computer vision, Roboflow digunakan untuk melatih model karena kemudahan dan kecepatan yang dimiliki dalam membuat model tanpa harus melalui proses pemrograman yang intensif. Roboflow memiliki fitur untuk mempersiapkan data, melatih dan mengembangkan model, serta banyak hal lainnya yang membuatnya menjadikan Roboflow sebagai platform dengan infrastruktur untuk mempercepat proses pembuatan model computer vision. RoboFlow menawarkan rangkaian lengkap alat dan layanan dengan tujuan menyederhanakan dan mempercepat pengembangan dan penerapan model computer vision. Hal ini sangat berguna dalam penggunaan visi komputer pada aplikasi dan proyek bagi pemrogram dan perusahaan. Fitur dan kemampuan utama RoboFlow meliputi anotasi data dan persiapan untuk memberi label pada video dan gambar, yang keduanya diperlukan untuk mengajarkan model computer vision untuk mengidentifikasi dan menemukan lokasi objek dalam gambar. Dengan menghasilkan varian data asli, RoboFlow menawarkan berbagai strategi augmentasi data yang membantu model machine learning menjadi lebih baik. Model computer vision tersebut nantinya dapat dilatih menggunakan TensorFlow, PyTorch, dan YOLO(Brucal et al., 2023).

2.2.7 YOLO

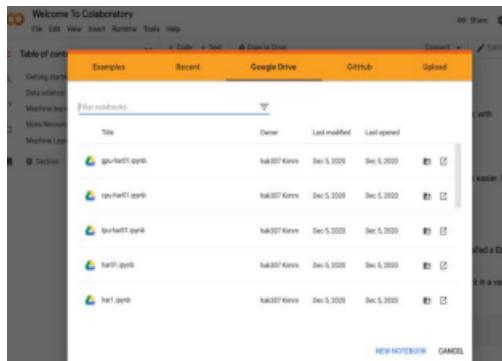
You Only Look Once (YOLO) adalah algoritma yang populer dan banyak digunakan. YOLO terkenal dengan karakteristik pendekripsi objeknya. YOLO pertamakali diperkenalkan pada tahun 2015. Dalam beberapa tahun terakhir, YOLO telah mengalami perkembangan dan mencapai versi yang lebih baru seperti YOLO V2, YOLO V3, YOLO V4, YOLO V5, dan seterusnya hingga YOLO V9. Ada beberapa versi revisi terbatas, seperti YOLO-LITE. Uku-

ran model yang kecil dan kecepatan perhitungan yang cepat merupakan inti dari algoritma deteksi target YOLO. YOLO cepat karena YOLO hanya perlu memasukkan gambar ke dalam jaringan untuk mendapatkan hasil deteksi akhir, sehingga YOLO juga dapat mengukur waktu deteksi video. YOLO secara langsung menggunakan gambar global untuk deteksi, yang dapat mengkodekan informasi global dan mengurangi kesalahan dalam mendeteksi latar belakang sebagai objek. Hasil tes YOLO tidak bagus untuk objek yang jaraknya sangat dekat satu sama lain dan dalam kelompok. Kinerja yang buruk ini disebabkan karena hanya dua kotak dalam grid yang diprediksi dan hanya termasuk dalam kelas objek baru dari kategori yang sama, sehingga muncul rasio aspek yang tidak normal, dan kondisi lain, seperti kemampuan generalisasi yang lemah(Jiang et al., 2022).

Arsitektur YOLO asli terdiri dari 24 convolution layers, diikuti oleh dua fully connected layers. YOLO memprediksi lebih dari satu kotak pembatas per sel kisi dimana kotak pembatas yang memiliki Intersection Over Union (IOU) tertinggi dengan ground truth dipilih, yang dikenal sebagai non-maxima suppression. YOLO memiliki dua cacat yaitu satu adalah posisi yang tidak akurat, dan yang lainnya adalah tingkat recall yang lebih rendah dibandingkan dengan metode berdasarkan rekomendasi area(Jiang et al., 2022). YOLO bukanlah algoritma pertama yang menggunakan Single Shot Detector (SSD) untuk mendeteksi objek. Single Shot Detector (SSD), Deconvolution Single Shot Detector (DSSD), RetinaNet, M2Det, dan RefineDet++ merupakan algoritma yang telah ada sebelum YOLO untuk melakukan pendekalian objek secara satu tahap. YOLO kelebihan sebagai pendekali objek dua tahap dalam hal akurasi dan waktu inferensi. Selain itu, dengan munculnya YOLO, berbagai aplikasi telah menggunakan YOLO untuk mendekali dan mengenali objek dalam berbagai konteks dan memiliki performa yang sangat baik dibandingkan dengan pendekali objek dua tahap lainnya(Dewan et al., 2022).

2.2.8 Google Colab

Google Colab adalah environment Jupyter notebook yang berjalan di Google cloud. Colab memungkinkan pengguna lain untuk menggunakan dan berbagi notebook Jupyter tanpa harus mengunduh, menginstal, atau menjalankan apa pun. Jupyter sendiri merupakan adalah proyek open source yang bekerja pada browser yang menyematkan bahasa skrip, pustaka, dan alat untuk visualisasi. Setiap file buku catatan di Jupyter terdiri dari beberapa sel, di mana memiliki outputnya menyatu dalam dokumen termasuk teks, tabel, bagan, dan grafik. Google Colab menyediakan akses langsung ke Google Drive dan GitHub serta layanan cloud gratis dengan GPU dan TPU. Platform perangkat keras yang banyak tersedia di cloud adalah CPU, TPU, dan GPU. CPU gratis untuk Google Colab dilengkapi dengan 2-core Intel Xeon @2.0GHz dan RAM 13 GB serta HDD 33 GB. Masa pakai maksimum VM di Google Colab adalah 12 jam dengan waktu idle 90 menit(Kimm et al., 2021).



Gambar 2.11: Google Colab (Kimm et al., 2021)

Google Colaboratory adalah platform dapat digunakan untuk melakukan edukasi dan penelitian mengenai machine learning. Colaboratory menyediakan runtime Python 2 dan 3 yang telah dikonfigurasi sebelumnya dengan pustaka machine learning dan AI yang penting, seperti TensorFlow, Matplotlib, dan Keras. Google Colaboratory dapat digunakan secara efektif untuk mempercepat tidak hanya deep learning tetapi juga aplikasi ilmiah berbasis GPU lainnya(Carneiro et al., 2018).

2.2.9 Tensorflow

Dibuat oleh peneliti yang berasal dari Google, TensorFlow adalah pustaka yang paling populer di antara kebanyakan pustaka lainnya. TensorFlow adalah pustaka perangkat lunak yang fleksibel dan skalabel untuk komputasi numerik menggunakan grafik dataflow. Dengan menggunakan pustaka TensorFlow ini, pengguna mampu memprogram dan melatih neural network serta model machine learning lainnya secara efisien dan menerapkannya ke produksi. Algoritme inti TensorFlow ditulis dalam C++ dan CUDA (Compute Unified Device Architecture) secara optimal. CUDA sendiri merupakan sebuah platform komputasi paralel dan API yang dibuat oleh NVIDIA. CUDA memiliki API yang tersedia dalam beberapa bahasa. Bahasa yang didukung secara resmi oleh TensorFlow adalah JavaScript, C++, Java, Go, dan Swift. TensorFlow juga tersedia untuk lebih banyak bahasa seperti C# dan Ruby. Akan tetapi, API untuk bahasa pemrograman Python adalah yang paling lengkap dan stabil(Pang et al., 2019).

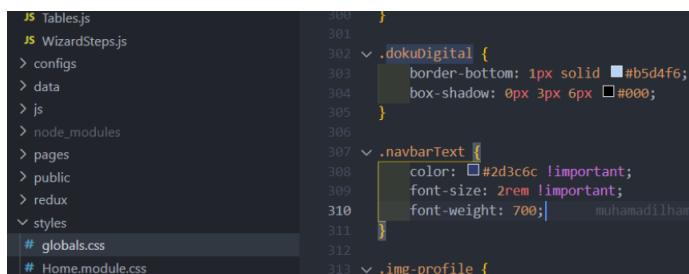
TensorFlow adalah platform machine learning menyeluruh yang didukung oleh Google. Pada dasarnya, program TensorFlow terdiri dari dua bagian utama: bagian konstruksi, yang memungkinkan pembuatan model jaringan machine learning atau deep learning, dan bagian eksekusi, yang memungkinkan untuk melatih dan mengevaluasi model jaringan. TensorFlow menyediakan API yang paling banyak digunakan karena kelengkapan dan stabilitasnya yang disajikan dalam bahasa pemrograman Python untuk mencapai kedua bagian tersebut. TensorFlow juga menyediakan pustaka TensorFlow Lite untuk penerapan model jaringan pada perangkat seluler, mikrokontroler, dan perangkat edge. TensorFlow Lite memungkinkan konversi model TensorFlow dasar menjadi versi terkompresi melalui apa yang disebut konverter TensorFlow Lite. Tensorflow sendiri memiliki Keras yang mana merupakan sebuah kerangka deep learning. Keras dibangun di atas TensorFlow versi 2, yang menyediakan API Python untuk memungkinkan pengembang menyederhanakan pembuatan dan eksperimen model jaringan(Contoli & Lattanzi, 2023).

2.2.10 PyCUDA

PyCUDA merupakan sebuah API (Application Programming Interface) dari bahasa pemrograman python untuk mengakses kartu grafis NVIDIA menggunakan CUDA. PyCUDA Banyak digunakan sebagai API untuk bahasa pemrograman python adalah karena kode yang ditulis seperti kode python biasa dan penanganan kesalahan juga lebih sederhana daripada bahasa C atau CUDA itu sendiri(Koprawi, 2020). PyCUDA merepresentasikan pendekatan berbasis skrip untuk generasi kode runtime GPU. PyCUDA memberikan kemudahan akses untuk bahasa pemrograman python. Salah satu kemudahan itu ditunjukkan dengan memberikan pengguna akses ke API komputasi paralel milil NVIDIA itu sendiri. Hal tersebut memungkinkan kode CUDA untuk disematkan sebagai string dalam skrip bahasa pemrograman python. String tersebut akan diuraikan oleh pycuda.compilerSourceModule, yang mana akan dikompilasi menggunakan nvcc sebagai driver kompilator CUDA dan terhubung dengan library runtime CUDA. PyCUDA memungkinkan pengguna untuk mengakses GPU NVIDIA menggunakan CUDA dari bahasa pemrograman Python. PyCUDA dirancang untuk pengembang CUDA yang ingin mengintegrasikan kode yang sudah ditulis dalam CUDA dengan Python(Leung, 2023).

2.2.11 NextJS

Next.JS adalah framework tangguh yang mampu meningkatkan kinerja situs web dan SEO melalui berbagai teknik pengoptimalan. Next.js juga menerapkan Incremental Static Regeneration (ISR) dan mengurangi ukuran beban JavaScript melalui pemisahan kode dan React. Selain kemampuan pengoptimalan dan peningkatan kinerjanya, Next.JS juga menawarkan serangkaian fitur yang memudahkan pengembang dalam membangun dan memelihara situs web mereka. Next.js menyediakan fitur hot reload yang memungkinkan pengembang membuat perubahan pada kode mereka dan melihat hasilnya secara real time tanpa memuat ulang halaman secara manual. Ini bisa menjadi penghemat waktu yang sangat besar bagi pengembang, memungkinkan mereka menguji dan mengulangi kode mereka dengan cepat tanpa perlu pemuatan ulang yang berulang(Patel, 2023). Sebagai salah satu framework yang sering terkenal dan sering digunakan untuk mengembangkan website, Next.js memiliki beberapa kelebihan dalam pengaplikasiannya. Kelebihan pertama yang dimiliki oleh Next.js adalah didukung dengan kemampuan untuk menggunakan CSS secara langsung pada Next.js. Berbeda dengan penggunaan CSS pada umumnya dimana modul CSS berisi pengaturan CSS untuk semua komponen, pada Next.js terdapat Global CSS yang memungkinkan kode CSS dapat diakses oleh semua komponen(Lazuardy & Anggraini, 2022).



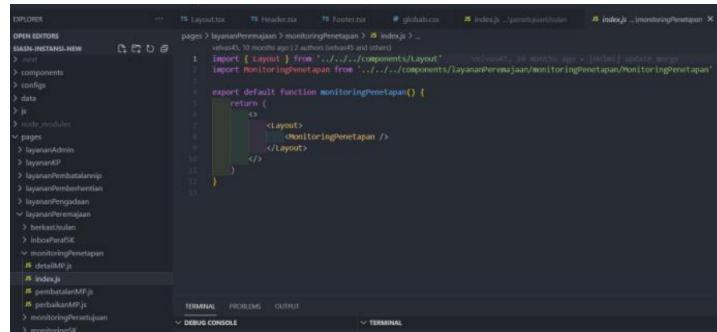
The screenshot shows a code editor interface. On the left, there is a tree view of files and folders. Files listed include Tables.js, WizardSteps.js, configs, data, js, node_modules, pages, public, redux, and styles. Under styles, there are two files: # globals.css and # Home.module.css. The main pane displays a portion of a CSS file with the following code:

```
300 }
301 < .dokuDigital {
302   border-bottom: 1px solid #b5d4f6;
303   box-shadow: 0px 3px 6px #000;
304 }
305 }
306 < .navbarText {
307   color: #2d3c6c !important;
308   font-size: 2rem !important;
309   font-weight: 700;
310 }
311 }
312 < .img-profile {
```

Gambar 2.12: Built-in CSS pada Next.js (Lazuardy & Anggraini, 2022)

Kelebihan selanjutnya adalah framework Next.js sudah memiliki dukungan mekanisme routing terletak pada folder pages, sehingga tidak memerlukan pustaka tambahan untuk melakukan routing. Routing komponen React cukup dengan menuliskan nama file langsung di folder

pages yang disediakan oleh Next.js. Dengan ini, pengguna mampu memasukkan komponen-komponen yang terdapat pada folder komponen ke dalam file routing yang telah disatukan dengan Layout komponen sebagai komponen statis sehingga suatu komponen dapat ditampilkan selama proses pengunggahan(Lazuardy & Anggraini, 2022).



Gambar 2.13: Routing pada Next.js(Lazuardy & Anggraini, 2022)

Selain itu, kelebihan lain yang dimiliki oleh Next.js adalah Framework Next.js sudah mendukung SSR dalam merender halaman web, sehingga membuat pengguna tidak akan melihat halaman kosong pada pemuatan awal dan mengurangi beban pada browser (Lazuardy & Anggraini, 2022).

```
measured at: https://nextjs.org/docs/messages/opt-out-auto-static-optimization
info - Collecting page data
info - Generating static pages (1/1)
info - Finalizing page optimization

Page Size
First load JS
  λ / 123 kB
  | _app 118 kB
  | λ /404 118 kB
  | λ /500 305 B
  | λ /layananPeremajaan/monitoringPenetapan 428 kB
  | λ /layananPeremajaan/monitoringPenetapan/detailMP 473 kB
  | λ /layananPeremajaan/monitoringPenetapan/pembatalanMP 192 kB
  | λ /layananPeremajaan/monitoringPenetapan/perbaikanMP 192 kB
  | λ /layananPeremajaan/monitoringPersetujuan 453 kB
  | λ /layananPeremajaan/monitoringSK 236 kB
  | λ /layananPeremajaan/monitoringTTDSK 194 kB
  | λ /layananPeremajaan/monitoringUsulan 900 kB

λ (Server) server-side renders at runtime (uses getInitialProps or getServerSideProps)
o (Static) automatically rendered as static HTML (uses no initial props)
• (SSG) automatically generated as static HTML + JSON (uses getStaticProps)
  (ISR) incremental static regeneration (uses revalidate in getStaticProps)
```

Gambar 2.14: Pre-rendering pada Next.js (Lazuardy & Anggraini, 2022)

Framework Next.js mendukung mekanisme pengambilan data dengan CSR, SSR, SSG, dan ISR yang mana tidak seperti aplikasi React.js biasa yang hanya mendukung pengambilan data dengan CSR. Sehingga mekanisme pengambilan data dapat disesuaikan dengan kebutuhan

aplikasi. Menggunakan SSR (server-side rendering) lebih baik untuk SEO (searchengine optimization), daripada CSR (client-side rendering) karena file HTML dirender di sisi server(Lazuardy & Anggraini, 2022). Terlepas dari kelebihan yang disebutkan di atas Next.JS mampu melakukan hosting situs web dengan mudah, dengan opsi untuk hosting di platform seperti Vercel dan GitHub Pages. Opsi hosting ini memberi pengembang cara yang nyaman sehingga lebih mudah untuk fokus dalam membangun dan mengoptimalkan situs(Patel, 2023).

2.2.12 FastAPI

FastAPI adalah framework Python berbasis web yang menyediakan lapisan bagi model ML untuk memberikan kinerja tinggi dan mengekspos fungsionalitas model ML sebagai layanan mikro yang tenang. FastAPI terintegrasi dengan baik ke dalam tampilan yang lebih sederhana untuk memprediksi hasil model ML. Aplikasi ini dapat dengan mudah diakses melalui web browser di Internet. Waktu respon aplikasi FastAPI cukup singkat dibandingkan aplikasi berbasis Flask. Selain itu, biometrik perilaku berpotensi meningkatkan keamanan akun pengguna dan mengurangi jumlah ancaman dan kerentanan tanpa memerlukan perangkat keras tambahan. Studi ini memberikan gambaran komprehensif mengenai penelitian atau upaya yang dilakukan untuk mengintegrasikan model ML dengan FastAPI untuk biometrik perilaku guna memberikan solusi yang lebih cepat dan hemat biaya dibandingkan Flask. Proses pengembangan lebih cepat berkat otomatisasi canggih editor dan pemeriksaan kesalahan otomatis yang disediakan oleh framework (Bansal & Ouda, 2022).

2.2.13 Redis

Redis adalah penyimpanan struktur data in-memory open-source yang dapat digunakan sebagai database, cache, dan perantara pesan. Redis mendukung berbagai struktur data, termasuk string, hashes, lists, sets, dan sorted sets. Dikenal karena kinerjanya yang luar biasa, Redis dapat menyimpan seluruh kumpulan data di memori, yang secara signifikan mempercepat akses data dibandingkan dengan database berbasis disk. Redis++ memperbaiki manajemen memori dan pengindeksan, yang meningkatkan kinerja dan mengurangi masalah fragmentasi memori dan kehilangan cache(Zhang et al., 2018). Redis telah diadaptasi untuk menggunakan teknologi RDMA melalui InfiniBand untuk lebih mempercepat kinerja dengan mengurangi latensi jaringan dan pemanfaatan CPU(Tang et al., 2017). Selain itu, Redis telah diperluas untuk menyertakan fitur keamanan seperti autentikasi dan enkripsi, sehingga lebih cocok untuk aplikasi perusahaan yang memerlukan penanganan data yang aman(Zaki & M., 2015).

2.2.14 Typescript

Salah satu bahasa pemrograman yang sedang populer untuk mengembangkan aplikasi web adalah TypeScript (Thu et al., 2021). TypeScript berkembang sebagai alternatif dari JavaScript yang menerapkan pengecekan tipe statis. Telah banyak perkembangan dan penambahan fitur serta sintax baru dalam bahasa ini. Namun, TypeScript tidak memiliki spesifikasi formal (Scarsbrook et al., 2023). Perkembangan dari TypeScript diikuti dengan banyaknya pengguna yang menggunakan bahasa ini sebagai bahasa untuk mengembangkan aplikasi website. Akan tetapi, tidak semua fitur yang ada di TypeScript diadopsi oleh pengguna. Pengadopsian TypeScript ini juga mempunyai tantangan (Thu et al., 2021) Beberapa fitur bahasa baru jarang diadopsi oleh proyek-proyek yang ada. Hal ini dikarenakan adopsi fitur baru dalam bahasa TypeScript membutuhkan waktu yang cukup lama. Sebuah proyek dapat memperbarui versi TypeScript tanpa mengubah kode mereka sama sekali, sehingga tanpa mengadopsi fitur baru. Sehingga, mengadopsi fitur baru mungkin memerlukan adopsi versi TypeScript yang baru, tetapi tidak se-

baliknya. Hal ini bisa dilihat dari jumlah repositori yang mengadopsi TypeScript terbaru yang berjumlah kurang lebih 1/3 dari mayoritas repositori setelah adanya versi baru dari TypeScript itu sendiri (Scarsbrook et al., 2023).

BAB III

METODOLOGI

Penelitian ini dilaksanakan sesuai dengan desain sistem berikut ini beserta implementasinya. Desain sistem merupakan konsep dari pembuatan dan perencangan infrastruktur dan kemudian diwujudkan dalam bentuk blok-blok alur yang harus dikerjakan.

3.1 Deskripsi Sistem

Tugas akhir ini merupakan penelitian yang menggunakan teknologi deep learning agar dapat mendeteksi keberadaan rongga udara di dalam beton. Secara umum penelitian kali ini akan menggunakan desain sistem sesuai dengan Gambar 3.1.



Gambar 3.1: Blok Diagram Penelitian

3.1.1 *Generate Data*

Data sinyal GPR yang didapatkan merupakan data hasil simulasi menggunakan software gprMax. Setiap sinyal yang dihasilkan berbentuk B-Scan 2 dimensi. Pada penelitian kali ini, dihasilkan berjumlah 4000 data sinyal dengan 2000 data sinyal untuk beton berisi rongga udara dan rebar dan 2000 data sinyal untuk beton berisi rebar. 2000 File input untuk generate data sinyal berisi rongga udara dihasilkan menggunakan kode python untuk mempersingkat waktu pembuatan file input generate data. Kode python tersebut menghasilkan file input yang menghasilkan data sinyal untuk rongga udara secara acak sesuai dengan ketentuan yang telah dijabarkan pada BAB I. Data sinyal gprMax yang berisi rongga udara akan digenerate dengan menggunakan GPU secara online pada platform vast.ai karena keterbatasan hardware yang dimiliki oleh penulis. Berikut ini adalah spesifikasi dari GPU yang digunakan sebagai mana pada tabel 3.1.

Brand	NVIDIA
Type	RTX 3060
Number of GPU	1
Disk Space	20
OS	Ubuntu 20.04
API	CUDA

Tabel 3.1: spesifikasi GPU

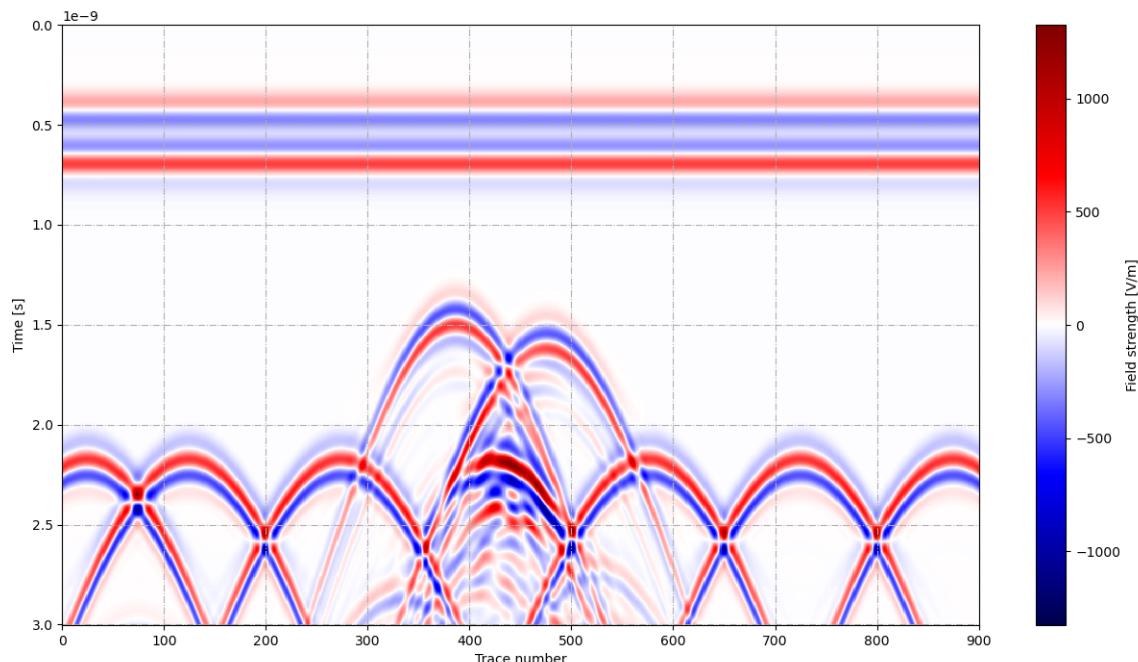
untuk data sinyal beton dengan rebar tanpa rongga udara, dihasilkan dengan cara yang sedikit berbeda. Pada awalnya, data ini digenerate dengan cara yang sama dengan data berisi rongga udara, akan tetapi setelah beberapa data, gambar sinyal yang dihasilkan memiliki banyak kemiripan. Sehingga data ini digenerate dengan cara lain. Pertama digenerate dua gambar sinyal

yakni sinyal beton dengan rebar terstruktur dan gambar sinyal beton dengan rebar yang memiliki sedikit perubahan posisi, tepatnya satu rebar memiliki jarak antar rebar 10 cm. Dua gambar tersebut disatukan secara menyamping dan diproses menggunakan python untuk merubah posisi sinyal pada frame satu gambar dari dua gambar yang telah disatukan dengan perpindahan posisi sinyal secara horizontal dan vertikal per beberapa pixel.

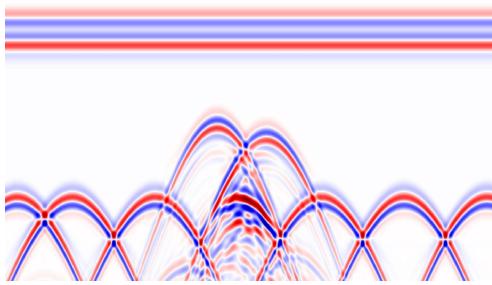
3.1.2 Preprocessing Data

Sinyal gprMax yang memiliki rongga udara dalam bentuk .out yang telah digenerate, akan dilakukan proses untuk menghilangkan atribut bawaan gambar dari gprMax sehingga gambar hanya akan menampilkan gambar sinyal dan memotong ukuran gambar tersebut. Untuk gambar sinyal yang tidak memiliki rongga udara, gambar tersebut akan di format ulang agar dapat diproses karena memiliki format gambar yang berbeda dengan gambar hasil *generate gprMax*. Data simulasi sinyal B-Scan gprMax yang telah dipotong akan dilakukan proses binarization yang akan membuat gambar menjadi berwarna hitam putih untuk mengurangi ukuran file dari data tersebut dan model hasil training. Berikut ini adalah gambar sinyal sebelum dan sesudah diproses seperti yang dapat dilihat pada Gambar 3.2, gambar 3.3, dan gambar 3.4

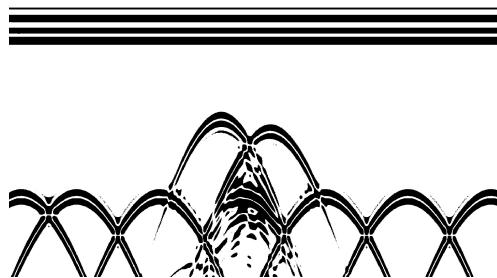
Berikut ini adalah contoh hasil generate data sinyal gprMax yang dapat dilihat pada Gambar 3.2 dan 3.3.



Gambar 3.2: Data Sinyal gprMax Setelah Dibersihkan



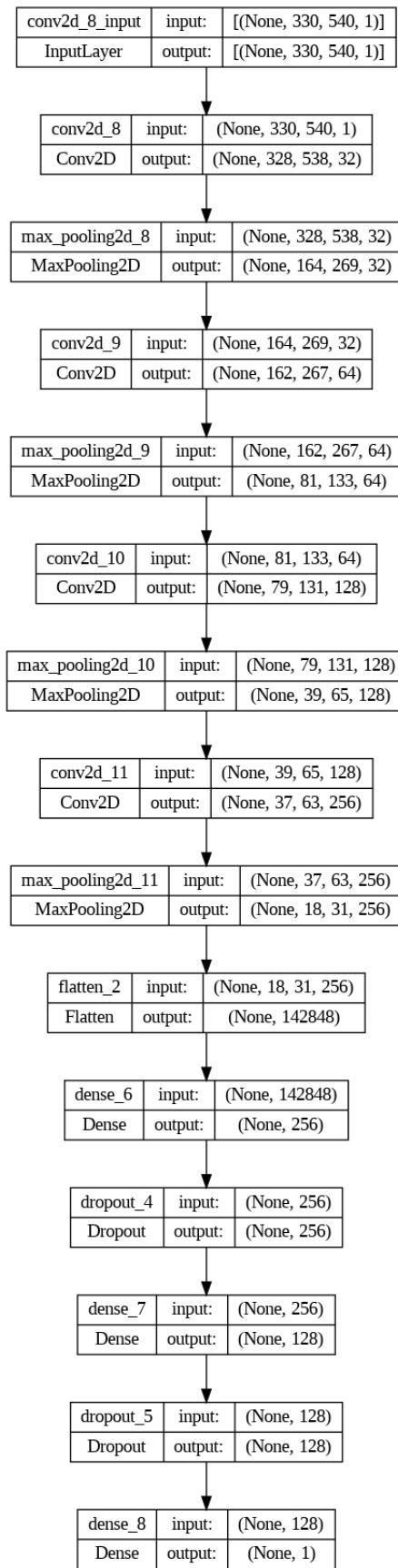
Gambar 3.3: Data Sinyal gprMax Setelah Dibersihkan



Gambar 3.4: Data Sinyal gprMax Setelah Dibersihkan

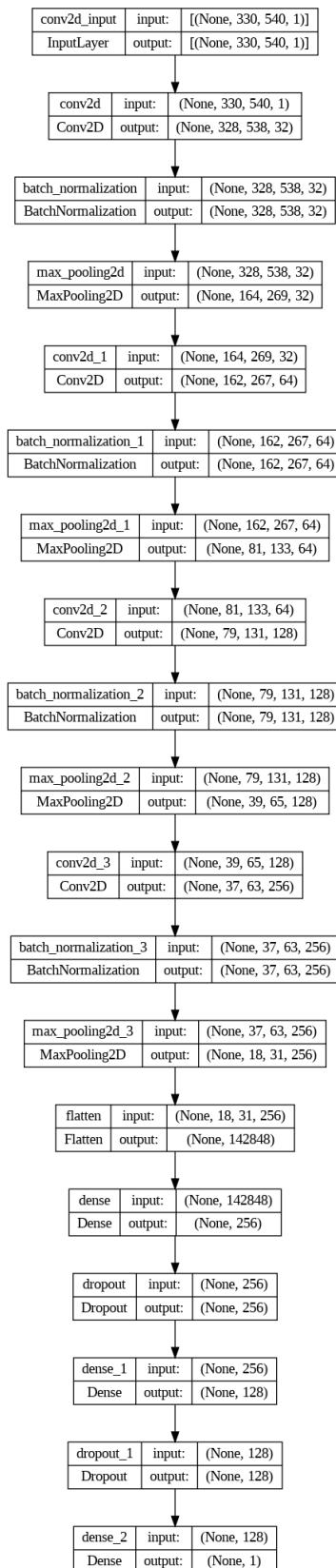
3.1.3 *Training Data*

Dataset yang telah dihasilkan terbagi menjadi dua kelas yakni 'airgap' dan 'noarigap'. Data tersebut akan dilakukan proses training untuk klasifikasi. Terdapat dua metode klasifikasi yang digunakan yakni CNN 2-Dimensi dan Roboflow. Pada CNN 2D, diterapkan berbagai macam pembagian dataset untuk training, validasi, dan testing. Pembagian dataset yang akan diujikan untuk training, validasi, dan testing antara lain 70/20/10, 70/15/15, dan 60/20/20. Model CNN 2D yang akan digunakan untuk pembagian dataset dapat dilihat pada tabel 3.5.

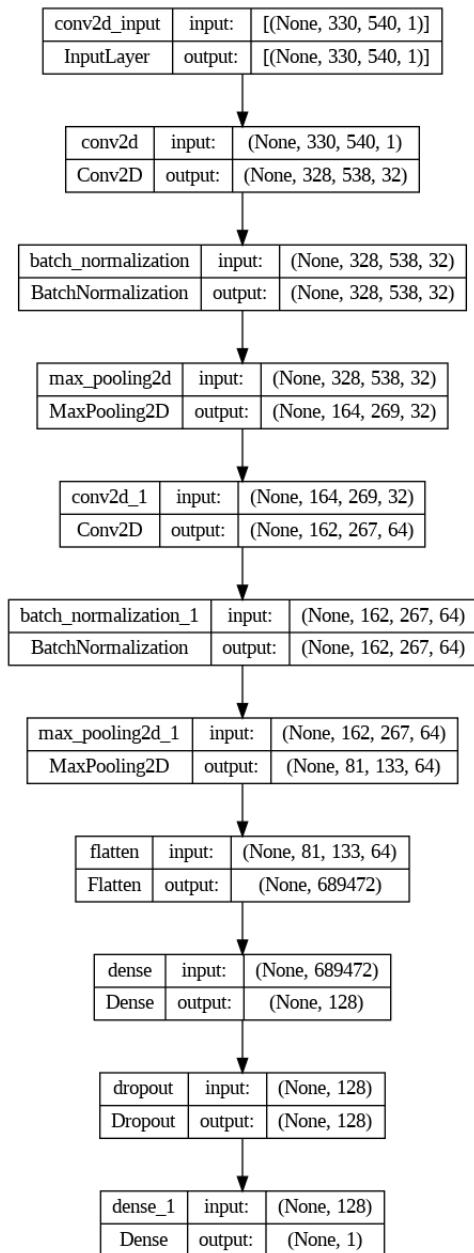


Gambar 3.5: Arsitektur Model CNN 2D Pertama

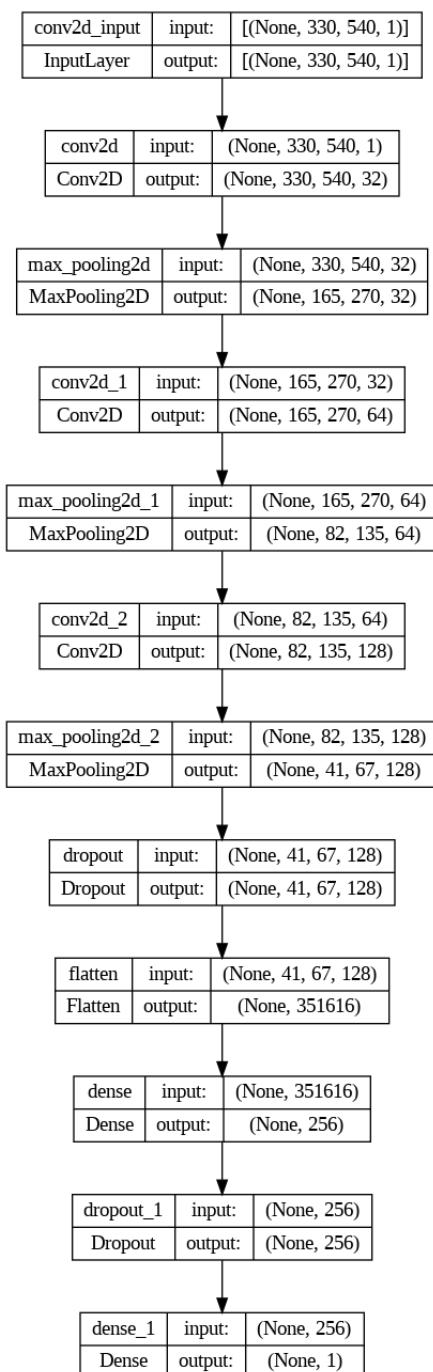
Setelah didapatkan hasil pembagian data yang baik, pembagian data tersebut nantinya akan diterapkan pada struktur CNN 2-Dimensi yang divariasi untuk menemukan bagaimana struktur CNN 2-Dimensi yang optimal pada percobaan keempat, kelima, keenam dan ketujuh. Variasi model CNN 2D yang akan digunakan dapat dilihat pada tabel 3.6, tabel 3.7, tabel 3.8, dan tabel 3.9. CNN 2-Dimensi yang digunakan pada percobaan ini didapatkan dengan bantuan ChatGPT. Sementara itu, untuk model CNN 2D yang digunakan pada percobaan ketujuh, diambil dari penelitian yang berjudul "*Classification of soil types from GPR B Scans using deep learning techniques*" (Barkataki et al., 2021).



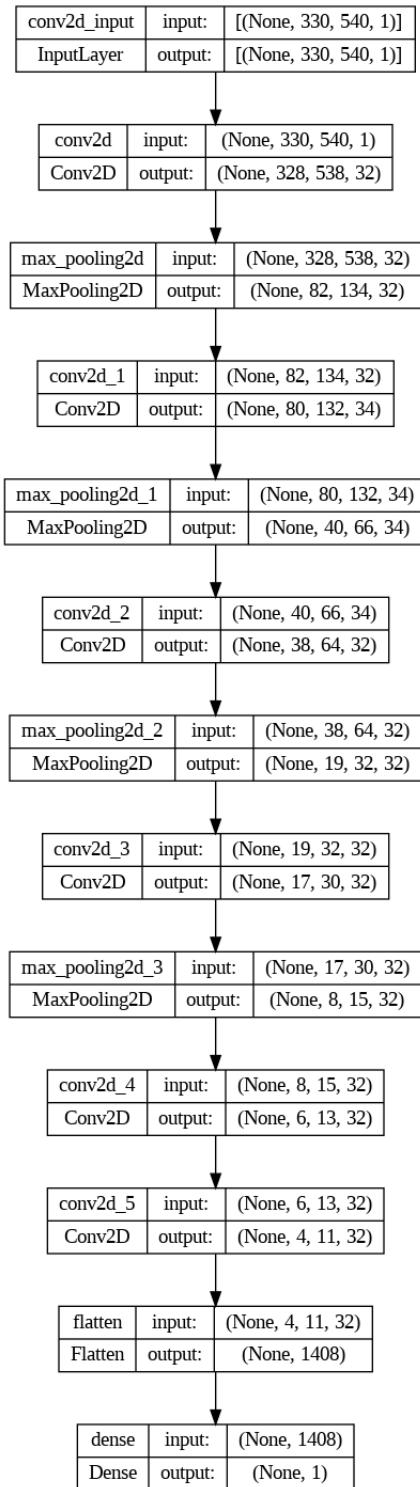
Gambar 3.6: Arsitektur Model CNN 2D Kedua



Gambar 3.7: Arsitektur Model CNN 2D Ketiga



Gambar 3.8: Arsitektur Model CNN 2D Keempat



Gambar 3.9: Arsitektur Model CNN 2D Kelima

Sementara pada Roboflow, proses klasifikasi cukup hanya dengan menetapkan gambar pada salah satu kelas. Kemudian dilakukan proses augmentasi pada data tersebut antara lain: flip (horizontal and vertical), rotate (90 degrees clockwise and counter-clockwise), crop (20% maximum zoom), noise (up to 0.93% pixels), dan blur (up to 1 px). Pada Roboflow, pembagian kelas awal adalah 70/20/10 untuk training, validation, dan testing. Persentase tersebut berubah

setelah ditambahkan proses augmentasi menjadi 88/8/4.

3.1.4 Pengujian Klasifikasi

Pengujian klasifikasi dilakukan dengan melakukan test pada data validasi yang kemudian hasilnya akan membentuk confussion matrix. Pengujian klasifikasi juga akan dilakukan terhadap data yang dipersiapkan untuk testing diawal proses training. Pada platform, roboflow, pengujian klasifikasi cukup dilakukan dengan mengguggah gambar yang hendak diuji, berikutnya platform Roboflow akan otomatis mengklasifikasikan gambar kelas tersebut.

3.1.5 Deteki YOLOv9

YOLO yang digunakan pada training kali ini adalah YOLOv9. Versi ini dipilih karena YOLOv9 masih tergolong baru sehingga belum banyak digunakan oleh banyak orang. Model YOLOv9 ini diambil dari ultralytics berdasarkan penelitian yang berjudul "*YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information*" (C.-Y. Wang et al., 2024). Berikut ini adalah rincian dari arsitektur YOLOv9 yang digunakan pada penelitian ini:

Index	Module	Route	Filters	Depth	Size	Stride
0	Conv	-	64	-	3	2
1	Conv	-	128	-	3	2
2	CSP-ELAN	0	256, 128	2	1	1
3	DOWN	1	256	-	-	-
4	CSP-ELAN	2	512, 256	2	1	1
5	DOWN	4	512	-	-	-
6	CSP-ELAN	5	512, 512	2	1	1
7	DOWN	6	512	-	-	-
8	CSP-ELAN	7	512, 512, 256	2	1	1
9	SPP-ELAN	8	512, 256, 256, 512	3	1	1
10	Up	9	512	-	-	-
11	Concat	10, 6	1024	-	-	-
12	CSP-ELAN	11	512, 512, 256	2	1	1
13	Up	12	512	-	-	-
14	Concat	13, 4	1024	-	-	-
15	CSP-ELAN	14	256, 256, 128	2	1	1
16	Down	15	256	-	-	-
17	Concat	16, 12	768	-	-	-
18	CSP-ELAN	17	512, 256, 128	1	1	1
19	Down	18	512	-	-	-
20	Concat	19, 9	1024	-	-	-
21	CSP-ELAN	20	512, 512, 256	2	1	1
22	Predict	15, 18, 21	-	-	-	-

Tabel 3.2: Konfigurasi Jaringan YOLOv9

Hyper Parameter	Value
epochs	500
optimizer	SGD
initial learning rate	0.01
finish learning rate	0.0001
learning rate decay	linear
momentum	0.937
weight decay	0.0005
warm-up epochs	3
warm-up momentum	0.8
warm-up bias learning rate	0.1
box loss gain	7.5
class loss gain	0.5
DFL loss gain	1.5
HSV saturation augmentation	0.7
HSV value augmentation	0.4
scale augmentation	0.9
mosaic augmentation	1.0
MixUp augmentation	1.5
copy & paste augmentation	0.3
close mosaic epochs	15

Tabel 3.3: Pengaturan Hyperparameter YOLOv9

3.1.6 Hasil

Hasil dari training CNN 2D nantinya akan menghasilkan grafik training, tingkat akurasi, confusion matriks, dan f1 score yang nantinya akan digunakan sebagai pertimbangan untuk mengetahui keberhasilan dari model yang telah di training. Sementara untuk hasil training dari Roboflow hanya akan menghasilkan grafik training dan confusion matriks yang disediakan secara gratis dari platform website tersebut. Hasil training dari YOLOv9 akan menghasilkan berbagai macam metriks termasuk akurasi dan confusion matriks.

3.2 Implementasi Deteksi

Pada penelitian ini dikembangkan suatu website yang dapat menerima gambar sinyal GPR maupun gprMax dan melakukan klasifikasi serta deteksi rongga udara pada gambar tersebut. website ini dibangun menggunakan framework Next.js karena memiliki fungsi yang sama dengan React dan proses pemrograman yang lebih mudah. Pengguna dapat menggunakan website ini dengan mengikuti alur dari flowchart yang dapat dilihat pada gambar 3.10.



Gambar 3.10: Flowchart Penggunaan Website

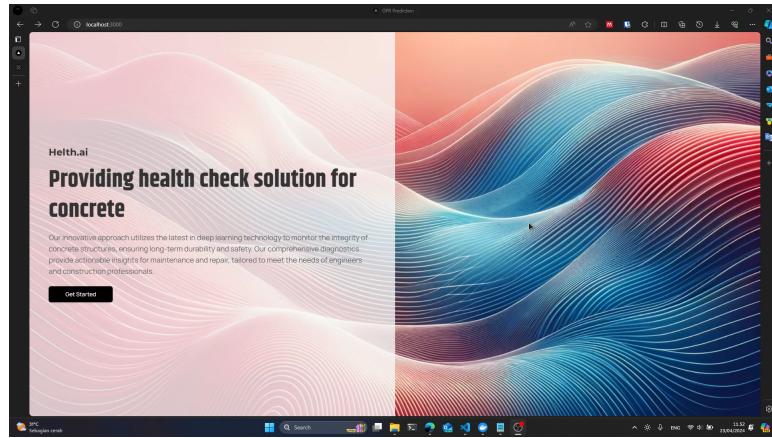
Berdasarkan flowchart diatas, pengguna dapat mengunjungi laman website <https://airgap-detect.vercel.app/> dan memasuki laman utama. Kemudian user dapat memasuki laman deteksi untuk mengunggah gambar sinyal yang ingin dideteksi keberadaan rongga udaranya. Peng-

guna dapat memilih model yang ingin digunakan antara model CNN 2D atau Roboflow. Setelah memilih model, pengguna dapat memulai proses klasifikasi dan deteksi. Hasil klasifikasi dan deteksi akan ditampilkan diakhir proses dengan gambar sinyal yang tidak memiliki rongga udara akan diberi filter merah dan yang memiliki rongga udara akan memiliki filter hijau dengan bounding box dari YOLOv9. Hasil tersebut dapat didownload dalam bentuk zip. Website ini dideploy pada vercel dan dihosting dengan biznet. Pada pembuatan website, proses terbagi menjadi dua tahap utama yakni frontend dan backend. Berikut ini adalah flowchart tentang bagaimana website ini bekerja sebagaimana dapat dilihat pada gambar 3.11.

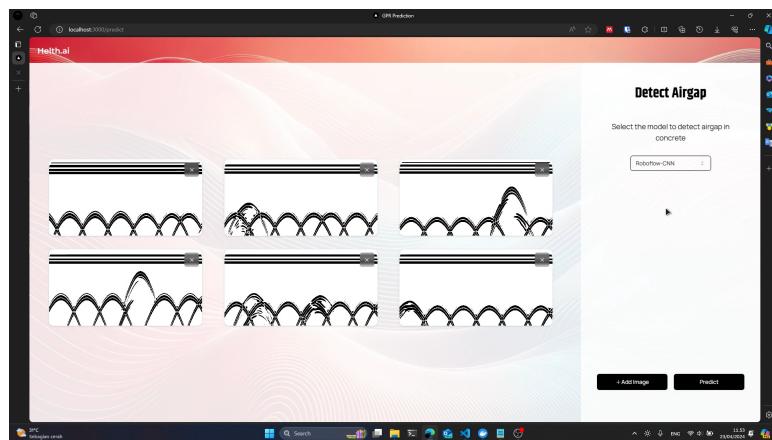


Gambar 3.11: Flowchart Website

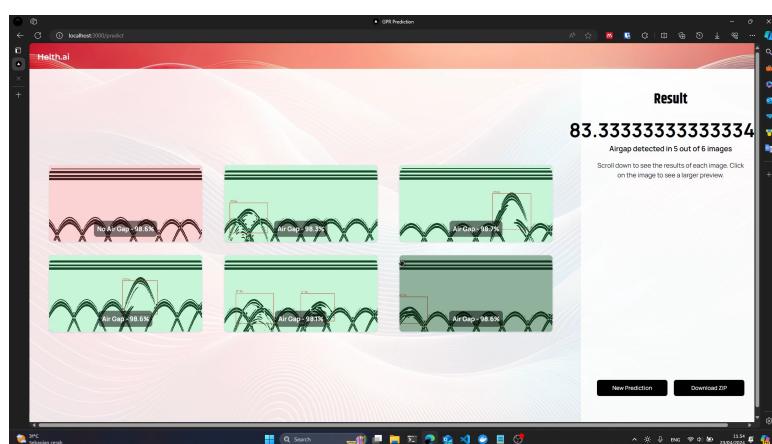
Tahap pertama yang perlu dilakukan adalah pengerjaan bagian frontend. Langkah pertama yang dilakukan adalah melakukan desain dari website tersebut. Proses desain dilakukan pada platform figma. Setelah desain selesai, langkah selanjutnya adalah melakukan implementasi desain tersebut ke dalam kode program. Proses implementasi ini dilakukan dengan menggunakan framework Next.js. Pada tahap ini, dilakukan pembuatan beberapa halaman website yang terdiri dari halaman utama, halaman upload gambar, halaman hasil klasifikasi, dan halaman hasil deteksi. Backend dibuat setelah frontend selesai dimana diperlukan pembuatan API agar frontend dapat bekerja. API yang digunakan pada pembuatan backend adalah FastAPI. FastAPI sendiri digunakan agar website dapat memproses berbagai lebih dari satu user request pada saat yang bersamaan. API ini nantinya akan terhubung dengan Redis sebagai platform database dan message broker yang mana akan menjadi perantara bagi frontend dengan virtual machine yang digunakan untuk melakukan klasifikasi dan deteksi. ketika pengguna mengunggah gambar, maka gambar akan disimpan sementara pada database Redis. Kemudian Redis akan mengirim pesan pada virtual machine sebagai worker untuk melakukan klasifikasi dan deteksi. Sebelumnya, model CNN 2-Dimensi dan YOLOv9 tidak dimasukkan ke dalam virtual machine, melainkan ke dalam Tensorflow Serving. Hal tersebut bertujuan agar model dapat diakses oleh lebih dari satu instance saat terdapat lebih dari satu request dan mengurangi jumlah memori yang akan digunakan oleh setiap instance. Model dari YOLOv9 perlu dikonversi terlebih dahulu dari pytorch ke keras (pb) agar bisa digunakan. Untuk model dari Roboflow, tidak perlu diletakkan pada Tensorflow Serving karena cukup dengan memanggil endpoint saja. Setelah model berhasil diakses, maka model akan melakukan klasifikasi dilanjut dengan deteksi YOLOv9. Hasil klasifikasi dan deteksi tersebut akan dikirimkan ke database untuk dikirimkan kembali ke user serta menghapus gambar yang telah diunggah. Berikut ini adalah contoh hasil dari website yang telah dibuat:



Gambar 3.12: Gambar Platform Website 1



Gambar 3.13: Gambar Platform Website 2



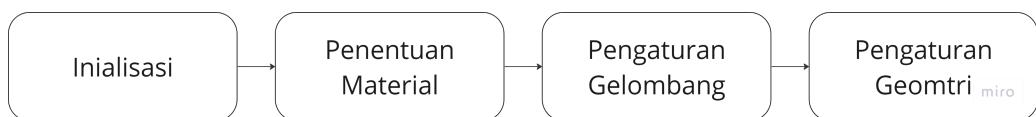
Gambar 3.14: Gambar Platform Website 3

3.3 Kode Program

Pada sub bab ini akan dijabarkan kode program yang digunakan pada penelitian ini.

3.3.1 File Input Generate Data

Gambar sinyal dihasilkan dengan memasukkan file input yang berisi parameter-parameter yang diperlukan untuk menghasilkan gambar sinyal. File input berisi parameter-parameter yang diperlukan untuk menghasilkan gambar sinyal. File input dihasilkan secara otomatis menggunakan python karena jumlahnya yang banyak. Berikut ini adalah flowchart dari file input generate data yang dapat dilihat pada gambar 3.15.



Gambar 3.15: Input File Generate Data

Algorithm 1 Simulation of B-scan from Rebar in Concrete

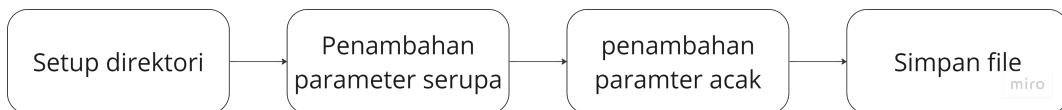
- 1: Title: "B-scan from rebar in concrete"
 - 2: Define domain: $1.0 \text{ m} \times 0.3 \text{ m} \times 0.001 \text{ m}$
 - 3: Cell size: $0.001 \text{ m} \times 0.001 \text{ m} \times 0.001 \text{ m}$
 - 4: Time window: $3 \times 10^{-9} \text{ s}$
 - 5: Define material concrete: $\epsilon = 7.0, \sigma = 0.0, \mu = 1.0, \chi = 0.0$
 - 6: Define material steel: $\epsilon = 12.0, \sigma = 1.0e6, \mu = 1.0, \chi = 0.0$
 - 7: Create waveform: Ricker, peak frequency 4.5×10^9 , name "my_ricker"
 - 8: Source steps: $0.001 \text{ m} \times 0 \text{ m} \times 0 \text{ m}$
 - 9: Receiver steps: $0.001 \text{ m} \times 0 \text{ m} \times 0 \text{ m}$
 - 10: Place box: Start (0.0, 0.0, 0.0), End (1.0, 0.203, 0.001), Material concrete
 - 11: Add Hertzian dipole: Position (0.010, 0.203, 0), Waveform "my_ricker"
 - 12: Place receiver: Position (0.050, 0.203, 0)
 - 13: Place cylinders: Positions and dimensions specified for steel
 - 14: Place cylinders: Positions and dimensions specified for free space
-

Pada awal program, dituliskan judul dari gelombang yang akan dihasilkan yang mana bersifat opsional. Langkah selanjutnya adalah ditentukan ukuran media yang akan dijadikan sebagai simulasi. Pada program juga diinputkan kode untuk menentukan waktu yang akan diamati pada saat berlangsungnya simulasi dan menyusun tingkat satuan terkecil untuk file input. Pada gprMax, terdapat dua material bawaan yakni udara dan *pec* yang merupakan konduktor material sempurna. Pada program ini, ditentukan material yang akan digunakan pada simulasi seperti rebar dan beton yang mana parameteranya diambil berdasarkan penelitian terdahulu yang menggunakan gprMax.

Langkah selanjutnya diatur tipe gelombang yakni Ricker wavelet yang mana merupakan gelombang yang paling sering digunakan pada simulasi GPR. Diatur pula letak *transmitter* dan *receiver* yang akan digunakan pada simulasi yang ketinggiannya sama dengan ketinggian/ketinggian/ketinggian dari beton yang akan dibuat. Pergerakan dari *transmitter* dan *receiver* diatur untuk mampu berpindah posisi hingga sekecil 1 mm per langkah. Hal ini dilakukan untuk mendapatkan data sinyal yang lebih banyak dan akurat. Terakhir, ditentukan objek yang nantinya akan ada didalam simulasi seperti beton, rebar, dan rongga udara yang mana masing-masing objek memiliki parameter tersendiri. Pada program ini, beton menggunakan parameter #box karena medianya kotak, rebar menggunakan parameter #cylinder karena medianya silinder, dan rongga udara menggunakan parameter #cylinder rongga udara memiliki bentuk yang abstrak.

3.3.2 Generate File Input gprMax dengan Rongga Udara

Program ini dirancang untuk melakukan proses generate file input gprMax dengan rongga udara. Hal tersebut dilakukan karena proses pembuatan file input gprMax dengan rongga udara dalam jumlah besar membutuhkan waktu yang lama. Program ini dibuat dengan menggunakan Python, berikut ini adalah flowchart tentang bagaimana program ini bekerja sebagaimana dapat dilihat pada gambar 3.17.



Gambar 3.16: Flowchart Setup GPU

Algorithm 2 Generate gprMax Simulation Files

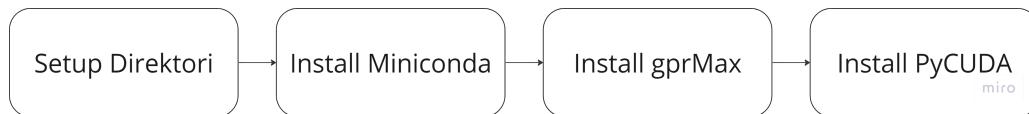
```
1: function GENERATEAIRGAPCYLINDERS(file, boxHeight)
2:   xMin, xMax  $\leftarrow$  0.05, 0.95
3:   yMin, yMax  $\leftarrow$  0.090, boxHeight – 0.05
4:   airgapCount  $\leftarrow$  Random integer from 1 to 2
5:   for i  $\leftarrow$  1 to airgapCount do
6:     cylinderCount  $\leftarrow$  Random integer from 3 to 5
7:     xPosition  $\leftarrow$  Random float between xMin and xMax
8:     yPosition  $\leftarrow$  Random float between yMin and yMax
9:     for j  $\leftarrow$  1 to cylinderCount do
10:      radius  $\leftarrow$  Random float from 0.005 to 0.0175
11:      Write cylinder data to file
12:      Adjust xPosition and yPosition within bounds
13:    end for
14:  end for
15: end function
16:
17: function GENERATEGPRMAXFILES
18:   Create directory "airgap"
19:   totalFiles  $\leftarrow$  2000
20:   for fileNumber  $\leftarrow$  1 to totalFiles do
21:     boxHeight  $\leftarrow$  Random float between 0.190 and 0.210
22:     Create file with initial configuration
23:     Write steel cylinder configuration to file
24:     GENERATEAIRGAPCYLINDERS(file, boxHeight)
25:     Print file creation message
26:   end for
27: end function
```

Pada langkah awal, dilakukan inisialisasi dengan membuat folder baru sebagai tempat disiapkannya hasil generate file input gprMax. Kemudian, dilakukan perulangan untuk membuat file text yang berisi struktur dari sinyal yang akan digenerate. Program akan menambahkan parameter-parameter yang akan dimiliki oleh semua file input (parameter serupa) ter-

lebih dahulu di awal seperti judul file, ukuran domain, rentang waktu, tingkat kerincian, material, jenis gelombang, dan pergerakan dari transmitter dan receiver gelombang. Setelah itu, dilakukan penambahan baris kode dengan parameter yang sifatnya akan divariasi secara acak. Parameter awal adalah dimensi beton yang berukuran panjang 1 meter variasi ketebalan antara 19 cm hingga 21 cm. Ketebalan dari beton ini akan menentukan posisi ketinggian dari transmitter dan receiver yang akan ditambahkan setelahnya. Selanjutnya, akan ditambahkan parameter rebar yang memiliki material besi dengan diameter 14 mm dengan setiap rebar akan memiliki ketinggian yang sama. Ketinggian dan posisi secara horizontal ini akan berubah setiap filenya dengan rebar memiliki variasi ketinggian antara 9 cm hingga 11 cm sementara posisi horizontal rebar akan bertambah sebanyak 1 mm setiap filenya yang mana apabila posisi rebar secara horizontal melebihi panjang rebar, maka kelebihan posisi tersebut akan ditempatkan pada awal posisi beton secara horizontal. Terakhir adalah parameter rongga udara yang akan ditempatkan secara acak pada beton. Rongga udara bisa terdiri dari satu atau lebih silinder dengan diameter antara 2 cm hingga 7 cm yang letak ketinggiannya akan berada diantara 5 cm hingga titik tertinggi beton dikurangi 5 cm. Posisi rongga udara secara horizontal akan diletakkan secara acak. Setelah semua parameter ditambahkan, file akan disimpan dalam format .in.

3.3.3 Setup GPU untuk Generate Data

Dalam penggunaan GPU untuk mempercepat proses generate data, diperlukan pengaturan GPU terlebih dahulu agar GPU dapat digunakan. Berikut ini adalah flowchart dari setup GPU untuk generate data yang dapat dilihat pada gambar 3.17.



Gambar 3.17: Flowchart Setup GPU

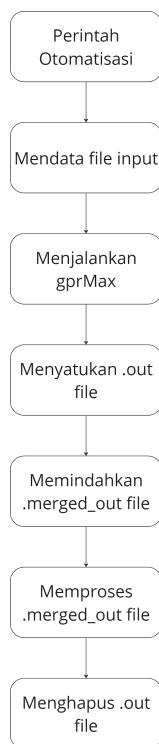
Pada awal pengaturan GPU, GPU diakses melalui ssh yang terhubung dengan ssh key milik PC penulis. Setelah terhubung, dibuat folder baru untuk instalasi miniconda. Instalasi miniconda dilakukan dengan melakukan replikasi repositori github dan menjalankan file bash yang ada pada repositori tersebut. Setelah instalasi selesai, dilakukan pembaruan terhadap versi conda dan instalasi Git untuk proses instalasi gprMax. Proses instalasi gprMax dilakukan dengan melakukan replikasi repositori Github. Sebelum menjalankan instalasi, dibuat environment baru untuk gprMax dan mengaktifkannya. Setelah aktif, dijalankan kode python dari repositori tersebut untuk melakukan setup gprMax. Langkah terakhir adalah melakukan instalasi PyCUDA agar proses generate data dapat menggunakan GPU.

3.3.4 Generate Data Sinyal GPR

Program ini dirancang untuk melakukan proses generate sinyal hasil simulasi file input untuk gprMax. Program ini dibuat dengan menggunakan python dan menggunakan library gprMax untuk menghasilkan data sinyal. Program ini akan menghasilkan data sinyal berupa gambar B-Scan 2 dimensi yang kemudian akan dijadikan sebagai data training untuk klasifikasi. Berikut ini adalah flowchart dari generate data yang dapat dilihat pada gambar 3.18 dan 3.19.

Algorithm 3 Setup Miniconda and Configure gprMax Environment

- 1: Echo "Creating Miniconda directory..."
 - 2: Execute "mkdir -p ~/miniconda3"
 - 3: Echo "Downloading Miniconda installer..."
 - 4: Execute "wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3/miniconda.sh"
 - 5: Echo "Installing Miniconda..."
 - 6: Execute "bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3"
 - 7: Echo "Removing Miniconda installer..."
 - 8: Execute "rm -rf ~/miniconda3/miniconda.sh"
 - 9: Echo "Initializing Conda for bash..."
 - 10: Execute "~/miniconda3/bin/conda init bash"
 - 11: Execute "source ./bashrc"
 - 12: Echo "Updating Conda..."
 - 13: Execute "conda update conda -y"
 - 14: Echo "Installing Git..."
 - 15: Execute "conda install git -y"
 - 16: Echo "Cloning gprMax repository..."
 - 17: Execute "git clone https://github.com/MaulanaGilang/gprMax.git"
 - 18: Echo "Changing directory to gprMax..."
 - 19: Execute "cd gprMax"
 - 20: Echo "Creating Conda environment from file..."
 - 21: Execute "conda env create -f conda_env.yml"
 - 22: Echo "Activating gprMax environment..."
 - 23: Execute "conda activate gprMax"
 - 24: Echo "Building gprMax..."
 - 25: Execute "python setup.py build"
 - 26: Echo "Installing gprMax..."
 - 27: Execute "python setup.py install"
 - 28: Echo "Installing PyCUDA..."
 - 29: Execute "pip install pycuda"
 - 30: Echo "Setup complete. gprMax is ready to use."
-



Gambar 3.18: Kode Otomatisasi Generate Data

Pada program ini, pertama-tama dilakukan import library yang dibutuhkan. Selanjutnya, dilakukan pembacaan file input yang telah dibuat sebelumnya. File input tersebut berisi parameter-parameter yang diperlukan untuk menghasilkan gambar sinyal. Setelah itu, program akan melakukan iterasi sebanyak jumlah data yang diinginkan. Pada setiap iterasi, program akan menghasilkan gambar sinyal berdasarkan parameter yang ada pada file input dengan menjalankan software gprMax. Saat setelah selesai generate, setiap file input akan menghasilkan file output berupa potongan-potongan sinyal dengan format .out yang akan disatukan menjadi satu file dengan format .merged.out. File ini kemudian akan dipindahkan ke folder yang telah disediakan dan akan menghapus semua file awal berformat .out untuk mengurangi memori yang digunakan oleh GPU. File yang dipindahkan tersebut akan dibaca dan diubah menjadi gambar sinyal menggunakan python. Gambar sinyal yang telah dihasilkan kemudian akan diproses agar parameter bawaan gambar dari gprMax hilang dan hanya menampilkan gambar sinyal saja serta memotong ukuran gambar tersebut.

Algorithm 4 Process GPRMax Output Files

```
1: function NATURALKEYS(text)
2:   Parse text and return parts as integers if they are digits or as text otherwise.
3: end function
4:
5: function LISTINFILESRECURSIVE(directory)
6:   Recursively list all .in files in the directory.
7:   Return a dictionary of filenames without extension as keys and paths as values.
8: end function
9:
10: function RUNGPRMAX(file_path, n, use_gpu)
11:   Construct and run a command to execute gprMax with optional GPU usage.
12: end function
13:
14: function MERGEOUTPUTFILES(directory, file_without_ext)
15:   Merge output files based on the .in filename and delete the original .out files.
16: end function
17:
18: function ENSUREDIRECTORYEXISTS(directory)
19:   Create the directory if it does not exist.
20: end function
21:
22: function MOVEOUTPUTFILE(source_path, dest_directory)
23:   Move the output file to a specified directory, ensuring the directory exists.
24:   Return the new path.
25: end function
26:
27: function PROCESSFILE(file_path, rxnumber, rxcomponent, non_greyscale_dir, greyscale_dir)
28:   Generate and save color and cropped grayscale images from the output data of gprMax.
29: end function
30:
31: procedure MAIN(args)
32:   Parse input directory, indices and settings from command-line arguments.
33:   Initialize directories and prepare file processing.
34:   for each file to process within index range do
35:     Run gprMax simulation.
36:     if merge is enabled then
37:       Merge output files.
38:       Move and process merged file.
39:     end if
40:     Increment processed files count.
41:   end for
42:   Clean up any remaining processed files.
43: end procedure
```



Gambar 3.19: Input File Generate Data

Algorithm 5 Image Manipulation and Transformation

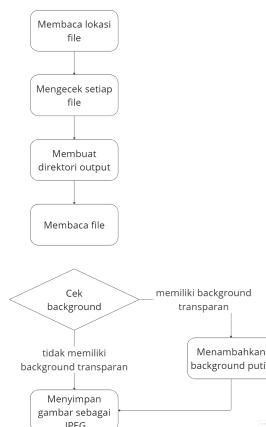
- 1: Load line and wave images from 'noairgap' directory.
- 2: Create a new transparent RGBA image 'result_img' with dimensions 1860x1160.
- 3: Paste the line image onto 'result_img' at position (0,0).
- 4: **function** TRANSFORMANDSAVE(wave_img, result_img, shift_x, shift_y, crop_size, file_name)
 - 5: Create a copy of 'result_img' to 'temp_img'.
 - 6: Paste 'wave_img' onto 'temp_img' with specified shifts 'shift_x' and 'shift_y'.
 - 7: Repaste the line image onto 'temp_img' at position (0,0).
 - 8: Crop 'temp_img' according to 'crop_size'.
 - 9: Save the cropped image to 'file_name'.
- 10: **end function**
- 11: Calculate number of images to be generated.
- 12: Initialize a counter to zero.
- 13: **for** each horizontal shift from -wave width + 1860 to 0 step by 5 **do**
 - 14: **for** each vertical shift from 0 to 100 step by 20 **do**
 - 15: Construct file name for output image.
 - 16: TRANSFORMANDSAVE(wave_img, result_img, horizontal shift, vertical shift, crop size, file name)
 - 17: Increment counter by one.
 - 18: **end for**
 - 19: **end for**

Untuk program generate data sinyal gprMax yang tidak memiliki rongga udara, data ini dihasilkan dengan cara yang berbeda karena tidak banyak perbedaan pada setiap gambar sinyal yang dihasilkan. Untuk menjalan program ini, dibutuhkan gambar sinyal tanpa rongga udara hasil generate gprMax dengan gambar pertama adalah gambar normal dan gambar kedua adalah gambar sinyal dengan rebar yang memiliki sebuah rebar yang tidak berjarak 15 cm. Selanjutnya, kedua gambar tersebut disatukan secara horizontal dan dipotong bagian sinyalnya. Se-

lanjutnya, diambil gambar garis pada salah satu gambar dengan cara dipotong. Dua gambar tersebut nantinya akan menjadi file input untuk augmentasi tersebut. Pada program, akan dibuat sebuah gambar baru dengan ukuran 1860x1160 piksel dengan latar belakang yang transparan. Kemudian, akan ditambahkan gambar garis pada bagian atas gambar tersebut. Selanjutnya ditambahkan gambar sinyal dibagian bawah gambar garis dengan ketentuan setiap gambar sinyal tersebut akan dipindahkan posisinya per gambar secara horizontal sebanyak 5 piksel hingga menyentuh batas gambar sinyal secara horizontal dan secara vertikal ke bawah 100 piksel dengan perpindahan 20 piksel per gambar. Gambar sinyal dan garis akan menimpa gambar latar belakang sehingga tercipta gambar sinyal baru. Gambar sinyal yang telah dihasilkan nantinya akan disimpan pada folder yang telah disediakan.

3.3.5 Preprocessing Data

Program berikut ini bertujuan untuk melakukan *preprocessing* data sinyal yang telah dihasilkan dari proses generate data. Program ini dibuat menggunakan python. Berikut ini adalah flowchart dari *preprocessing* data yang dapat dilihat pada gambar 3.20 dan 3.21

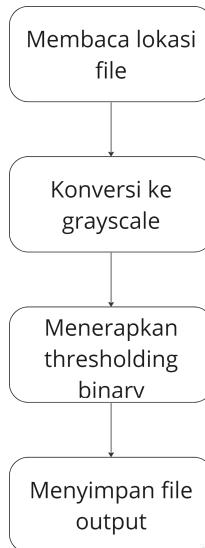


Gambar 3.20: Mengubah Format Gambar Sinyal

Flowchart diatas menjelaskan proses mengubah format gambar sinyal yang dihasilkan dari proses augmentasi python untuk sinyal tanpa rongga udara. Pada program ini, pertama-tama dilakukan pembacaan lokasi file yang ingin diubah formatnya. Kemudian dilakukan import library yang dibutuhkan. Selanjutnya, program akan membaca gambar sinyal yang telah dihasilkan dari proses generate data. Gambar sinyal tersebut kemudian akan diubah formatnya agar dapat diproses. Gambar sinyal yang telah diubah formatnya akan disimpan pada folder yang telah disediakan.

Algorithm 6 Convert Directory Images to JPEG Format

```
1: procedure CONVERTDIRECTORYToJPEG(input_dir, output_base_dir)
2:   for each root, dirs, files in input_dir do
3:     for each file in files do
4:       if file ends with (.png, .jpg, .jpeg, .tiff, .bmp, .gif) then
5:         Calculate input path
6:         Calculate relative path from input_dir
7:         Define output directory based on relative path
8:         if output directory does not exist then
9:           Create output directory
10:          end if
11:          Define output file path with .jpg extension
12:          Open the image from input path
13:          if image has alpha channel then
14:            Create a white background image of same size
15:            Paste image onto the background using alpha channel as mask
16:            Set image to the new composite image
17:          end if
18:          Save the image in JPEG format to output file path
19:          Break loop           ▷ Remove this line to process all files in the directory
20:        end if
21:      end for
22:    end for
23: end procedure
```



Gambar 3.21: Proses Binarisasi

Algorithm 7 Process Images in a Directory

```

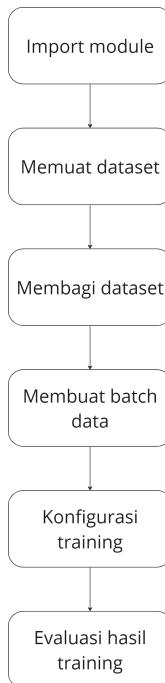
1: function PROCESSIMAGE(image_path, output_dir)
2:   Read the image from image_path
3:   Convert the image to grayscale
4:   Apply Otsu's thresholding to the grayscale image to obtain a binary image
5:   Construct the output path relative to output_dir
6:   Ensure the parent directory of output path exists
7:   Save the binary image to the output path
8: end function
9:
10: procedure PROCESSDIRECTORY(input_dir, output_dir)
11:   Convert input_dir and output_dir to Path objects
12:   Prepare a list of image files to process, filtering by extensions (.png, .jpg, .jpeg, .bmp)
13:   for each image_path in the list of image files do
14:     PROCESSIMAGE(image_path, output_dir)
15:   end for
16: end procedure
17:
18: Specify the input and output directories
19: Call PROCESSDIRECTORY(input_dir, output_dir)

```

Selanjutnya, flowchart diatas menjelaskan proses untuk mengubah gambar menjadi binarisasi. Pada program ini, pertama-tama dilakukan pembacaan lokasi file yang ingin diubah formatnya. Kemudian dilakukan import library yang dibutuhkan. Selanjutnya, program akan membaca gambar sinyal yang telah dihasilkan dari proses generate data. Gambar sinyal tersebut kemudian akan diubah formatnya menjadi grayscale. Setelah itu, diterapkan threshold agar gambar sinyal tersebut menjadi biner. Gambar sinyal yang telah diubah formatnya akan disimpan pada folder yang telah disediakan.

3.3.6 Training Data

Pada proses training data menggunakan CNN 2D, perlu dilakukan penyusunan program untuk training CNN 2D pada Google Colab. Berikut ini adalah flowchart dari training data CNN 2D yang dapat dilihat pada gambar 3.22.



Gambar 3.22: Training CNN 2D

Pada CNN 2D, digunakan Google Colab sebagai platform untuk training data. Pada CNN ini, digunakan tensorflow untuk mengimport semua kebutuhan CNN 2D. Kemudian mendefinisikan dua generator data gambar, train datagen dan val datagen, yang masing-masing digunakan untuk augmentasi data training secara otomatis dan menyiapkan data validasi. Kedua generator mengubah skala nilai pixel gambar dari [0, 255] menjadi [0, 1] dengan mengalikannya dengan 1,0/255 untuk menormalisasi data. train datagen juga menerapkan transformasi acak seperti geser dan zoom (masing-masing hingga 20%) dan mengatur fill mode ke "nearest" untuk menangani pixel di luar batas gambar selama transformasi. Generator data ini dikonfigurasi untuk membagi kumpulan data menjadi subset training dan validasi, dengan 20% data dicadangkan untuk validasi. Kemudian gambar diproses agar sesuai dengan ukuran input model 330x540 piksel, menanganinya dalam 10 batch, dan menghasilkan gambar grayscale. Mode kelas diatur ke "biner" yang menunjukkan bahwa klasifikasi tersebut adalah klasifikasi biner.

Kode CNN ini memiliki variasi dalam layerya. Hal serupa yang dimiliki setiap variasi adalah setiap variasi dari CNN 2-Dimensi memiliki setidaknya sebuah Convolutional layer dan Dense layer yang jumlah serta pengaturan lanjutan yang berbeda-beda pada setiap eksperimen. Kemudian ditambahkan dense layer terakhir dengan unit tunggal dan aktivasi sigmoid untuk klasifikasi biner. Selanjutnya, dilakukan konfigurasi model training dengan menggunakan model.compile() yang mengatur optimizer, loss function, dan metrik evaluasi. Pada model ini, digunakan optimizer Adam dengan loss function binary crossentropy, dan metrik evaluasi akurasi. Kemudian, model dilatih dengan model.fit() yang mengatur generator data training, jumlah epoch sebanyak, generator data validasi, dan langkah validasi. Pada model

Algorithm 8 Training and Evaluation of CNN for Image Classification

```
1: Import necessary modules and libraries
2: Mount Google Drive and set paths for datasets
3: Initialize random seeds for reproducibility
4: Define data generators with augmentation settings for training and validation
5: Prepare data generators for training and validation subsets
6: function CREATEMODEL
7:   Define a sequential model with multiple convolutional blocks
8:   Add pooling, flattening, dense, and dropout layers
9:   Configure compilation settings (optimizer, loss, metrics)
10:  return model
11: end function
12: Instantiate the model using CREATEMODEL
13: Print model summary
14: Define EarlyStopping callback for training
15: Train model using fit method with callbacks
16: Evaluate model on validation and training data
17: Plot accuracy and loss graphs for training and validation
18: Save the trained model to Google Drive
19: Predict on new data using test generators
20: Display and analyze confusion matrices for training and validation sets
21: Calculate and print F1 scores for training, validation, and test sets
22: Annotate test images with predictions and save outputs
```

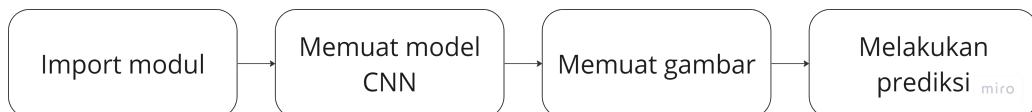
ini, dilakukan training sebanyak 100 epoch dengan early stopping dan langkah validasi yang disesuaikan dengan ukuran batch dan jumlah validasi. Setelah training selesai, model disimpan dalam format .h5 dan proses training menghasilkan data yang diperlukan terkait hasil training itu sendiri.

Pada percobaan pertama, kedua, dan ketiga, digunakan model CNN 2D yang sama untuk mengetahui pembagian data yang baik. Model pada ketiga percobaan ini dapat dilihat pada tabel 3.5 dimana model ini memiliki total 14 layer dengan rincian 4 Convolutional layer, 4 MaxPooling layer, 1 flatten layer, 2 dropout layer dan 3 dense layer. Untuk percobaan keempat, digunakan pembagian data yang terbaik dari tiga percobaan pertama. Terdapat perbedaan pada model CNN 2D pada percobaan keempat ini dimana model ini memiliki total 18 layer dengan rincian 4 Convolutional layer, 4 MaxPooling layer, 1 flatten layer, 2 dropout layer, 3 dense layer dan 4 BatchNormalization layer yang ditambahkan setelah setiap Convolutional layer. Model pada percobaan keempat ini dapat dilihat pada tabel 3.6. Sementara itu, pada percobaan kelima, digunakan model CNN 2D dengan layer yang lebih sedikit. Model pada percobaan kelima ini dapat dilihat pada tabel 3.7. Pada model tersebut, terdapat total 10 layer dengan rincian 2 Convolutional layer, 2 MaxPooling layer, 2 BatchNormalization layer, 1 flatten layer, 1 dropout layer, dan 2 dense layer. Pada percobaan keenam, digunakan model CNN 2D yang juga berbeda yang bisa dilihat pada tabel 3.8. Model pada percobaan keenam ini memiliki total 11 layer dengan rincian 3 Convolutional layer, 3 MaxPooling layer, 1 flatten layer, 2 dropout layer dan 2 dense layer. Pada percobaan ketujuh, digunakan model CNN 2D yang juga berbeda yang bisa dilihat pada tabel 3.9. Model pada percobaan keenam ini memiliki total 12 layer dengan rincian 6 Convolutional layer, 4 MaxPooling layer, 1 flatten layer dan 1 dense layer. Dari semua model

yang digunakan, digunakan input size dan target size yang sama yakni 550 x 330 piksel dengan mode klasifikasi biner dan warna grayscale.

3.3.7 Pengujian Klasifikasi Lanjut Revisi

Model CNN yang telah di training, akan dilakukan pengujian klasifikasi. Klasifikasi dilaksanakan dengan menggunakan data training, validasi, dan testing. Berikut ini adalah flowchart dari pengujian klasifikasi yang dapat dilihat pada gambar 3.23.



Gambar 3.23: Pengujian Klasifikasi

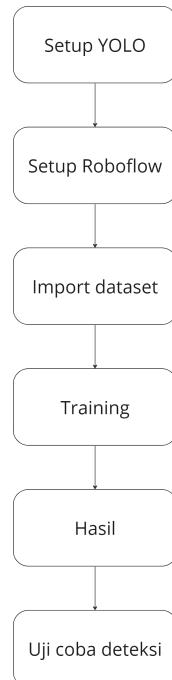
Algorithm 9 Image Classification with a Pre-trained Model

- 1: Mount Google Drive to access files
 - 2: Import necessary libraries and modules: TensorFlow, NumPy, Matplotlib
 - 3: Define class names for classification
 - 4: Load the pre-trained model from specified path
 - 5: Load the image from a path
 - 6: Preprocess the image: resize, convert color mode, normalize
 - 7: Expand dimensions of the image array for model input
 - 8: Perform image classification using the model
 - 9: Determine the predicted class based on the highest probability
 - 10: Display the image and the predicted class
-

Pengujian dapat dilakukan terhadap file tunggal maupun banyak file sekaligus. Pada program, dilakukan import module yang dibutuhkan dalam proses pengujian. Langkah selanjutnya, dimuat model CNN 2D yang telah di training. Kemudian, dimuat gambar yang ingin diuji klasifikasi. Setelah menjalankan proses klasifikasi, maka akan dihasilkan output berupa gambar yang telah diuji klasifikasinya dengan label kelas terklasifikasi. Untuk pengujian data secara keseluruhan, dilakukan pengujian sebagaimana yang tertera pada pseudocode subbab sebelumnya. Data yang dimuat tadi disesuaikan target ukurannya menjadi 550 x 330 piksel dengan mode klasifikasi biner dan warna grayscale. Pengaturan ini dilakukan untuk masing-masing data training, validasi, dan testing. Dilakukan prediksi pada masing-masing data tersebut yang kemudian hasilnya akan disajikan dalam bentuk confussion matrix. Confussion matrix ini akan menunjukkan hasil prediksi yang tepat dan tidak tepat dalam empat kategori yakni TP, FP, TN, dan FN. Selain itu, akan dihitung pula nilai akurasi dari model yang telah di training. Proses pengujian dilakukan lebih lanjut dengan setiap data testing yang diuji, disimpan hasil dari setiap gambar yang diuji tersebut ke dalam folder yang telah disediakan.

3.3.8 Deteksi YOLOv9

Gambar yang telah berhasil diklasifikasi akan dilakukan proses deteksi menggunakan YOLOv9. Pada proses deteksi YOLO, perlu dilakukan penyusunan program untuk deteksi YOLO pada Google Colab. Berikut ini adalah flowchart dari deteksi YOLO yang dapat dilihat pada gambar 3.24.



Gambar 3.24: Deteksi YOLOv9

Algorithm 10 Training and Inference with YOLOv9 on a Custom Dataset

- 1: Mount Google Drive for file access
 - 2: Check GPU availability using `nvidia-smi`
 - 3: Define the home directory and verify it with a print statement
 - 4: Clone YOLOv9 repository and navigate into it
 - 5: Install required libraries from the requirements file
 - 6: Install additional package, Roboflow
 - 7: Download model weights to the specified home directory weights folder
 - 8: Prepare the data directory within the home directory
 - 9: Download the dataset using Roboflow and configure it for YOLOv9
 - 10: Train the model using custom weights and hyperparameters
 - 11: Mount Google Drive if necessary
 - 12: Download the best model weights after training
 - 13: List training results including images and plots
 - 14: Display training result images using IPython display functionality
 - 15: Validate the custom model with specified parameters
 - 16: Perform detection using the trained model on validation images
 - 17: Display confusion matrix and a subset of detection result images
-

Pada program ini, pertama-tama dilakukan import library yang dibutuhkan. Selanjutnya, dilakukan setup YOLOv9 dengan melakukan clone YOLOv9 dari repository github yang kemudian dilakukan instalasi berdasarkan file requirement.txt agar YOLOv9 bisa bekerja. Selanjutnya, program melakukan instalasi Roboflow yang mana hal ini bertujuan agar program bisa melakukan import dari platform Roboflow karena dataset yang digunakan dianotasi pada platform tersebut. Didapatkan bobot YOLOv9 dari repository github yang telah diclone dan melakukan penyesuaian nilai jumlah kelas yang akan dideteksi sesuai dengan jumlah kelas yang ada.

Program kemudian akan melakukan import dataset dari Roboflow dimana dataset tersebut sudah dianotasi. Dilakukan proses training dengan mengatur beberapa parameter seperti ukuran batch, jumlah epoch, ukuran gambar. Ukuran batch diatur menjadi 8, jumlah epoch diatur menjadi 10, dan ukuran gambar diatur menjadi 640x640 piksel menyesuaikan dengan dataset yang telah dilakukan bounding box pada Roboflow. Setelah proses training, didapatkan nilai dari akurasi model tersebut. Kemudian dilakukan prediksi pada yang telah tersedia.

BAB IV

PENGUJIAN DAN ANALISIS

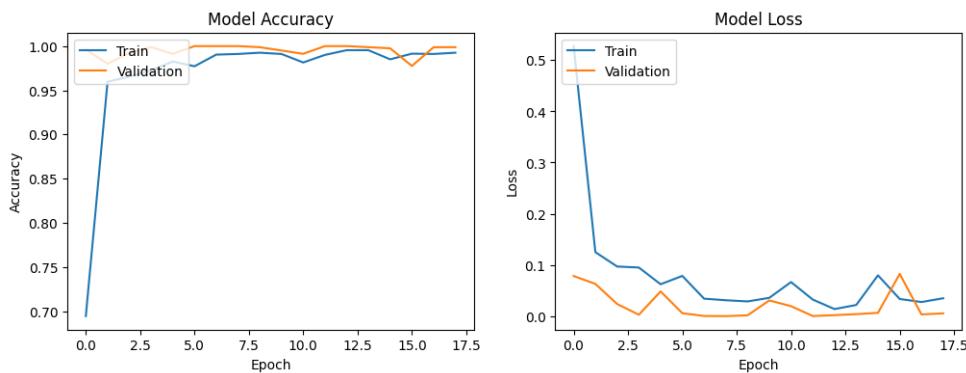
Pada bab ini akan dipaparkan mengenai beberapa hasil pengujian sesuai dengan telah dijelaskan pada metodologi. Hasil pengujian ini dilakukan guna untuk mendapatkan hasil klasifikasi dan deteksi yang optimal. Pelaksanaan metodologi serta hasil pengujian yang akan dipaparkan dalam bab ini diharapkan dapat memberikan pemahaman mengenai hasil dan pembahasan sehingga dapat ditarik kesimpulan dari Tugas Akhir yang telah dilaksanakan.

4.1 Pengujian Klasifikasi Menggunakan CNN 2-Dimensi

Proses klasifikasi menggunakan CNN 2-Dimensi dilakukan sebanyak tiga kali percobaan. Pada setiap percobaan, dilakukan training dengan pengaturan CNN 2-Dimensi yang sama kecuali dalam satu hal. Setiap percobaan dilakukan dengan pembagian jumlah data berbeda-beda untuk training, validation, dan testing. Dari hasil tersebut akan dilakukan variasi struktur CNN 2-Dimensi dengan pembagian dataset yang memiliki hasil yang baik. Hasil klasifikasi menggunakan CNN 2-Dimensi akan dibandingkan dengan hasil klasifikasi menggunakan Roboflow dan CNN 2-Dimensi pada percobaan yang lain. Pada pengujian CNN 2-Dimensi ini nantinya akan digunakan gambar yang sama dalam masing-masing eksperimen.

4.1.1 Percobaan Pertama

Pada percobaan pertama, data yang digunakan untuk training, validation, dan testing masing-masing sebanyak 70%, 20%, dan 10%. Hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan pertama mendapatkan tingkat akurasi training sebesar 0.9918 dengan training loss sebesar 0.0235. Sementara itu, didapatkan tingkat akurasi validasi sebesar 1.0 dengan validation loss sebesar 4.6915e-05. Berikut ini adalah grafik model accuracy dan model loss pada percobaan pertama yang dapat dilihat pada Gambar 4.1.



Gambar 4.1: Grafik Akurasi dan Loss Model Percobaan Pertama

Selain itu, didapatkan pula hasil confusion matrix dari data training, validation, dan testing pada percobaan pertama yang masing-masing dapat dilihat pada Gambar 4.2 dan 4.3. Dari hasil tersebut, dapat dilihat bahwa hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan

pertama memiliki tingkat akurasi yang cukup tinggi. Hal ini didukung dengan nilai f1 score pada data training sebesar 0.98926264, validation sebesar 1.0, dan testing sebesar 0.99750614. Inference time yang didapatkan pada percobaan pertama untuk data training sebanyak 2804 data adalah 79 detik (283ms/step), untuk data validation sebanyak 800 data adalah 9 detik (117ms/step), dan untuk data testing sebanyak 400 data adalah 164 detik (4s/step).

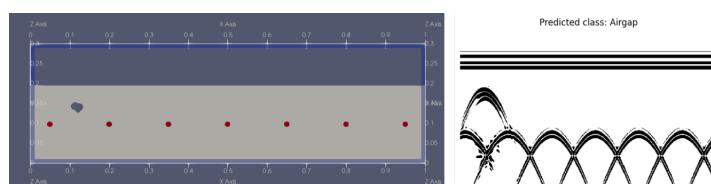
		Training Confusion Matrix		Validation Confusion Matrix	
		Negative	Positive	Negative	Positive
True Labels	Negative	1392	8	400	0
	Positive	22	1382	0	400
		Negative	Positive	Negative	Positive
		Predicted Labels	Predicted Labels	Predicted Labels	Predicted Labels

Gambar 4.2: Confussion Matrix Training dan Validasi Percobaan Pertama

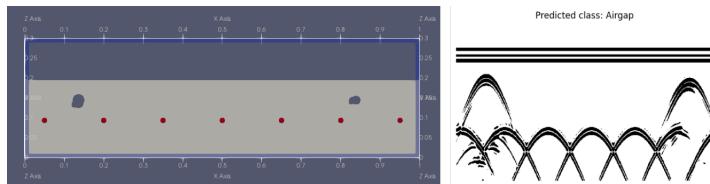
		Test Confusion Matrix	
		Negative	Positive
True Labels	Negative	199	1
	Positive	0	200
		Negative	Positive
		Predicted Labels	Predicted Labels

Gambar 4.3: Confussion Matrix Testing Percobaan Pertama

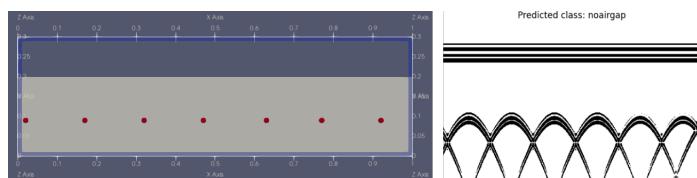
Hasil training yang cukup baik dari percobaan pertama dapat dilihat pula pada keberhasilannya dalam mengklasifikasikan gambar dengan tepat seperti yang dapat dilihat pada gambar berikut:



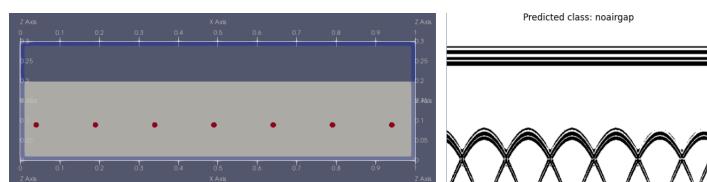
Gambar 4.4: Hasil Klasifikasi Percobaan Pertama 1



Gambar 4.5: Hasil Klasifikasi Percobaan Pertama 2



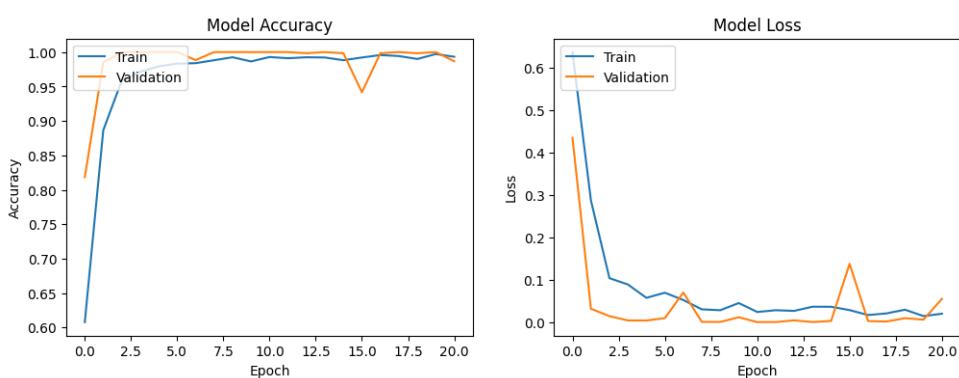
Gambar 4.6: Hasil Klasifikasi Percobaan Pertama 3



Gambar 4.7: Hasil Klasifikasi Percobaan Pertama 4

4.1.2 Percobaan Kedua

Pada percobaan kedua, data yang digunakan untuk training, validation, dan testing masing-masing sebanyak 70%, 15%, dan 15%. Hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan kedua mendapatkan tingkat akurasi training sebesar 0.9964 dengan training loss sebesar 0.0103. Sementara itu, didapatkan tingkat akurasi validasi sebesar 1.0 dengan validation loss sebesar 2.0294e-04. Berikut ini adalah grafik model accuracy dan model loss pada percobaan kedua yang dapat dilihat pada Gambar 4.8.



Gambar 4.8: Grafik Akurasi dan Loss Model Percobaan Kedua

Selain itu, didapatkan pula hasil confusion matrix dari data training, validation, dan testing pada percobaan kedua yang masing-masing dapat dilihat pada Gambar 4.9 dan 4.10. Dari hasil

tersebut, dapat dilihat bahwa hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan kedua memiliki tingkat akurasi yang cukup tinggi. Hal ini didukung dengan nilai f1 score pada data training sebesar 0.99714893, validation sebesar 1.0, dan testing sebesar 1.0. Inference time yang didapatkan pada percobaan kedua untuk data training sebanyak 2800 data adalah 80 detik (285ms/step), untuk data validation sebanyak 600 data adalah 7 detik (115ms/step), dan untuk data testing sebanyak 600 data adalah 8 detik (128ms/step).

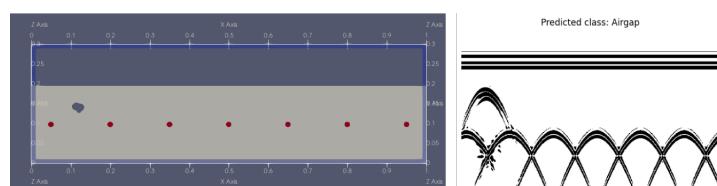
		Training Confusion Matrix		Validation Confusion Matrix	
		Negative	Positive	Negative	Positive
True Labels	Negative	1393	7	300	0
	Positive	1	1399	0	300
Predicted Labels		Negative	Positive	Negative	Positive
		Predicted Labels		Predicted Labels	

Gambar 4.9: Confussion Matrix Training dan Validasi Percobaan Kedua

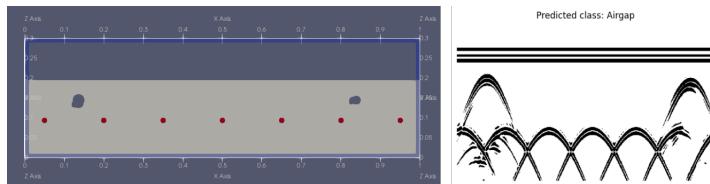
		Test Confusion Matrix	
		Negative	Positive
True Labels	Negative	300	0
	Positive	0	300
Predicted Labels		Negative	Positive
		Predicted Labels	

Gambar 4.10: Confussion Matrix Testing Percobaan Kedua

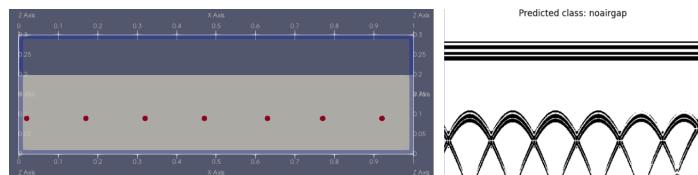
Hasil training yang cukup baik dari percobaan kedua dapat dilihat pula pada keberhasilannya dalam mengklasifikasikan gambar dengan tepat seperti yang dapat dilihat pada gambar berikut:



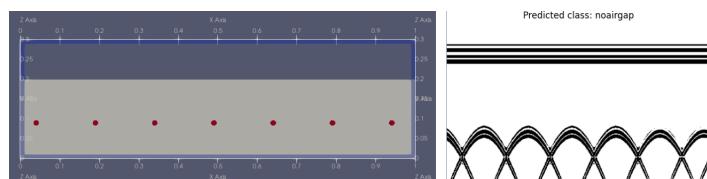
Gambar 4.11: Hasil Klasifikasi Percobaan Kedua 1



Gambar 4.12: Hasil Klasifikasi Percobaan Kedua 2



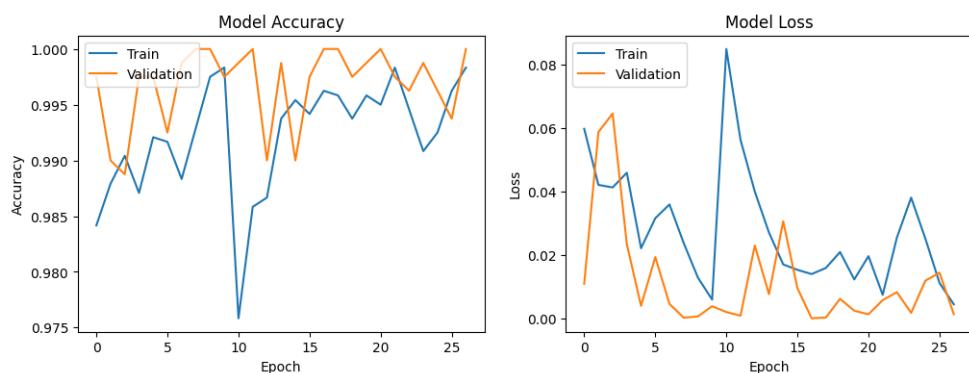
Gambar 4.13: Hasil Klasifikasi Percobaan Kedua 3



Gambar 4.14: Hasil Klasifikasi Percobaan Kedua 4

4.1.3 Percobaan Ketiga

Pada percobaan ketiga, data yang digunakan untuk training, validation, dan testing masing-masing sebanyak 60%, 20%, dan 20%. Hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan ketiga mendapatkan tingkat akurasi training sebesar 0.9946 dengan training loss sebesar 0.0147. Sementara itu, didapatkan tingkat akurasi validasi sebesar 1.0 dengan validation loss sebesar 1.1496e-04. Berikut ini adalah grafik model accuracy dan model loss pada percobaan ketiga yang dapat dilihat pada Gambar 4.15.



Gambar 4.15: Grafik Akurasi dan Loss Model Percobaan Ketiga

Meskipun pada hasil training, model memiliki hasil yang baik, grafik hasil training menunjukkan training yang tidak begitu baik dan mengindikasikan overfitting dengan adanya spiking.

Tidak dipungkiri bahwa hal ini dapat terjadi karena pada salah satu kelas dataset yakni noair-gap, kelas tersebut memiliki data yang hampir serupa satu sama lain. Selain itu, didapatkan pula hasil confusion matrix dari data training, validation, dan testing pada percobaan ketiga yang masing-masing dapat dilihat pada Gambar 4.16 dan 4.17. Dari hasil tersebut, dapat dilihat bahwa hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan ketiga memiliki tingkat akurasi yang baik. Meskipun pada confusion matrix data training dan valiasi masih menunjukkan kesalahan, namun pada data testing menunjukkan keakuratan yang baik. Hal ini didukung dengan nilai f1 score pada data training sebesar 0.99502486, validation sebesar 1.0, dan testing sebesar 1.0. Inference time yang didapatkan pada percobaan ketiga untuk data training sebanyak 2400 data adalah 72 detik (301ms/step), untuk data validation sebanyak 800 data adalah 9 detik (114ms/step), dan untuk data testing sebanyak 800 data adalah 115 detik (1s/step).

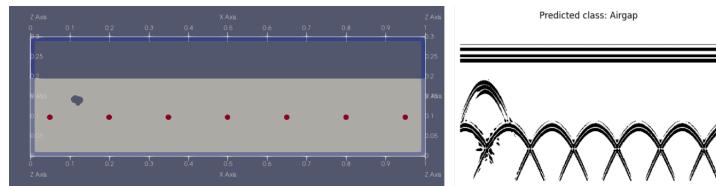
		Training Confusion Matrix		Validation Confusion Matrix	
		True Labels		True Labels	
		Negative	Positive	Negative	Positive
True Labels	Negative	1188	12	400	0
	Positive	0	1200	0	400
		Negative	Positive	Negative	Positive
Predicted Labels		Predicted Labels		Predicted Labels	

Gambar 4.16: Confusion Matrix Training dan Validasi Percobaan Ketiga

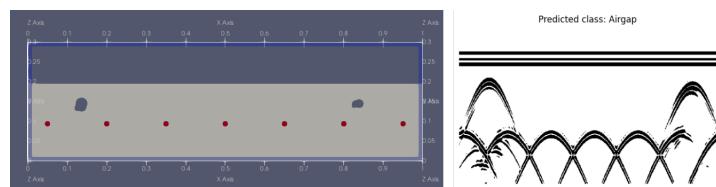
		Test Confusion Matrix	
		True Labels	
		Negative	Positive
True Labels	Negative	400	0
	Positive	0	400
		Negative	Positive
Predicted Labels		Predicted Labels	

Gambar 4.17: Confusion Matrix Testing Percobaan Ketiga

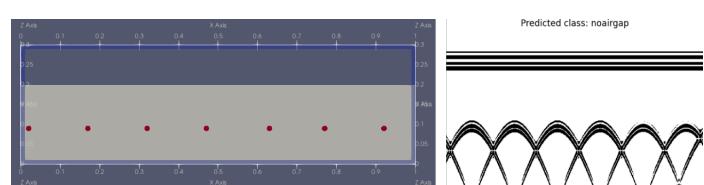
Hasil training yang cukup baik dari percobaan ketiga dapat dilihat pula pada keberhasilannya dalam mengklasifikasikan gambar dengan tepat seperti yang dapat dilihat pada gambar berikut:



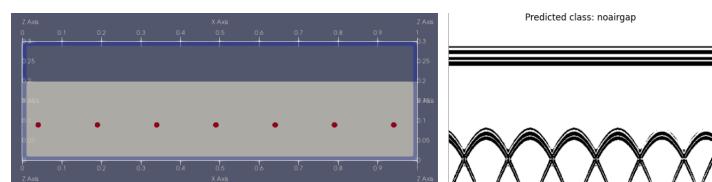
Gambar 4.18: Hasil Klasifikasi Percobaan Ketiga 1



Gambar 4.19: Hasil Klasifikasi Percobaan Ketiga 2



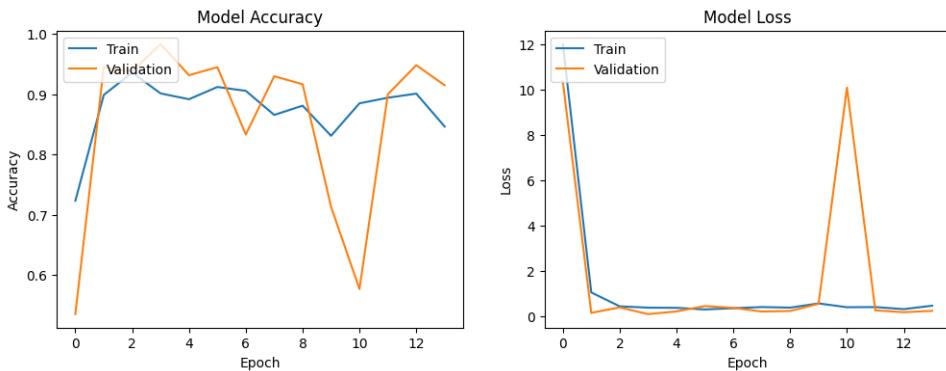
Gambar 4.20: Hasil Klasifikasi Percobaan Ketiga 3



Gambar 4.21: Hasil Klasifikasi Percobaan Ketiga 4

4.1.4 Percobaan Keempat

Pada percobaan keempat, data yang digunakan untuk training, validation, dan testing masing-masing sebanyak 70%, 15%, dan 15%. Selain itu, ditambahkan variasi untuk kode CNN 2-Dimensi dengan menambahkan BatchNormalization setelah setiap layer Conv2D. Hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan keempat mendapatkan tingkat akurasi training sebesar 0.9621 dengan training loss sebesar 0.1670. Sementara itu, didapatkan tingkat akurasi validasi sebesar 0.9833 dengan validation loss sebesar 0.1061. Berikut ini adalah grafik model accuracy dan model loss pada percobaan keempat yang dapat dilihat pada Gambar 4.22.



Gambar 4.22: Grafik Akurasi dan Loss Model Percobaan Keempat

Berdasarkan grafik hasil training, meskipun hasil memiliki tingkat keakurasaian yang baik, grafik menunjukkan perilaku yang tidak wajar dimana terjadi spiking yang tinggi. Hal ini dapat terjadi karena pada salah satu kelas dataset yakni noairgap, kelas tersebut memiliki data yang hampir serupa satu sama lain. Penyebab lain dapat bersumber dari perubahan struktur layer CNN 2-Dimensis yang digunakan. Selain itu, didapatkan pula hasil confusion matrix dari data training, validation, dan testing pada percobaan keempat yang masing-masing dapat dilihat pada Gambar 4.23 dan 4.24. Dari hasil tersebut, dapat dilihat bahwa hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan keempat memiliki tingkat akurasi sedang. Hal ini didukung dengan hasil confusion matrix pada data training, validasi, dan testing yang memiliki error yang lebih banyak dibandingkan dengan percobaan yang lain. Nilai f1 score yang didapatkan pada data training sebesar 0.961155, validation sebesar 0.98360646, dan testing sebesar 0.9803921. Inference time yang didapatkan pada percobaan keempat untuk data training sebanyak 2800 data adalah 79 detik (281ms/step), untuk data validation sebanyak 600 data adalah 10 detik (162ms/step), dan untuk data testing sebanyak 600 data adalah 130 detik (2s/step).

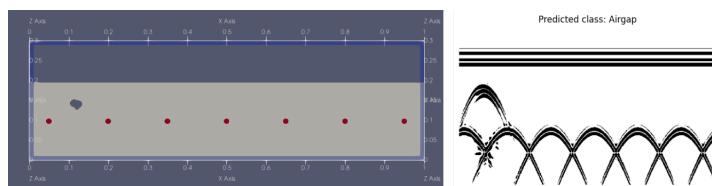
		Training Confusion Matrix		Validation Confusion Matrix	
		Negative	Positive	Negative	Positive
True Labels	Negative	1289	111	290	10
	Positive	2	1388	0	300
		Negative	Positive	Negative	Positive
		Predicted Labels	Predicted Labels	Predicted Labels	Predicted Labels

Gambar 4.23: Confussion Matrix Training dan Validasi Percobaan Keempat

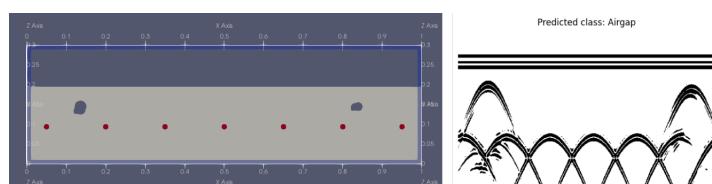
		Test Confusion Matrix	
		Negative	Positive
True Labels	Negative	288	12
	Positive	0	300
		Negative	Positive
		Predicted Labels	

Gambar 4.24: Confussion Matrix Testing Percobaan Keempat

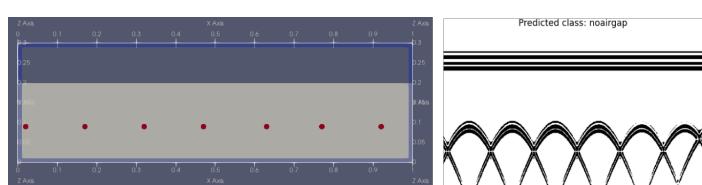
Hasil training yang cukup baik dari percobaan keempat dapat dilihat pula pada keberhasilannya dalam mengklasifikasikan gambar dengan tepat seperti yang dapat dilihat pada gambar berikut:



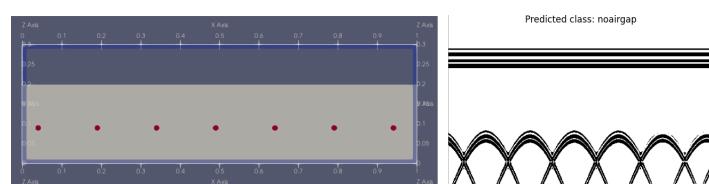
Gambar 4.25: Hasil Klasifikasi Percobaan Keempat 1



Gambar 4.26: Hasil Klasifikasi Percobaan Keempat 2



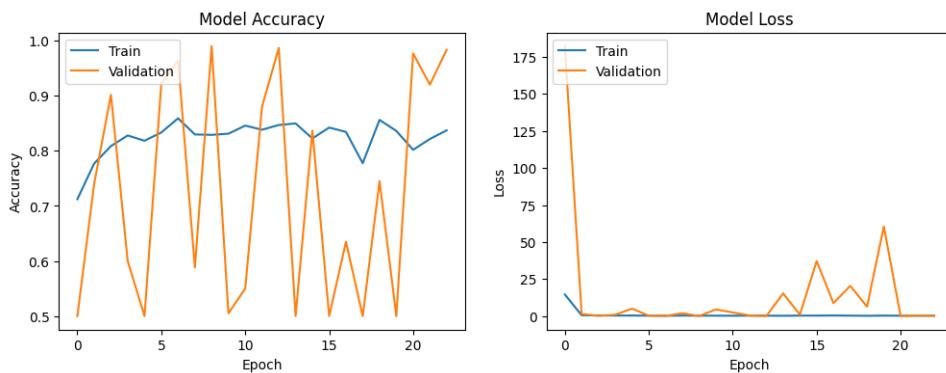
Gambar 4.27: Hasil Klasifikasi Percobaan Keempat 3



Gambar 4.28: Hasil Klasifikasi Percobaan Keempat 4

4.1.5 Percobaan Kelima

Pada percobaan kelima, data yang digunakan untuk training, validation, dan testing masing-masing sebanyak 70%, 15%, dan 15%. Pada percobaan ini, ditambahkan variasi untuk kode CNN 2-Dimensi dengan mengurangi jumlah layer Conv2D menjadi dua dan layer Dense menjadi 1. Layer Conv2D berawal dari 32 dan berakhir di 64 dengan MaxPooling dan Batch-Normalization yang sama dengan percobaan keempat. Hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan kelima mendapatkan tingkat akurasi training sebesar 0.9154 dengan training loss sebesar 0.2236. Sementara itu, didapatkan tingkat akurasi validasi sebesar 0.9867 dengan validation loss sebesar 0.0336. Berikut ini adalah grafik model accuracy dan model loss pada percobaan kelima yang dapat dilihat pada Gambar 4.29.



Gambar 4.29: Grafik Akurasi dan Loss Model Percobaan Kelima

Berdasarkan grafik hasil training, meskipun hasil memiliki tingkat keakurasi yang lebih rendah dibandingkan percobaan sebelumnya, grafik menunjukkan perilaku yang tidak wajar dimana grafik menunjukkan nilai yang tinggi kemudian rendah berulang kali. Hal ini dapat terjadi karena pada salah satu kelas dataset yakni noairgap, kelas tersebut memiliki data yang hampir serupa satu sama lain. Penyebab lain dapat bersumber dari perubahan struktur layer CNN 2-Dimensi yang digunakan. Selain itu, didapatkan pula hasil confusion matrix dari data training, validation, dan testing pada percobaan kelima yang masing-masing dapat dilihat pada Gambar 4.30 dan 4.31. Dari hasil tersebut, dapat dilihat bahwa hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan kelima terjadi penurunan dalam hal tingkat akurasi. Didapatkan nilai f1 score pada data training sebesar 0.92211217, validation sebesar 0.98684204, dan testing sebesar 0.98360646. Inference time yang didapatkan pada percobaan kelima untuk data training sebanyak 2800 data adalah 85 detik (304ms/step), untuk data validation sebanyak 600 data adalah 8 detik (125ms/step), dan untuk data testing sebanyak 600 data adalah 83 detik (1s/step).

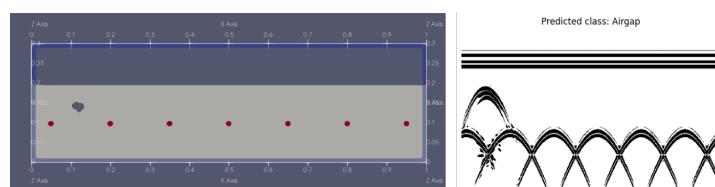
		Training Confusion Matrix		Validation Confusion Matrix	
		Negative	Positive	Negative	Positive
True Labels	Negative	1167	233	292	8
	Positive	3	1397	0	300
		Negative	Positive	Negative	Positive
		Predicted Labels	Predicted Labels	Predicted Labels	Predicted Labels

Gambar 4.30: Confussion Matrix Training dan Validasi Percobaan Kelima

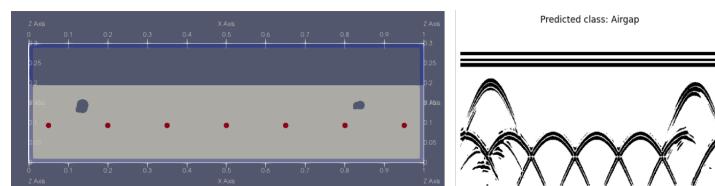
		Test Confusion Matrix	
		Negative	Positive
True Labels	Negative	290	10
	Positive	0	300
		Negative	Positive
		Predicted Labels	Predicted Labels

Gambar 4.31: Confussion Matrix Testing Percobaan Kelima

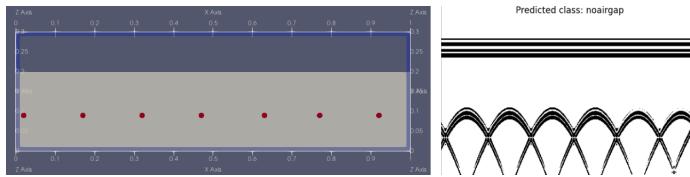
Hasil training yang cukup baik dari percobaan kelima dapat dilihat pula pada keberhasilannya dalam mengklasifikasikan gambar dengan tepat seperti yang dapat dilihat pada gambar berikut:



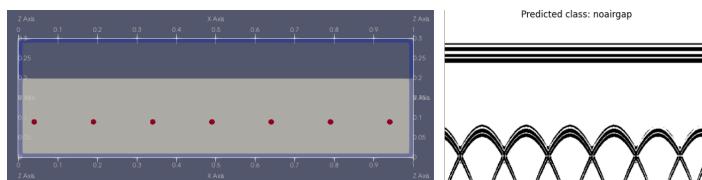
Gambar 4.32: Hasil Klasifikasi Percobaan Kelima 1



Gambar 4.33: Hasil Klasifikasi Percobaan Kelima 2



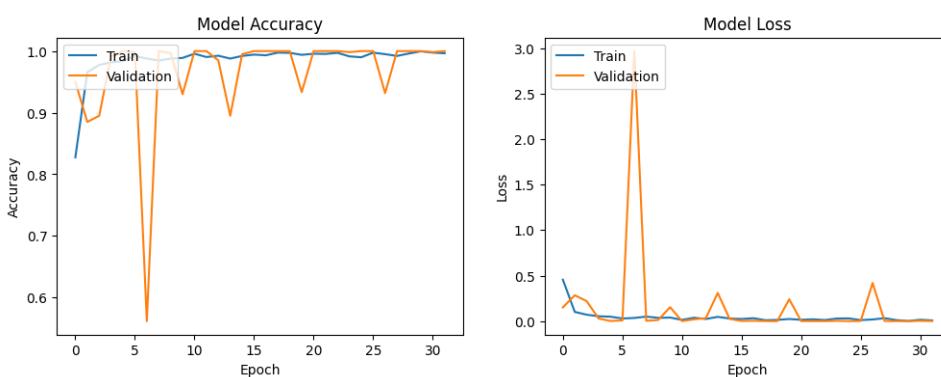
Gambar 4.34: Hasil Klasifikasi Percobaan Kelima 3



Gambar 4.35: Hasil Klasifikasi Percobaan Kelima 4

4.1.6 Percobaan Keenam

Pada percobaan keenam, data yang digunakan untuk training, validation, dan testing masing-masing sebanyak 70%, 15%, dan 15%. Pada percobaan ini, ditambahkan variasi untuk kode CNN 2-Dimensi dengan menjadikan struktur CNN 2-Dimensi hanya memiliki 3 layer Conv2D tanpa BatchNormalization, masih memiliki MaxPooling yang sama namun layer Conv2D memiliki parameter tambahan yakni padding yang nilainya diatur menjadi 'same' dengan layer Conv2D bermula dari 32 dan berakhir pada 128. Selain itu, ditambahkan dropout layer senilai 0.25 sebelum memasuki flatten layer dan ditambahkan 1 Dense layer setelahnya. Hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan keenam mendapatkan tingkat akurasi training sebesar 0.9993 dengan training loss sebesar 0.0028. Sementara itu, didapatkan tingkat akurasi validasi sebesar 1.0 dengan valiation loss sebesar 4.2127e-06. Berikut ini adalah grafik model accuracy dan model loss pada percobaan keenam yang dapat dilihat pada Gambar 4.36.



Gambar 4.36: Grafik Akurasi dan Loss Model Percobaan Keenam

Berdasarkan grafik hasil training, meskipun hasil memiliki tingkat keakurasi yang perlahan-lahan membaik dibandingkan percobaan sebelumnya, grafik menunjukkan perilaku yang tidak wajar dimana grafik menunjukan spiking yang tinggi. Hal ini dapat terjadi karena pada salah satu kelas dataset yakni noairgap, kelas tersebut memiliki data yang hampir serupa satu sama lain. Selain itu, didapatkan pula hasil confusion matrix dari data training, validation, dan testing pada percobaan keenam yang masing-masing dapat dilihat pada Gambar 4.37 dan 4.38.

Dari hasil tersebut, dapat dilihat bahwa hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan keenam memiliki tingkat akurasi yang cukup tinggi. Hal ini didukung dengan nilai f1 score pada data training sebesar 0.9985734, validation sebesar 1.0, dan testing sebesar 1.0. Inference time yang didapatkan pada percobaan keenam untuk data training sebanyak 2800 data adalah 76 detik (269ms/step), untuk data validation sebanyak 600 data adalah 7 detik (119ms/step), dan untuk data testing sebanyak 600 data adalah 97 detik (2s/step).

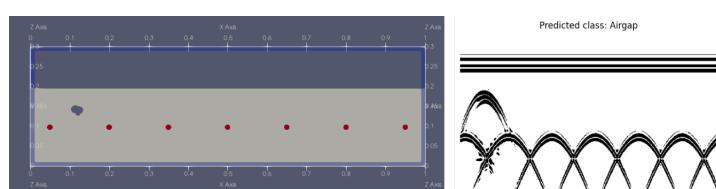
		Training Confusion Matrix		Validation Confusion Matrix	
		Negative	Positive	Negative	Positive
True Labels	Negative	1396	4	300	0
	Positive	0	1400	0	300
		Negative	Positive	Negative	Positive
		Predicted Labels	Predicted Labels	Predicted Labels	Predicted Labels

Gambar 4.37: Confussion Matrix Training dan Validasi Percobaan Keenam

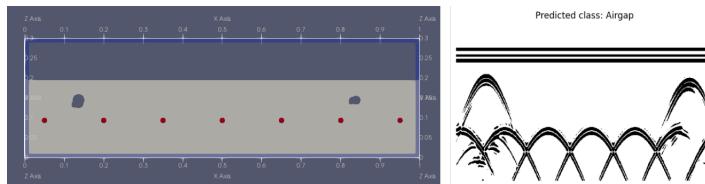
		Test Confusion Matrix	
		Negative	Positive
True Labels	Negative	300	0
	Positive	0	300
		Negative	Positive
		Predicted Labels	Predicted Labels

Gambar 4.38: Confussion Matrix Testing Percobaan Keenam

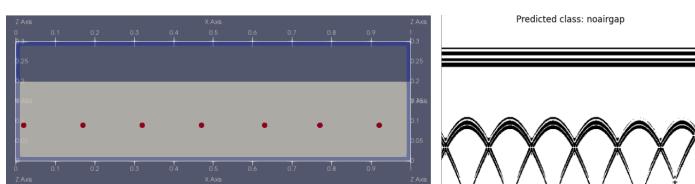
Hasil training yang cukup baik dari percobaan keenam dapat dilihat pula pada keberhasilannya dalam mengklasifikasikan gambar dengan tepat seperti yang dapat dilihat pada gambar berikut:



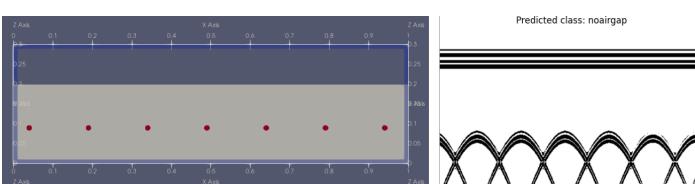
Gambar 4.39: Hasil Klasifikasi Percobaan Keenam 1



Gambar 4.40: Hasil Klasifikasi Percobaan Keenam 2



Gambar 4.41: Hasil Klasifikasi Percobaan Keenam 3

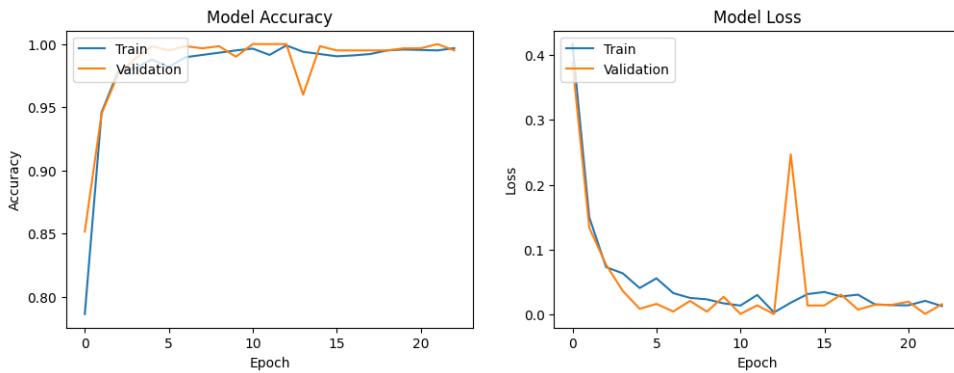


Gambar 4.42: Hasil Klasifikasi Percobaan Keenam 4

Dari semua percobaan diatas, dapat diketahui bahwa hasil terbaik didapatkan pada percobaan kedua. Meskipun semua percobaan memiliki hasil klasifikasi yang baik, percobaan kedua memiliki hasil metrics yang lebih baik dibandingkan dengan percobaan lainnya.

4.1.7 Percobaan Ketujuh

Pada percobaan Ketujuh, data yang digunakan untuk training, validation, dan testing masing-masing sebanyak 70%, 15%, dan 15%. Pada percobaan ini, ditambahkan variasi untuk kode CNN 2-Dimensi dengan menjadikan struktur CNN 2-Dimensi memiliki 6 layer Conv2D dengan masih memiliki MaxPooling yang sama namun hanya pada empat layer pertama. Selain itu, model ini tidak menggunakan dropout layer serta sebelum memasuki flatten layer dan ditambahkan 1 Dense layer setelahnya. Hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan ketujuh mendapatkan tingkat akurasi training sebesar 0.9979 dengan training loss sebesar 0.0060. Sementara itu, didapatkan tingkat akurasi validasi sebesar 1.0 dengan validation loss sebesar 1.7544e-04. Berikut ini adalah grafik model accuracy dan model loss pada percobaan ketujuh yang dapat dilihat pada Gambar 4.43.



Gambar 4.43: Grafik Akurasi dan Loss Model Percobaan Ketujuh

Berdasarkan grafik hasil training, grafik menunjukkan perilaku yang tidak wajar dimana grafik menunjukkan spiking yang tinggi. Hal ini dapat terjadi karena pada salah satu kelas dataset yakni noairgap, kelas tersebut memiliki data yang hampir serupa satu sama lain. Selain itu, didapatkan pula hasil confusion matrix dari data training, validation, dan testing pada percobaan ketujuh yang masing-masing dapat dilihat pada Gambar 4.44 dan 4.45. Dari hasil tersebut, dapat dilihat bahwa hasil klasifikasi menggunakan CNN 2-Dimensi pada percobaan ketujuh memiliki tingkat akurasi yang tinggi. Hal ini didukung dengan nilai f1 score pada data training sebesar 0.9985734, validation sebesar 1.0, dan testing sebesar 1.0. Inference time yang didapatkan pada percobaan ketujuh untuk data training sebanyak 2800 data adalah 76 detik (269ms/step), untuk data validation sebanyak 600 data adalah 6 detik (107ms/step), dan untuk data testing sebanyak 600 data adalah 81 detik (1s/step).

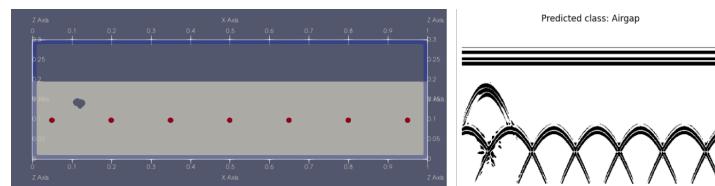
		Training Confusion Matrix		Validation Confusion Matrix	
		Negative	Positive	Negative	Positive
True Labels	Negative	1396	4	300	0
	Positive	0	1400	0	300
		Negative	Positive	Negative	Positive
		Predicted Labels	Predicted Labels	Predicted Labels	Predicted Labels

Gambar 4.44: Confussion Matrix Training dan Validasi Percobaan Ketujuh

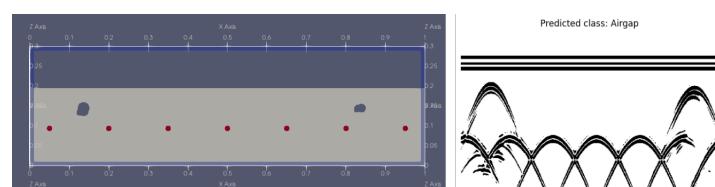
		Test Confusion Matrix	
		Negative	Positive
True Labels	Negative	300	0
	Positive	0	300
		Negative	Positive
		Predicted Labels	

Gambar 4.45: Confussion Matrix Testing Percobaan Ketujuh

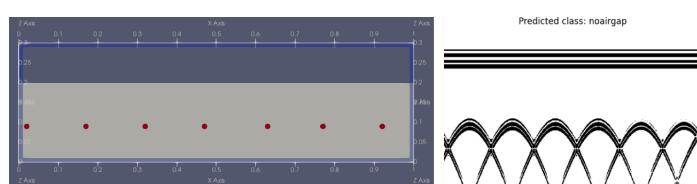
Hasil training yang cukup baik dari percobaan ketujuh dapat dilihat pula pada keberhasilannya dalam mengklasifikasikan gambar dengan tepat seperti yang dapat dilihat pada gambar berikut:



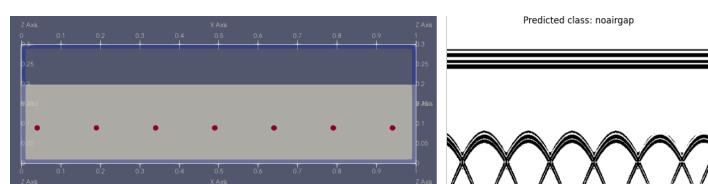
Gambar 4.46: Hasil Klasifikasi Percobaan Ketujuh 1



Gambar 4.47: Hasil Klasifikasi Percobaan Ketujuh 2



Gambar 4.48: Hasil Klasifikasi Percobaan Ketujuh 3



Gambar 4.49: Hasil Klasifikasi Percobaan Ketujuh 4

Dari semua percobaan diatas, dapat diketahui bahwa hasil terbaik didapatkan pada percobaan kedua. Meskipun semua percobaan memiliki hasil klasifikasi yang baik, percobaan kedua memiliki hasil metrics yang lebih baik dibandingkan dengan percobaan lainnya. Hal tersebut dapat terlihat dari tabel 4.1 berikut.

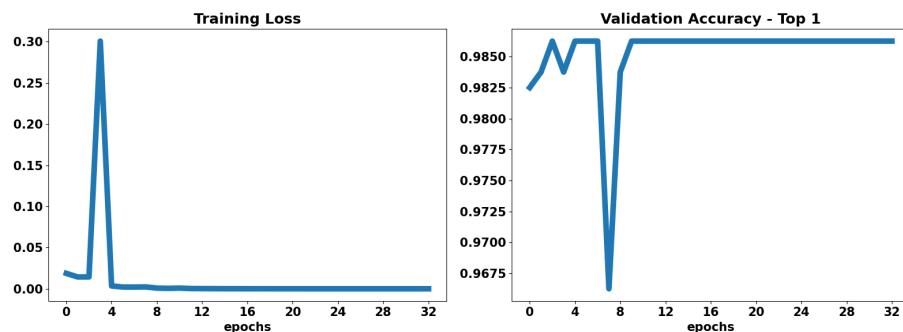
Percobaan ke	F1 Score			Inference Time / step		
	Training	Validation	Testing	Training	Validation	Testing
1	0.98926264	1.0	0.99750614	283ms	117ms	4s
2	0.99714893	1.0	1.0	285ms	115ms	125ms
3	0.99502486	1.0	1.0	301ms	114ms	1s
4	0.961155	0.98360646	0.9803921	281ms	162ms	2s
5	0.92211217	0.98684204	0.98360646	304ms	125ms	1s
6	0.9985734	1.0	1.0	269ms	119ms	2s
7	0.9985734	1.0	1.0	269ms	107ms	1s

Tabel 4.1: Tabel Perbandingan Hasil F1 Score dan Inference Time

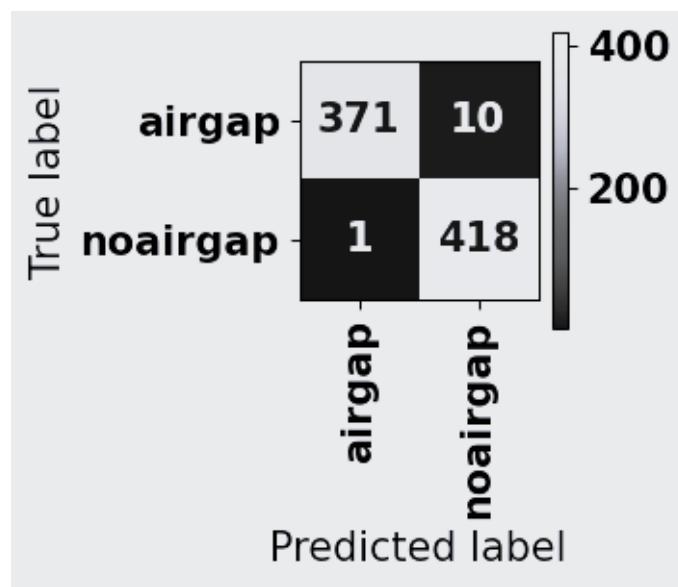
Berdasarkan tabel 4.1 diatas, dapat dilihat bahwa semua percobaan memiliki hasil F1 score dan inference time yang relatif sama. Akan tetapi, pada percobaan kedua, dihasilkan inference time yang paling cepat pada data testing dibandingkan dengan percobaan lainnya dengan selisih yang cukup besar. Oleh karena itu, percobaan kedua dipilih sebagai model terbaik untuk klasifikasi menggunakan CNN 2D.

4.2 Pengujian Klasifikasi Roboflow

Proses klasifikasi menggunakan Roboflow hanya dilakukan satu kali dan hasil yang didapatkan akan dibandingkan dengan hasil klasifikasi menggunakan CNN 2-Dimensi. Hasil klasifikasi menggunakan Roboflow mendapatkan tingkat akurasi validasi sebesar 98.6%. Selain itu, klasifikasi menggunakan juga menghasilkan grafik training loss dan validation accuracy serta confusion matrix yang masing-masing dapat dilihat pada Gambar 4.50 dan Gambar 4.51.

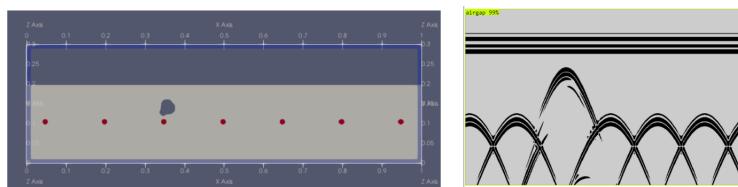


Gambar 4.50: Hasil Training Loss dan Validation Accuracy Roboflow

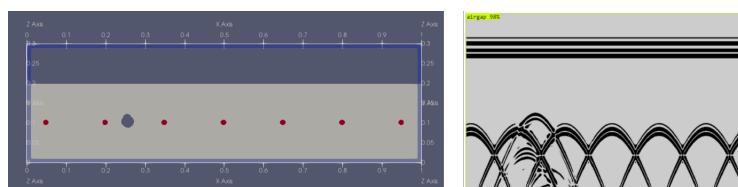


Gambar 4.51: Confussion Matrix Roboflow

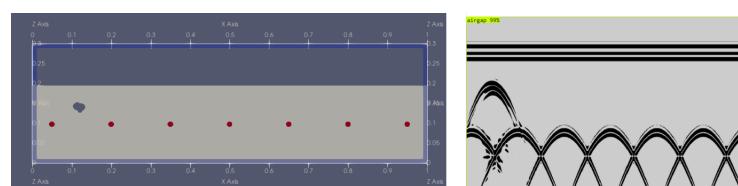
Dari hasil diatas, dapat didapatkan bahwa hasil klasifikasi menggunakan Roboflow memiliki tingkat akurasi yang cukup baik. Namun, hasil klasifikasi menggunakan Roboflow tidak dapat dijadikan sebagai acuan karena hanya dilakukan satu kali. Oleh karena itu, hasil klasifikasi menggunakan Roboflow akan dibandingkan dengan hasil klasifikasi menggunakan CNN 2-Dimensi serta akan diuji lagi melalui deteksi YOLOv9. Berikut ini merupakan hasil klasifikasi menggunakan Roboflow yang dapat dilihat pada gambar berikut:



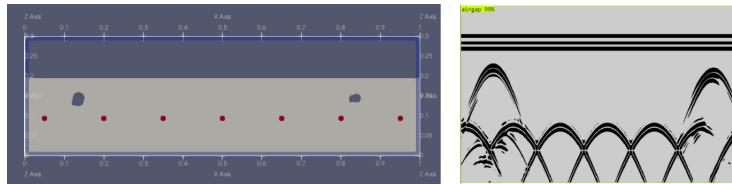
Gambar 4.52: Hasil Klasifikasi Roboflow 1



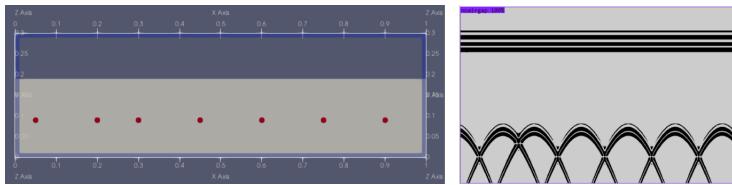
Gambar 4.53: Hasil Klasifikasi Roboflow 2



Gambar 4.54: Hasil Klasifikasi Roboflow 3



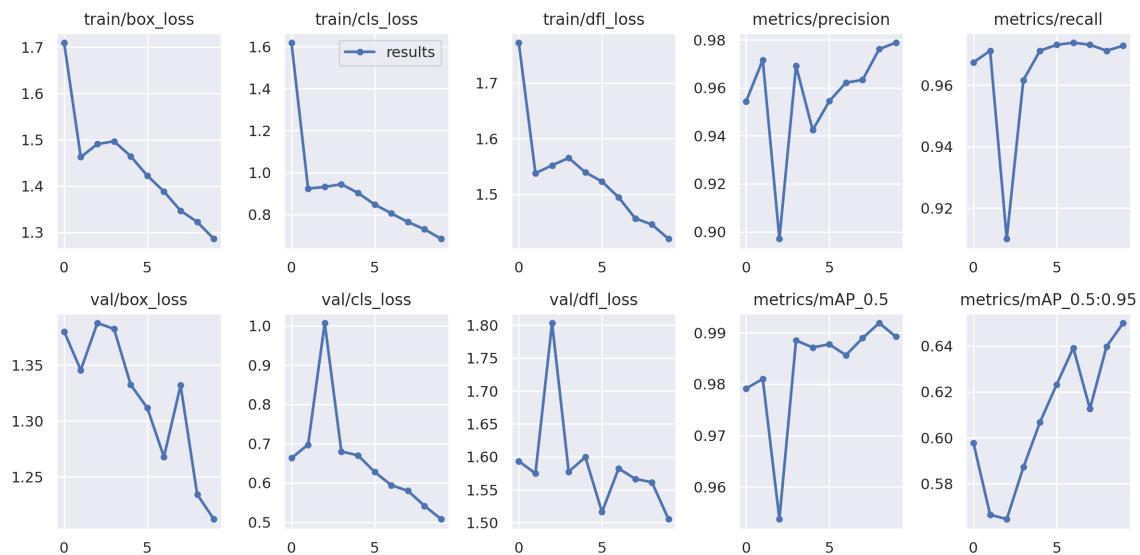
Gambar 4.55: Hasil Klasifikasi Roboflow 4



Gambar 4.56: Hasil Klasifikasi Roboflow 5

4.3 Pengujian Deteksi YOLOv9

Proses deteksi menggunakan YOLOv9 dilakukan sebanyak satu kali percobaan. Hasil klasifikasi menggunakan CNN 2-Dimensi dan Roboflow, nantinya akan dideteksi keberadaan rongga udaranya menggunakan YOLOv9 sehingga keberlangsungan pengujian deteksi menggunakan YOLOv9 sangat penting. Training YOLOv9 dilakukan dengan menggunakan dataset yang dilabeli melalui Roboflow. Dataset yang ada, dilakukan training dengan 10 epochs dan 8 batch. Dari hasil training, didapatkan hasil akurasi sebesar 0.979. Hasil metrics lainnya dapat dilihat pada gambar 4.57. Selain itu, didapatkan pula hasil confusion matrix dari data training pada Gambar 4.58. Dari hasil tersebut, dapat dilihat bahwa hasil deteksi menggunakan YOLOv9 memiliki tingkat akurasi yang cukup tinggi.

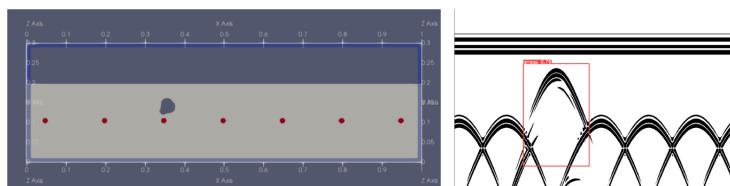


Gambar 4.57: Metrics YOLOv9

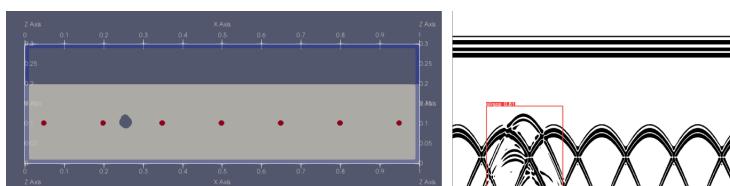
Confusion Matrix	
Predicted	Airgap
Airgap	0.98 1.00
Background	0.02

Gambar 4.58: Confussion Matrix YOLOv9

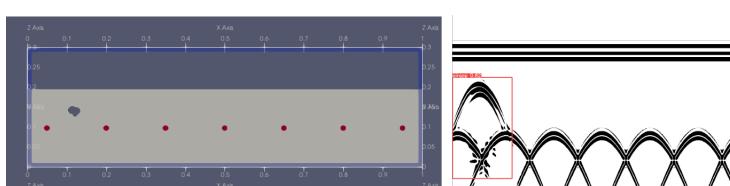
Hasil confussion matrix diatas dapat terbentuk karena pada training YOLOv9 ini hanya digunakan satu kelas yakni kelas airgap atau rongga udara. Hal ini dilakukan karena pada penggunaan YOLOv9 ditunjukkan untuk mendeteksi satu objek saja yakni rongga udara pada sinyal B-Scan gprMax. Keakurasi YOLOv9 tersebut dapat dilihat dari hasil pengujian berikut dimana YOLOv9 mampu mendeteksi rongga udara pada sinyal B-Scan gprMax dengan baik. Berikut ini adalah contoh hasil deteksi menggunakan YOLOv9 yang dapat dilihat pada gambar berikut:



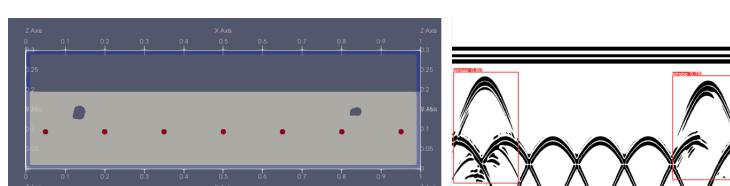
Gambar 4.59: Hasil Deteksi YOLOv9 1



Gambar 4.60: Hasil Deteksi YOLOv9 2



Gambar 4.61: Hasil Deteksi YOLOv9 3



Gambar 4.62: Hasil Deteksi YOLOv9 4

BAB V

PENUTUP

Pada bab ini akan dipaparkan kesimpulan dari hasil pengujian yang akan menjawab dari permasalahan yang diangkat dari pelaksanaan tugas akhir ini. Pada bab ini juga diapaparkan saran mengenai hal yang dapat dilakukan untuk mengembangkan penelitian kedepannya.

5.1 Kesimpulan

Berdasarkan hasil pengujian yang dilakukan selama pelaksanaan tugas akhir ini adalah sebagai berikut:

1. Model CNN 2D memiliki akurasi yang baik bila dataset dibagi menjadi 70% training, 15% validasi, dan 15% testing
2. Model dengan akurasi tertinggi didapatkan pada percobaan kedua dengan nilai akurasi sebesar 0.9964 dan training loss sebesar 0.0103. Didapatkan pula nilai f1 score sebesar 0.99714893
3. Hasil training cenderung memicu lonjakan baik kecil maupun besar yang disebabkan oleh dataset yang digunakan dimana dataset sinyal B-Scan dari simulasi gprMax pada kelas tanpa rongga udara memiliki bentuk yang hampir serupa dengan sedikit variasi
4. Hasil klasifikasi dengan Roboflow memiliki hasil yang tidak kalah baik dibandingkan dengan CNN 2D
5. Pendeksteksian dengan YOLOv9 memiliki akurasi yang baik sebesar 0.979 dapat mendeksteki rongga udara dengan baik

5.2 Saran

Berdasarkan hasil yang diperoleh dari penelitian ini maka saran yang dapat dipertimbangkan untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Memperbanyak variasi dari kedua kelas dataset yang digunakan
2. Mencoba model CNN 2D yang lebih beragam untuk mengetahui model yang lebih efektif dan efisien

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

- Bansal, P., & Ouda, A. (2022). Study on integration of fastapi and machine learning for continuous authentication of behavioral biometrics. *2022 International Symposium on Networks, Computers and Communications (ISNCC)*, 1–6. <https://doi.org/10.1109/ISNCC55209.2022.9851790>
- Barkataki, N., Mazumdar, S., Singha, P. B. D., Kumari, J., Tiru, B., & Sarma, U. (2021). Classification of soil types from gpr b scans using deep learning techniques. *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, 840–844. <https://doi.org/10.1109/RTEICT52294.2021.9573702>
- Brucal, S. G. E., de Jesus, L. C. M., Peruda, S. R., Samaniego, L. A., & Yong, E. D. (2023). Development of tomato leaf disease detection using yolov8 model via roboflow 2.0. *2023 IEEE 12th Global Conference on Consumer Electronics (GCCE)*, 692–694. <https://doi.org/10.1109/GCCE59613.2023.10315251>
- Carneiro, T., Medeiros Da Nóbrega, R. V., Nepomuceno, T., Bian, G.-B., De Albuquerque, V. H. C., & Filho, P. P. R. (2018). Performance analysis of google colabatory as a tool for accelerating deep learning applications. *IEEE Access*, 6, 61677–61685. <https://doi.org/10.1109/ACCESS.2018.2874767>
- Ciaglia, F., Zuppichini, F. S., Guerrie, P., McQuade, M., & Solawetz, J. (2022). Roboflow 100: A rich, multi-domain object detection benchmark.
- Contoli, C., & Lattanzi, E. (2023). A study on the application of tensorflow compression techniques to human activity recognition. *IEEE Access*, 11, 48046–48058. <https://doi.org/10.1109/ACCESS.2023.3276438>
- Craig Warren, I. G., Antonios Giannopoulos. (2016). Gprmax: Open source software to simulate electromagnetic wave propagation for ground penetrating radar. *Computer Physics Communications*.
- Daniels, D. J. (2004). *Ground penetrating radar - 2nd edition*. The Institution of Electrical Engineers London.
- Deepa, D., Sivasangari, A., Roonwal, R., & Nayan, R. (2023). Pothole detection using roboflow convolutional neural networks. *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 560–564. <https://doi.org/10.1109/ICICCS56967.2023.10142700>
- Diwan, T., Ani, A., & Tembhurne, J. (2022). Object detection using yolo: Challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, 82. <https://doi.org/10.1007/s11042-022-13644-y>
- F. M. Rodrigues, J. M., A. L. Moreno Junior. (2022). Short columns composed of concrete-filled steel tubes in a fire situation – numerical model and the “air-gap” effect. *IBRACON Structures and Material Journal*.
- Feistkorn, S. (2016). Rebar detection — pod approach to determine the reliability of gpr systems and to quantify the influence of different material parameters. *2016 16th International Conference on Ground Penetrating Radar (GPR)*.
- Goiannopoulos, A. (2005). Modelling ground penetrating radar by gprmax. *Construction and Building Materials*.

- Goncharov, A. (2021). Increase of impact strength and frost resistance of concrete hydrotechnical piles at deaeration of concrete mixture with air-entraining admixtures. *IOP Conference Series: Materials Science and Engineering*.
- Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*.
- Jeply Murdiaman Guci, d. A. N., Rully Angraeni Safitri. (2021). Perencanaan bangunan gedung tahan gempa 11 lantai dengan sistem ganda. *Structur: Jurnal Teknik Sipil*.
- Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A review of yolo algorithm developments [The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021): Developing Global Digital Economy after COVID-19]. *Procedia Computer Science*, 199, 1066–1073. [https://doi.org/https://doi.org/10.1016/j.procs.2022.01.135](https://doi.org/10.1016/j.procs.2022.01.135)
- Karisma, D. A. (2022). The optimum vibration of the compressive strength of concrete specimen. *INERSIA*.
- Keiron O Shea, R. N. (2015). An introduction to convolutional neural networks. *ArXiv e-prints*.
- Kimm, H., Paik, I., & Kimm, H. (2021). Performance comparision of tpu, gpu, cpu on google colaboratory over distributed deep learning. *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, 312–319. <https://doi.org/10.1109/MCSoC51149.2021.00053>
- Koprawi, M. (2020). Parallel computation in uncompressed digital images using computer unified device architecture and open computing language. *PIKSEL : Penelitian Ilmu Komputer Sistem Embedded and Logic*, 8, 31–38. <https://doi.org/10.33558/piksel.v8i1.2017>
- Lazuardy, M. F. S., & Anggraini, D. (2022). Modern front end web architectures with react. js and next. js. *Research Journal of Advanced Engineering and Science*, 7(1), 132–141.
- Leung, R. (2023). An experience with pycuda: Refactoring an existing implementation of a ray-surface intersection algorithm.
- Pang, B., Nijkamp, E., & Wu, Y. (2019). Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics*, 45, 107699861987276. <https://doi.org/10.3102/1076998619872761>
- Patel, V. (2023). Analyzing the impact of next.js on site performance and seo. *International Journal of Computer Applications Technology and Research*, 12, 24–27. <https://doi.org/10.7753/IJCATR1210.1004>
- Peng, K., Wu, L., Zandi, Y., Agdas, A., Majdi, A., Denic, N., Zakić, A., Ebid, A., Khadimallah, A., & Elhosiny Ali, H. (2022). Application of polyacrylic hydrogel in durability and reduction of environmental impacts of concrete through ann. *Gels*.
- Persico, R. (2014). *Introduction to ground penetrating radar: Inverse scattering and data processing*. Wiley.
- Ruyue Xin, Y. S., Jiang Zhang. (2020). Complex network classification with convolutional neural network. *Tsinghua Science and Technology*.
- Scarsbrook, J. D., Utting, M., & Ko, R. L. (2023). Typescripts evolution: An analysis of feature adoption over time. *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, 109–114. <https://doi.org/10.1109/MSR59073.2023.00027>
- Tang, W., Lu, Y., Xiao, N., Liu, F., & Chen, Z. (2017). Accelerating redis with rdma over infiniband, 472–483. https://doi.org/10.1007/978-3-319-61845-6_47

- Thu, D. T. H., Nguyen, D.-A., & Hung, P. N. (2021). Automated test data generation for type-script web applications. *2021 13th International Conference on Knowledge and Systems Engineering (KSE)*, 1–6. <https://doi.org/10.1109/KSE53942.2021.9648782>
- Trela, C., Kind, T., & Schubert, M. (2015). Detection of air voids in concrete by radar in transmission mode. *2015 8th International Workshop on Advanced Ground Penetrating Radar (IWAGPR)*.
- Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2024). Yolov9: Learning what you want to learn using programmable gradient information.
- Wang, H. (2023). *Introduction to computer programming with python*. Remix Athabasca University.
- Wang, Z., Bu, D., Sun, A., Gou, S., Wang, Y., & Chen, L. (2022). An empirical study on bugs in python interpreters. *IEEE Transactions on Reliability*.
- Warren, C., Giannopoulos, A., & Giannakis, I. (2015). An advanced gpr modelling framework: The next generation of gprmax. *2015 8th International Workshop on Advanced Ground Penetrating Radar (IWAGPR)*.
- Wu, J. (2017). *Introduction to convolutional neural networks*. LAMDA Group.
- Zaki, A. K., & M., I. (2015). A novel redis security extension for nosql database using authentication and encryption. *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 1–6. <https://doi.org/10.1109/ICECCT.2015.7226101>
- Zhang, P., Xing, L., Yang, N., Tan, G., Liu, Q., & Zhang, C. (2018). Redis++: A high performance in-memory database based on segmented memory management and two-level hash index. *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, 840–847. <https://doi.org/10.1109/BDCloud.2018.00125>
- Zhu, J., & Popovics, J. (2007). Imaging concrete structures using air-coupled impact-echo. *Journal of Engineering Mechanics*, 133, 628. [https://doi.org/10.1061/\(ASCE\)0733-9399\(2007\)133:6\(628\)](https://doi.org/10.1061/(ASCE)0733-9399(2007)133:6(628))

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Kode Program

Program Contoh Data Input gprMax

Program 5.1: Contoh File Input gprMax

```
1 #title: B-scan from rebar in concrete
2
3 #domain: 1.0 0.3 0.001
4 #dx_dy_dz: 0.001 0.001 0.001
5 #time_window: 3e-9
6
7 #material: 7.0 0.0 1.0 0.0 concrete
8 #material: 12.0 1.0e6 1.0 0.0 steel
9
10 #waveform: ricker 1 4.5e9 my_ricker
11 #src_steps: 0.001 0 0
12 #rx_steps: 0.001 0 0
13 #box: 0.0 0.0 0.0 1.0 0.203 0.001 concrete
14 #hertzian_dipole: z 0.010 0.203 0 my_ricker
15 #rx: 0.050 0.203 0
16 #cylinder: 0.154 0.092 0.0 0.154 0.092 0.001 0.007 steel
17 #cylinder: 0.304 0.092 0.0 0.304 0.092 0.001 0.007 steel
18 #cylinder: 0.454 0.092 0.0 0.454 0.092 0.001 0.007 steel
19 #cylinder: 0.604 0.092 0.0 0.604 0.092 0.001 0.007 steel
20 #cylinder: 0.754 0.092 0.0 0.754 0.092 0.001 0.007 steel
21 #cylinder: 0.904 0.092 0.0 0.904 0.092 0.001 0.007 steel
22 #cylinder: 0.054 0.092 0.0 0.054 0.092 0.001 0.007 steel
23 #cylinder: 0.416 0.129 0.0 0.416 0.129 0.001 0.013 free_space
24 #cylinder: 0.425 0.121 0.0 0.425 0.121 0.001 0.009 free_space
25 #cylinder: 0.416 0.122 0.0 0.416 0.122 0.001 0.015 free_space
26 #cylinder: 0.505 0.120 0.0 0.505 0.120 0.001 0.015 free_space
27 #cylinder: 0.499 0.117 0.0 0.499 0.117 0.001 0.010 free_space
28 #cylinder: 0.499 0.110 0.0 0.499 0.110 0.001 0.014 free_space
29 #cylinder: 0.495 0.116 0.0 0.495 0.116 0.001 0.011 free_space
30 #cylinder: 0.503 0.120 0.0 0.503 0.120 0.001 0.007 free_space
```

Program Generate File Input gprMax

Program 5.2: Program untuk Menghasilkan File Input gprMax

```
1 import os
2 import random
3
4 def generate_airgap_cylinders(file, box_height):
```

```

5   # Define bounds for airgap cylinder placement
6   x_min, x_max = 0.05, 0.95
7   y_min, y_max = 0.090, box_height - 0.05 # Adjusted Y ←
        bounds to ensure cylinders are within the concrete ←
        slab
8
9   airgap_count = random.randint(1, 2) # Generate 1 or 2 ←
        airgaps per file
10  for _ in range(airgap_count):
11      # Each airgap formed by 3 to 5 cylinders
12      cylinder_count = random.randint(3, 5)
13
14      # Initial positions for the first cylinder in an ←
            airgap
15      x_position = random.uniform(x_min, x_max)
16      y_position = random.uniform(y_min, y_max)
17
18      for _ in range(cylinder_count):
19          radius = random.uniform(0.010, 0.035) / 2 # ←
                Radius between 0.010m and 0.035m
20          # Write cylinder, then adjust positions for the ←
                next one
21          file.write(f"#cylinder: {x_position:.3f} {←
                y_position:.3f} 0.0 {x_position:.3f} {←
                y_position:.3f} 0.001 {radius:.3f} free_space←
                n")
22
23      # Tighten adjustments for X and Y to encourage ←
            collisions
24      x_adjust = random.uniform(-0.010, 0.010)
25      y_adjust = random.uniform(-0.010, 0.010)
26
27      # Update positions ensuring they remain within ←
            defined bounds
28      x_position = min(max(x_position + x_adjust, x_min←
                ), x_max)
29      y_position = min(max(y_position + y_adjust, y_min←
                ), y_max)
30
31 def generate_gprMax_files():
32     base_content = """#title: B-scan from rebar in concrete
33
34 #domain: 1.0 0.3 0.001
35 #dx_dy_dz: 0.001 0.001 0.001
36 #time_window: 3e-9
37
38 #material: 7.0 0.0 1.0 0.0 concrete
39 #material: 12.0 1.0e6 1.0 0.0 steel
40

```

```

41 #waveform: ricker 1 4.5e9 my_ricker
42 #src_steps: 0.001 0 0
43 #rx_steps: 0.001 0 0
44 """
45
46     parent_folder = "airgap"
47     os.makedirs(parent_folder, exist_ok=True)
48
49     box_height_start = 0.190
50     box_height_end = 0.210
51     total_files = 2000 # Adjust total files generated to ←
52     2000
53
54     for file_number in range(1, total_files + 1):
55         box_height = random.uniform(box_height_start, ←
56             box_height_end)
57         folder_name = os.path.join(parent_folder, str(←
58             file_number))
59         os.makedirs(folder_name, exist_ok=True)
60         filename = os.path.join(folder_name, f"data{←
61             file_number}.in")
62
63         with open(filename, "w") as file:
64             file.write(base_content)
65             file.write(f"#box: 0.0 0.0 0.0 1.0 {box_height:.3f} ←
66                 0.001 concrete\n")
67             file.write(f"#hertzian_dipole: z 0.010 ←
68                 box_height:.3f} 0 my_ricker\n")
69             file.write(f"#rx: 0.050 {box_height:.3f} 0\n")
70
71             # Generate steel cylinders with a common Y ←
72             # position
73             common_y_position = random.uniform(0.090, 0.110) ←
74                 # Fixed Y position for steel cylinders within←
75                 the range
76             for position in [0.050, 0.200, 0.350, 0.500, ←
77                 0.650, 0.800, 0.950]:
78                 new_x = (position + (file_number * 0.001)) % ←
79                     1.0
80                 file.write(f"#cylinder: {new_x:.3f} {←
81                     common_y_position:.3f} 0.0 {new_x:.3f} {←
82                     common_y_position:.3f} 0.001 0.007 steel\n")
83
84             # Generate abstract airgap shapes with tightened ←
85             # X and Y adjustments
86             generate_airgap_cylinders(file, box_height)
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779

```

```
74     print(f"Generated file: {folder_name}/data{←  
    file_number}.in")  
75  
76 generate_gprMax_files()
```

Program Setup GPU

Program 5.3: Program Setup GPU

```
1 #!/bin/bash  
2  
3 # Creating Miniconda directory  
4 echo "Creating Miniconda directory..."  
5 mkdir -p ~/miniconda3  
6  
7 # Downloading Miniconda installer  
8 echo "Downloading Miniconda installer..."  
9 wget https://repo.anaconda.com/miniconda/Miniconda3-latest-←  
    Linux-x86_64.sh -O ~/miniconda3/miniconda.sh  
10  
11 # Installing Miniconda  
12 echo "Installing Miniconda..."  
13 bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3  
14  
15 # Removing the installer to save space  
16 echo "Removing Miniconda installer..."  
17 rm -rf ~/miniconda3/miniconda.sh  
18  
19 # Initializing Conda for bash  
20 echo "Initializing Conda for bash..."  
21 ~/miniconda3/bin/conda init bash  
22  
23 source ./bashrc  
24  
25 # Updating Conda  
26 echo "Updating Conda..."  
27 conda update conda -y  
28  
29 # Installing Git  
30 echo "Installing Git..."  
31 conda install git -y  
32  
33 # Cloning the gprMax repository  
34 echo "Cloning gprMax repository..."  
35 git clone https://github.com/MaulanaGilang/gprMax.git  
36  
37 # Changing directory to gprMax  
38 echo "Changing directory to gprMax..."  
39 cd gprMax  
40
```

```

41 # Creating Conda environment from file
42 echo "Creating Conda environment from conda_env.yml..."
43 conda env create -f conda_env.yml
44
45 # Activating the gprMax environment
46 echo "Activating gprMax environment..."
47 conda activate gprMax
48
49 # Building gprMax
50 echo "Building gprMax..."
51 python setup.py build
52
53 # Installing gprMax
54 echo "Installing gprMax..."
55 python setup.py install
56
57 # Installing PyCUDA
58 echo "Installing PyCUDA..."
59 pip install pycuda
60
61 echo "Setup complete. gprMax is ready to use."

```

Program Otomasi Generate Data Melalui gprMax dengan GPU

Program 5.4: Program Otomasi Generate Data gprMax dengan GPU

```

1 import argparse
2 import os
3 import subprocess
4 from shutil import move
5 import re
6 import matplotlib.pyplot as plt
7 from tools.plot_Bscan import get_output_data, mpl_plot
8 from PIL import Image
9
10 def natural_keys(text):
11     """Helper function for natural sorting (sorts text with ↵
12         embedded numbers correctly)."""
13     return [int(c) if c.isdigit() else c for c in re.split('↵
14         (\d+)', text)]
15
16 def list_in_files_recursive(directory):
17     """List all .in files recursively in the given directory.↵
18         """
19     in_files = {}
20     for root, dirs, files in os.walk(directory):
21         for f in files:
22             if f.endswith('.in'):
23                 key = os.path.splitext(f)[0]
24                 in_files[key] = os.path.join(root, f)

```

```
22     return in_files
23
24 def run_gprMax(file_path, n, use_gpu):
25     """Run gprMax command on a file."""
26     gpu_flag = '-gpu' if use_gpu else ''
27     command = f'python -m gprMax {file_path} -n {n} {gpu_flag}'
28     print("Running command: ", command)
29     subprocess.run(command, shell=True, check=True)
30
31 def merge_output_files(directory, file_without_ext):
32     """Merge output files in the directory based on the .in
33         filename and delete the original .out files."""
34     subprocess.run(f'python -m tools.outputfiles_merge {{directory}}/{{file_without_ext}}', shell=True, check=True)
35
36     # New code to delete original .out files
37     pattern = f'^{file_without_ext}\d+\.out$'
38     for f in os.listdir(directory):
39         if re.match(pattern, f):
40             os.remove(os.path.join(directory, f))
41
42 def ensure_directory_exists(directory):
43     """Ensure the specified directory exists, create if it
44         doesn't."""
45     if not os.path.exists(directory):
46         os.makedirs(directory)
47
48 def move_output_file(source_path, dest_directory):
49     """Move the file to the specified output directory,
50         creating the directory if necessary."""
51     ensure_directory_exists(dest_directory)
52     dest_path = os.path.join(dest_directory, os.path.basename(source_path))
53     move(source_path, dest_path)
54
55     return dest_path
56
57 def process_file(file_path, rxnumber, rxcomponent,
58                 non_greyscale_dir, greyscale_dir):
59     """Generate plots for a .out file, save color and cropped
60         images (without converting to grayscale)."""
61     outputdata, dt = get_output_data(file_path, rxnumber,
62                                     rxcomponent)
63     plt_figure = mpl_plot(file_path, outputdata, dt, rxnumber,
64                           rxcomponent)
65     plt_figure.axis('off')
```

```

60     savefile = os.path.splitext(os.path.basename(file_path))←
61         [0]
62     image_path_with_colorbar = os.path.join(non_greyscale_dir←
63         , savefile + '.jpg')
64     cropped_image_path = os.path.join(greyscale_dir, savefile←
65         + '_cropped.jpg') # Renamed variable for clarity
66
67     plt.figure.savefig(image_path_with_colorbar, dpi=150, ←
68         format='JPEG', bbox_inches='tight', pad_inches=0.1)
69     plt.close()
70
71     color_bar_width = 300 # Adjusted to 300 pixels as per ←
72         requirement
73     left_crop = 20 # Left side crop
74     bottom_crop = 20 # Bottom side crop adjustment
75
76     image = Image.open(image_path_with_colorbar)
77     cropped_image = image.crop((left_crop, 0, image.width - ←
78         color_bar_width - left_crop, image.height - ←
79         bottom_crop))
80     cropped_image.save(cropped_image_path) # Save the ←
81         cropped image directly
82
83     def main(args):
84         in_files = list_in_files_recursive(args.input)
85         sorted_keys = sorted(in_files.keys(), key=natural_keys)
86
87         start = args.start if args.start is not None else 0
88         end = args.end if args.end is not None else len(←
89             sorted_keys)
90
91         processed_count = 0
92         processed_files = [] # Keep track of processed _merged.←
93             out files
94
95         rxnumber = 1
96         rxcomponent = 'Ez'
97         non_greyscale_dir = 'images/non'
98         greyscale_dir = 'images/greyscale'
99         ensure_directory_exists(non_greyscale_dir)
100        ensure_directory_exists(greyscale_dir)
101
102        for key in sorted_keys[start:end]:
103            file_path = in_files[key]
104            run_gprMax(file_path, args.n, args.gpu)
105
106            if args.merge:
107                merge_output_files(os.path.dirname(file_path), ←
108                    key)

```

```

98     merged_file = key + '_merged.out'
99     # Adjust merged_path to use the output directory ←
100    where the file has been moved
100    merged_path = os.path.join(os.path.dirname(←
101        file_path), merged_file) # Adjusted to ←
101        reflect the correct directory
101    new_path = move_output_file(merged_path, args.←
101        output) # This operation may be redundant if ←
101        merged_path already points to the correct ←
101        location
102
103    process_file(new_path, rxnumber, rxcomponent, ←
103        non_greyscale_dir, greyscale_dir)
104    processed_files.append(new_path)
105    processed_count += 1
106
107    # After all files processed, check for any remaining ←
107    _merged.out files to process
108    if processed_files:
109        for file in processed_files:
110            os.remove(file)
111
112 if __name__ == "__main__":
113     parser = argparse.ArgumentParser(description='Process .in←
113         files with gprMax.')
114     parser.add_argument('-i', '--input', required=True, help=←
114         'Input folder path')
115     parser.add_argument('--start', type=int, help='Start ←
115         index (optional)')
116     parser.add_argument('--end', type=int, help='End index (←
116         optional)')
117     parser.add_argument('--merge', action='store_true', ←
117         default=False, help='Merge output files')
118     parser.add_argument('--gpu', action='store_true', default=←
118         False, help='Use GPU for processing')
119     parser.add_argument('-n', required=True, type=int, help=←
119         'Number of iterations')
120     parser.add_argument('-o', '--output', required=True, help=←
120         'Output folder path')
121     args = parser.parse_args()
122
123     main(args)

```

Program Generate Data Melalui Augmentasi

Program 5.5: Program Generate Data Melalui Augmentasi

```

1 from PIL import Image
2 import os
3 from tqdm import tqdm

```

```

4
5 # Load the images
6 line_path = os.path.join('noairgap', 'line.png')
7 wave_path = os.path.join('noairgap', 'wave.png')
8
9 line_img = Image.open(line_path)
10 wave_img = Image.open(wave_path)
11
12 # Create a new image with the same size as the line image but←
    with alpha to composite images over it
13 result_img = Image.new('RGBA', (1860, 1160), (255, 255, 255, ←
    0))
14
15 # Paste the line image at (0, 0)
16 result_img.paste(line_img, (0, 0), line_img)
17
18 # Function to apply transformations and save the results
19 def transform_and_save(wave_img, result_img, shift_x, shift_y←
    , crop_size, file_name):
20     temp_img = result_img.copy()
21     # Paste the wave image at (0, 0) with the specified ←
        horizontal and vertical shifts
22     temp_img.paste(wave_img, (shift_x, shift_y), wave_img)
23     temp_img.paste(line_img, (0, 0), line_img)
24     # Crop the image
25     cropped_img = temp_img.crop((0, 0, crop_size[0], ←
        crop_size[1]))
26     # Save the image
27     cropped_img.save(file_name)
28
29 crop_size = (1860, 1160)
30
31 wave_width, wave_height = wave_img.size
32 horizontal_step = 5
33 vertical_step = 20
34 vertical_limit = 100
35
36 print("It will generate", (wave_width - 1860) // ←
    horizontal_step * (vertical_limit // vertical_step + 1), "←
    images")
37
38 counter = 0
39 for horizontal_shift in tqdm(range(-wave_width + 1860, 0, ←
    horizontal_step)):
40     for vertical_shift in range(0, vertical_limit + 1, ←
        vertical_step):
41         file_name = f'noairgap/fix/data_{counter}.png'
42         transform_and_save(wave_img, result_img, horizontal_shift←
            , vertical_shift, crop_size, file_name)

```

43 counter += 1

Program Preprocessing Data

Program 5.6: Program untuk Memformat Ulang Tipe File

```
1 from PIL import Image
2 import os
3 def convert_directory_to_jpeg(input_dir, output_base_dir):
4     for root, dirs, files in os.walk(input_dir):
5         for file in files:
6             if file.lower().endswith('.png', '.jpg', '.jpeg'←
7                 , '.tiff', '.bmp', '.gif')):
8                 input_path = os.path.join(root, file)
9                 relative_path = os.path.relpath(root, ←
10                   input_dir)
11                 output_dir = os.path.join(output_base_dir, ←
12                   relative_path)
13
14                 if not os.path.exists(output_dir):
15                     os.makedirs(output_dir)
16
17                 output_file = os.path.splitext(os.path.join(←
18                   output_dir, file))[0] + '.jpg'
19
20                 # Load the image
21                 image = Image.open(input_path)
22
23                 # If the image has an alpha channel, convert ←
24                 # it to RGB and fill transparency with white
25                 if image.mode in ('RGBA', 'LA') or (image.←
26                   mode == 'P' and 'transparency' in image.←
27                   info):
28                     # Create a white background image
29                     background = Image.new('RGB', image.size,←
30                         (255, 255, 255))
31                     # Paste the image onto the background
32                     background.paste(image, mask=image.split←
33                         ()[3]) # 3 is the alpha channel
34                     image = background
35
36                     # Save the image in JPEG format
37                     image.save(output_file, 'JPEG')
38
39                     # print(f"Image saved as {output_file}")
40                     break
41
42
43 # Example usage
44 input_dir = 'dataset' # Change this to your input directory ←
45 path
```

```

35 output_base_dir = 'out' # Change this to your desired output←
    base directory
36
37 convert_directory_to_jpeg(input_dir, output_base_dir)

```

Program 5.7: Program untuk Binarization Gambar

```

1 import cv2
2 import os
3 from pathlib import Path
4 from tqdm import tqdm
5
6 def process_image(image_path, output_dir):
7     # Read the image
8     img = cv2.imread(str(image_path))
9
10    # Convert to grayscale
11    gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
12
13    # Apply Otsu's thresholding
14    _, binary_image = cv2.threshold(gray_image, 0, 255, cv2.←
15        THRESH_BINARY + cv2.THRESH_OTSU)
16
17    # Construct the output path
18    output_path = output_dir / image_path.relative_to(←
19        input_dir)
20    output_path.parent.mkdir(parents=True, exist_ok=True)
21
22    # Save the binary image
23    cv2.imwrite(str(output_path), binary_image)
24
25 def process_directory(input_dir, output_dir):
26     input_dir = Path(input_dir)
27     output_dir = Path(output_dir)
28
29     # Prepare a list of image files to process
30     image_files = [Path(root) / file for root, _, files in os.←
31         .walk(input_dir) for file in files if file.lower().←
32         endswith('.png', '.jpg', '.jpeg', '.bmp')]
33
34     # Process each image with a progress bar
35     for image_path in tqdm(image_files, desc="Processing ←
36         images"):
37         process_image(image_path, output_dir)
38
39 # Specify the input and output directories
40 input_dir = 'dataset'
41 output_dir = 'out'
42
43 # Process the directory

```

```
39 process_directory(input_dir, output_dir)
```

Program CNN 2-Dimensi Percobaan Pertama

Program 5.8: Program CNN 2-Dimensi Percobaan Pertama

```
1 # -*- coding: utf-8 -*-
2 """CNN TA v1 d1.ipynb
3
4 Automatically generated by Colab.
5
6 Original file is located at
7     https://colab.research.google.com/drive/1↔
8         DBPtAJpUwnFAbiazC_suoOmEWDtla0wC
9 """
10 import tensorflow as tf
11 tf.__version__
12
13 from tensorflow.keras.models import Sequential
14 from tensorflow.keras.layers import Conv2D, MaxPooling2D, ←
15     Flatten, Dense, Dropout
16 from tensorflow.keras.preprocessing.image import ←
17     ImageDataGenerator
18 from tensorflow.keras.metrics import Precision, Recall
19 import os
20 import time
21 import time
22 from sklearn.metrics import confusion_matrix
23 import seaborn as sns
24 import matplotlib.pyplot as plt
25 import numpy as np
26
27 # Set the seed for TensorFlow's random number generator
28 np.random.seed(42)
29 tf.random.set_seed(42)
30
31 from google.colab import drive
32 drive.mount('/content/drive')
33
34 datasetDir = "/content/drive/MyDrive/KeperluanTA/another/out"
35
36 train_datagen = ImageDataGenerator(
37     rescale=1.0/255,
38     shear_range=0.2,
39     zoom_range=0.2,
40     fill_mode="nearest",
41     validation_split=0.22225
42 )
```

```

42 val_datagen = ImageDataGenerator(
43     rescale=1.0/255,
44     validation_split=0.22225
45 )
46
47 train_generator = train_datagen.flow_from_directory(
48     datasetDir,
49     target_size=(330, 540),
50     batch_size=10,
51     class_mode="binary", # Set class_mode to "categorical" ←
52         for multi-class classification
53     color_mode="grayscale", # Generate grayscale images
54     subset="training"
55 )
56
57 validation_generator = val_datagen.flow_from_directory(
58     datasetDir,
59     target_size=(330, 540),
60     batch_size=10,
61     class_mode="binary", # Set class_mode to "categorical" ←
62         for multi-class classification
63     color_mode="grayscale", # Generate grayscale images
64     subset="validation"
65 )
66
67 print(train_generator.class_indices)
68
69 train_generator.image_shape
70
71 # Now your model definition follows
72 model = Sequential()
73
74 # First Convolutional Block
75 model.add(Conv2D(32, (3, 3), activation="relu", input_shape=
76                 =(330, 540, 1)))
77 model.add(MaxPooling2D((2, 2)))
78
79 # Second Convolutional Block
80 model.add(Conv2D(64, (3, 3), activation="relu"))
81 model.add(MaxPooling2D((2, 2)))
82
83 # Third Convolutional Block
84 model.add(Conv2D(128, (3, 3), activation="relu"))
85 model.add(MaxPooling2D((2, 2)))
86
87 # Fourth Convolutional Block
88 model.add(Conv2D(256, (3, 3), activation="relu"))
89 model.add(MaxPooling2D((2, 2)))

```

```

88 # Flattening and Fully Connected Layers
89 model.add(Flatten())
90 model.add(Dense(256, activation="relu"))
91 model.add(Dropout(0.5)) # Dropout layer to reduce ←
    overfitting
92 model.add(Dense(128, activation="relu"))
93 model.add(Dropout(0.5)) # Another dropout layer
94
95 # Output Layer for binary classification
96 model.add(Dense(1, activation="sigmoid"))
97
98 model.summary()
99
100 model.compile(
101     optimizer="adam",
102     loss="binary_crossentropy",
103     metrics=["accuracy"]
104 )
105
106 from tensorflow.keras.callbacks import EarlyStopping
107
108 # Define EarlyStopping callback
109 early_stopping = EarlyStopping(
110     monitor='val_loss',          # Metric to monitor
111     patience=10,                # Number of epochs with no ←
        improvement after which training will be stopped
112     verbose=1,                  # To log when training is being ←
        stopped
113     mode='min',                # Stops training when the ←
        quantity monitored has stopped decreasing
114     restore_best_weights=True   # Restores model weights from ←
        the epoch with the best value of the monitored ←
        quantity
115 )
116
117 # Fit the model with the EarlyStopping callback
118 history = model.fit(
119     train_generator,
120     validation_data=validation_generator,
121     epochs=100,
122     validation_steps=80,
123     verbose=1,
124     callbacks=[early_stopping] # Include the callback in the←
        list
125 )
126
127 model.evaluate(validation_generator)
128
129 model.evaluate(train_generator)

```

```

130
131 # Specify the directory path
132 save_dir = '/content/drive/MyDrive/KeperluanTA/another/1rev'
133
134 # Create the directory if it doesn't exist
135 os.makedirs(save_dir, exist_ok=True)
136
137 # Save the model to the specified directory
138 model.save(save_dir + 'model.h5')
139
140 model.save("model.h5")
141
142 import matplotlib.pyplot as plt
143
144 # Plot training & validation accuracy values
145 plt.figure(figsize=(12, 4))
146
147 plt.subplot(1, 2, 1)
148 plt.plot(history.history['accuracy'])
149 plt.plot(history.history['val_accuracy'])
150 plt.title('Model Accuracy')
151 plt.ylabel('Accuracy')
152 plt.xlabel('Epoch')
153 plt.legend(['Train', 'Validation'], loc='upper left')
154
155 # Plot training & validation loss values
156 plt.subplot(1, 2, 2)
157 plt.plot(history.history['loss'])
158 plt.plot(history.history['val_loss'])
159 plt.title('Model Loss')
160 plt.ylabel('Loss')
161 plt.xlabel('Epoch')
162 plt.legend(['Train', 'Validation'], loc='upper left')
163
164 plt.show()
165
166 import time
167
168 from sklearn.metrics import confusion_matrix
169 import seaborn as sns
170 import matplotlib.pyplot as plt
171 import numpy as np
172
173 # Create new generators without shuffling for evaluating ←
174     confusion matrix
175 eval_train_generator = train_datagen.flow_from_directory(
176     datasetDir,
177     target_size=(330, 540),
178     batch_size=10, # Adjust this based on your setup

```

```

178     class_mode="binary",  # Adjusted for binary ↵
179         classification
180     color_mode="grayscale",
181     shuffle=False,  # Important for matching predictions to ↵
182         labels
183     subset="training"
184 )
185
186 eval_validation_generator = val_datagen.flow_from_directory(
187     datasetDir,
188     target_size=(330, 540),
189     batch_size=10,  # Match the batch size used for training/←
190         validation
191     class_mode="binary",  # Adjusted for binary ↵
192         classification
193     color_mode="grayscale",
194     shuffle=False,  # Important for matching predictions to ↵
195         labels
196     subset="validation"
197 )
198
199 # Start timer for inference
200 start_time = time.time()
201
202 # Predict the data
203 train_predictions = model.predict(eval_train_generator, ←
204     verbose=1)
205 validation_predictions = model.predict(←
206     eval_validation_generator, verbose=1)
207
208 # End timer and calculate inference time
209 inference_time = time.time() - start_time
210 print("Inference time for the test set: {:.2f} seconds".←
211     format(inference_time))
212
213 # Convert probabilities to binary predictions based on a 0.5 ←
214     threshold
215 train_pred_classes = (train_predictions > 0.5).astype(int).←
216     reshape(-1)
217 validation_pred_classes = (validation_predictions > 0.5).←
218     astype(int).reshape(-1)
219
220 # True labels (already in binary format)
221 train_true_classes = eval_train_generator.classes
222 validation_true_classes = eval_validation_generator.classes
223
224 # Compute confusion matrices
225 train_cm = confusion_matrix(train_true_classes, ←
226     train_pred_classes)

```

```

214 validation_cm = confusion_matrix(validation_true_classes, ←
    validation_pred_classes)
215
216 # Plotting the confusion matrices
217 fig, ax = plt.subplots(1, 2, figsize=(12, 5))
218
219 sns.heatmap(train_cm, annot=True, fmt='d', cmap='Blues', ax=←
    ax[0])
220 ax[0].set_title('Training Confusion Matrix')
221 ax[0].set_xlabel('Predicted Labels')
222 ax[0].set_ylabel('True Labels')
223 ax[0].set_xticklabels(['Negative', 'Positive'])
224 ax[0].set_yticklabels(['Negative', 'Positive'])
225
226 sns.heatmap(validation_cm, annot=True, fmt='d', cmap='Greens'←
    , ax=ax[1])
227 ax[1].set_title('Validation Confusion Matrix')
228 ax[1].set_xlabel('Predicted Labels')
229 ax[1].set_ylabel('True Labels')
230 ax[1].set_xticklabels(['Negative', 'Positive'])
231 ax[1].set_yticklabels(['Negative', 'Positive'], va='center')
232
233 plt.tight_layout()
234 plt.show()
235
236 import time
237 from sklearn.metrics import confusion_matrix
238 import seaborn as sns
239 import matplotlib.pyplot as plt
240 import numpy as np
241
242 testDir = "/content/drive/MyDrive/KeperluanTA/another/testout←
    "
243
244 # Set up the test data generator
245 test_datagen = ImageDataGenerator(rescale=1.0/255)
246
247 test_generator = test_datagen.flow_from_directory(
248     testDir, # Directory with test images
249     target_size=(330, 540),
250     batch_size=10, # Adjust based on your setup
251     class_mode="binary", # Ensure this matches your label ←
        setup
252     color_mode="grayscale",
253     shuffle=False # Important for matching predictions to ←
        labels
254 )
255
256 # Start timer for inference

```

```

257 start_time = time.time()
258
259 # Predict the data
260 test_predictions = model.predict(test_generator, verbose=1)
261
262 # End timer and calculate inference time
263 inference_time = time.time() - start_time
264 print("Inference time for the test set: {:.2f} seconds".format(inference_time))
265
266 # Convert probabilities to binary predictions based on a 0.5 threshold
267 test_pred_classes = (test_predictions > 0.5).astype(int).reshape(-1)
268
269 # True labels (already in binary format)
270 test_true_classes = test_generator.classes
271
272 # Compute the confusion matrix
273 test_cm = confusion_matrix(test_true_classes, test_pred_classes)
274
275 # Plotting the confusion matrix
276 plt.figure(figsize=(6, 5))
277 sns.heatmap(test_cm, annot=True, fmt='d', cmap='Purples')
278 plt.title('Test Confusion Matrix')
279 plt.xlabel('Predicted Labels')
280 plt.ylabel('True Labels')
281 plt.xticks([0.5, 1.5], ['Negative', 'Positive'])
282 plt.yticks([0.5, 1.5], ['Negative', 'Positive'], va='center')
283 plt.show()
284
285 import tensorflow as tf
286 from tensorflow.keras.metrics import Precision, Recall
287
288 # Define a custom F1 score metric
289 class F1Score(tf.keras.metrics.Metric):
290     def __init__(self, name='f1_score', **kwargs):
291         super(F1Score, self).__init__(name=name, **kwargs)
292         self.precision = Precision()
293         self.recall = Recall()
294
295     def update_state(self, y_true, y_pred, sample_weight=None):
296         self.precision.update_state(y_true, y_pred, sample_weight)
297         self.recall.update_state(y_true, y_pred, sample_weight)
298

```

```

299     def result(self):
300         p = self.precision.result()
301         r = self.recall.result()
302         return 2 * ((p * r) / (p + r + tf.keras.backend.←
303                     epsilon())))
303
304     def reset_state(self):
305         self.precision.reset_state()
306         self.recall.reset_state()
307
308 # Instantiate the F1Score metric for training, validation, ←
309 # and test
309 f1_score_train = F1Score()
310 f1_score_val = F1Score()
311 f1_score_test = F1Score()
312
313 # Update the state of the F1Score metric with training data
314 f1_score_train.update_state(train_true_classes, ←
315                         train_pred_classes)
315 # Update the state of the F1Score metric with validation data
316 f1_score_val.update_state(validation_true_classes, ←
317                           validation_pred_classes)
317 # Update the state of the F1Score metric with test data
318 f1_score_test.update_state(test_true_classes, ←
319                           test_pred_classes)
320
320 # Calculate and print the F1 scores
321 print("Calculated F1 Score for Training Set:", f1_score_train.←
322       .result().numpy())
322 print("Calculated F1 Score for Validation Set:", f1_score_val.←
323       .result().numpy())
323 print("Calculated F1 Score for Test Set:", f1_score_test.←
324       .result().numpy())
324
325 import os
326 import matplotlib.pyplot as plt
327 from tensorflow.keras.preprocessing.image import ←
328   ImageDataGenerator
328 from PIL import Image, ImageDraw
329
330 # Assuming `model` and `test_generator` have already been ←
331 # defined as in your original code.
331
332 # Create a directory to save the annotated images
333 save_dir = '/content/drive/MyDrive/Deteksi/Eks1'
334 os.makedirs(save_dir, exist_ok=True)
335
336 # Iterate over each batch in the test generator
337 for i in range(len(test_generator)):

```

```

338     images, labels = test_generator.next()
339     predictions = model.predict(images)
340     pred_classes = (predictions > 0.5).astype(int)
341
342     # Process each image in the batch
343     for j in range(len(images)):
344         # Correctly handle images based on channel ←
345         # information
346         if images[j].shape[-1] == 3: # Color images
347             img = Image.fromarray((images[j] * 255).astype('←
348             uint8')).convert('L')
349         else: # Grayscale images, potentially with a channel←
350             dimension of 1
351             # Ensure the image is 2D
352             img_array = (images[j] * 255).squeeze() # Remove←
353             # singleton dimensions
354             img = Image.fromarray(img_array.astype('uint8'), ←
355             'L')
356
357             # Determine predicted class name
358             predicted_label = pred_classes[j]
359             predicted_class_name = "noairgap" if predicted_label ←
360             == 1 else "airgap"
361
362             # Prepare text to draw on the image
363             annotation = f'Predicted class: {predicted_class_name}←
364             '
365
366             # Draw text on the image using PIL (default font)
367             draw = ImageDraw.Draw(img)
368             draw.text((10, 10), annotation, fill='white')
369
370             # Add title to the image
371             plt.imshow(img, cmap='gray') # Ensure the image is ←
372             # displayed as grayscale
373             plt.title(f'Predicted class: {predicted_class_name}')←
374             # Display the name of the predicted class
375             plt.axis('off')
376
377             # Get the original filename
378             original_filename = test_generator.filenames[←
379                 test_generator.batch_index * test_generator.←
380                 batch_size + j]
381             original_filename = os.path.basename(←
382                 original_filename) # Get just the file name
383
384             # Generate new filename based on predicted class
385             predicted_filename = original_filename.split('.')[0] ←
386             + '_' + predicted_class_name + '.' + ←

```

```

            original_filename.split('.')[ -1]
374
375     # Save the image with the new filename
376     plt.savefig(os.path.join(save_dir, predicted_filename))
377         , bbox_inches='tight' , pad_inches=0)
378     plt.close()
379 print("Images have been annotated and saved.")

```

Program CNN 2-Dimensi Percobaan Kedua

Program 5.9: Program CNN 2-Dimensi Percobaan Kedua

```

1 # -*- coding: utf-8 -*-
2 """"CNN TA v1 d2.ipynb
3
4 Automatically generated by Colab.
5
6 Original file is located at
7     https://colab.research.google.com/drive/1↔
8         Rbs0ElldUPIz1tzB4VAn-gmnBZ1Wl7nD
9 """
10 import tensorflow as tf
11 tf.__version__
12
13 from tensorflow.keras.models import Sequential
14 from tensorflow.keras.layers import Conv2D, MaxPooling2D, ←
15     Flatten, Dense, Dropout
16 from tensorflow.keras.preprocessing.image import ←
17     ImageDataGenerator
18 from tensorflow.keras.metrics import Precision, Recall
19 import os
20 import time
21 import time
22 from sklearn.metrics import confusion_matrix
23 import seaborn as sns
24 import matplotlib.pyplot as plt
25 import numpy as np
26
27 # Set the seed for TensorFlow's random number
28 np.random.seed(42)
29 tf.random.set_seed(42)
30
31 from google.colab import drive
32 drive.mount('/content/drive')
33 datasetDir = "/content/drive/MyDrive/KeperluanTA/another1/out↔
"

```

33

```

34 train_datagen = ImageDataGenerator(
35     rescale=1.0/255,
36     shear_range=0.2,
37     zoom_range=0.2,
38     fill_mode="nearest",
39     validation_split=0.1765
40 )
41
42 val_datagen = ImageDataGenerator(
43     rescale=1.0/255,
44     validation_split=0.1765
45 )
46
47 train_generator = train_datagen.flow_from_directory(
48     datasetDir,
49     target_size=(330, 540),
50     batch_size=10,
51     class_mode="binary", # Set class_mode to "categorical" ←
52         for multi-class classification
53     color_mode="grayscale", # Generate grayscale images
54     subset="training"
55 )
56
57 validation_generator = val_datagen.flow_from_directory(
58     datasetDir,
59     target_size=(330, 540),
60     batch_size=10,
61     class_mode="binary", # Set class_mode to "categorical" ←
62         for multi-class classification
63     color_mode="grayscale", # Generate grayscale images
64     subset="validation"
65 )
66
67 print(train_generator.class_indices)
68
69 # Now your model definition follows
70 model = Sequential()
71
72 # First Convolutional Block
73 model.add(Conv2D(32, (3, 3), activation="relu", input_shape=
74     =(330, 540, 1)))
75 model.add(MaxPooling2D((2, 2)))
76
77 # Second Convolutional Block
78 model.add(Conv2D(64, (3, 3), activation="relu"))
79 model.add(MaxPooling2D((2, 2)))

```

```

80 # Third Convolutional Block
81 model.add(Conv2D(128, (3, 3), activation="relu"))
82 model.add(MaxPooling2D((2, 2)))
83
84 # Fourth Convolutional Block
85 model.add(Conv2D(256, (3, 3), activation="relu"))
86 model.add(MaxPooling2D((2, 2)))
87
88 # Flattening and Fully Connected Layers
89 model.add(Flatten())
90 model.add(Dense(256, activation="relu"))
91 model.add(Dropout(0.5)) # Dropout layer to reduce ←
    overfitting
92 model.add(Dense(128, activation="relu"))
93 model.add(Dropout(0.5)) # Another dropout layer
94
95 # Output Layer for binary classification
96 model.add(Dense(1, activation="sigmoid"))
97
98 model.summary()
99
100 model.compile(
101     optimizer="adam",
102     loss="binary_crossentropy",
103     metrics=["accuracy"])
104 )
105
106 from tensorflow.keras.callbacks import EarlyStopping
107
108 # Define EarlyStopping callback
109 early_stopping = EarlyStopping(
110     monitor='val_loss',      # Metric to monitor
111     patience=10,            # Number of epochs with no ←
        improvement after which training will be stopped
112     verbose=1,              # To log when training is being ←
        stopped
113     mode='min',             # Stops training when the ←
        quantity monitored has stopped decreasing
114     restore_best_weights=True # Restores model weights from ←
        the epoch with the best value of the monitored ←
        quantity
115 )
116
117 # Fit the model with the EarlyStopping callback
118 history = model.fit(
119     train_generator,
120     validation_data=validation_generator,
121     epochs=100,
122     validation_steps=60,

```

```

123     verbose=1,
124     callbacks=[early_stopping] # Include the callback in the←
125 )
126
127 model.evaluate(validation_generator)
128
129 model.evaluate(train_generator)
130
131 # Specify the directory path
132 save_dir = '/content/drive/MyDrive/KeperluanTA/another1/2rev'
133
134 # Create the directory if it doesn't exist
135 os.makedirs(save_dir, exist_ok=True)
136
137 # Save the model to the specified directory
138 model.save(save_dir + 'model.h5')
139
140 model.save("model.h5")
141
142 import matplotlib.pyplot as plt
143
144 # Plot training & validation accuracy values
145 plt.figure(figsize=(12, 4))
146
147 plt.subplot(1, 2, 1)
148 plt.plot(history.history['accuracy'])
149 plt.plot(history.history['val_accuracy'])
150 plt.title('Model Accuracy')
151 plt.ylabel('Accuracy')
152 plt.xlabel('Epoch')
153 plt.legend(['Train', 'Validation'], loc='upper left')
154
155 # Plot training & validation loss values
156 plt.subplot(1, 2, 2)
157 plt.plot(history.history['loss'])
158 plt.plot(history.history['val_loss'])
159 plt.title('Model Loss')
160 plt.ylabel('Loss')
161 plt.xlabel('Epoch')
162 plt.legend(['Train', 'Validation'], loc='upper left')
163
164 plt.show()
165
166 import time
167
168 from sklearn.metrics import confusion_matrix
169 import seaborn as sns
170 import matplotlib.pyplot as plt

```

```

171 import numpy as np
172
173 # Create new generators without shuffling for evaluating ←
174     confusion matrix
174 eval_train_generator = train_datagen.flow_from_directory(
175     datasetDir,
176     target_size=(330, 540),
177     batch_size=10, # Adjust this based on your setup
178     class_mode="binary", # Adjusted for binary ←
179         classification
179     color_mode="grayscale",
180     shuffle=False, # Important for matching predictions to ←
181         labels
181     subset="training"
182 )
183
184 eval_validation_generator = val_datagen.flow_from_directory(
185     datasetDir,
186     target_size=(330, 540),
187     batch_size=10, # Match the batch size used for training/←
187         validation
188     class_mode="binary", # Adjusted for binary ←
188         classification
189     color_mode="grayscale",
190     shuffle=False, # Important for matching predictions to ←
190         labels
191     subset="validation"
192 )
193 # Start timer for inference
194 start_time = time.time()
195
196 # Predict the data
197 train_predictions = model.predict(eval_train_generator, ←
197     verbose=1)
198 validation_predictions = model.predict(←
198     eval_validation_generator, verbose=1)
199
200 # End timer and calculate inference time
201 inference_time = time.time() - start_time
202 print("Inference time for the test set: {:.2f} seconds".←
202     format(inference_time))
203
204 # Convert probabilities to binary predictions based on a 0.5 ←
204     threshold
205 train_pred_classes = (train_predictions > 0.5).astype(int).←
205     reshape(-1)
206 validation_pred_classes = (validation_predictions > 0.5).←
206     astype(int).reshape(-1)
207

```

```

208 # True labels (already in binary format)
209 train_true_classes = eval_train_generator.classes
210 validation_true_classes = eval_validation_generator.classes
211
212 # Compute confusion matrices
213 train_cm = confusion_matrix(train_true_classes, ←
    train_pred_classes)
214 validation_cm = confusion_matrix(validation_true_classes, ←
    validation_pred_classes)
215
216 # Plotting the confusion matrices
217 fig, ax = plt.subplots(1, 2, figsize=(12, 5))
218
219 sns.heatmap(train_cm, annot=True, fmt='d', cmap='Blues', ax=←
    ax[0])
220 ax[0].set_title('Training Confusion Matrix')
221 ax[0].set_xlabel('Predicted Labels')
222 ax[0].set_ylabel('True Labels')
223 ax[0].set_xticklabels(['Negative', 'Positive'])
224 ax[0].set_yticklabels(['Negative', 'Positive'])
225
226 sns.heatmap(validation_cm, annot=True, fmt='d', cmap='Greens'←
    , ax=ax[1])
227 ax[1].set_title('Validation Confusion Matrix')
228 ax[1].set_xlabel('Predicted Labels')
229 ax[1].set_ylabel('True Labels')
230 ax[1].set_xticklabels(['Negative', 'Positive'])
231 ax[1].set_yticklabels(['Negative', 'Positive'], va='center')
232
233 plt.tight_layout()
234 plt.show()
235
236 import time
237 from sklearn.metrics import confusion_matrix
238 import seaborn as sns
239 import matplotlib.pyplot as plt
240 import numpy as np
241
242 testDir = "/content/drive/MyDrive/KeperluanTA/another1/←
    testout"
243
244 # Set up the test data generator
245 test_datagen = ImageDataGenerator(rescale=1.0/255)
246
247 test_generator = test_datagen.flow_from_directory(
248     testDir, # Directory with test images
249     target_size=(330, 540),
250     batch_size=10, # Adjust based on your setup

```

```

251     class_mode="binary", # Ensure this matches your label ←
252         setup
253     color_mode="grayscale",
254     shuffle=False # Important for matching predictions to ←
255         labels
256 )
257
258 # Start timer for inference
259 start_time = time.time()
260
261 # Predict the data
262 test_predictions = model.predict(test_generator, verbose=1)
263
264 # End timer and calculate inference time
265 inference_time = time.time() - start_time
266 print("Inference time for the test set: {:.2f} seconds".←
267       format(inference_time))
268
269 # Convert probabilities to binary predictions based on a 0.5 ←
270         threshold
271 test_pred_classes = (test_predictions > 0.5).astype(int).←
272         reshape(-1)
273
274 # True labels (already in binary format)
275 test_true_classes = test_generator.classes
276
277 # Compute the confusion matrix
278 test_cm = confusion_matrix(test_true_classes, ←
279     test_pred_classes)
280
281 # Plotting the confusion matrix
282 plt.figure(figsize=(6, 5))
283 sns.heatmap(test_cm, annot=True, fmt='d', cmap='Purples')
284 plt.title('Test Confusion Matrix')
285 plt.xlabel('Predicted Labels')
286 plt.ylabel('True Labels')
287 plt.xticks([0.5, 1.5], ['Negative', 'Positive'])
288 plt.yticks([0.5, 1.5], ['Negative', 'Positive'], va='center')
289 plt.show()
290
291 import tensorflow as tf
292 from tensorflow.keras.metrics import Precision, Recall
293
294 # Define a custom F1 score metric
295 class F1Score(tf.keras.metrics.Metric):
296     def __init__(self, name='f1_score', **kwargs):
297         super(F1Score, self).__init__(name=name, **kwargs)
298         self.precision = Precision()
299         self.recall = Recall()

```

```

294
295     def update_state(self, y_true, y_pred, sample_weight=None):
296         self.precision.update_state(y_true, y_pred,
297                                     sample_weight)
297         self.recall.update_state(y_true, y_pred,
298                                     sample_weight)
298
299     def result(self):
300         p = self.precision.result()
301         r = self.recall.result()
302         return 2 * ((p * r) / (p + r + tf.keras.backend.ε))
303
304     def reset_state(self):
305         self.precision.reset_state()
306         self.recall.reset_state()
307
308 # Instantiate the F1Score metric for training, validation, and test
309 f1_score_train = F1Score()
310 f1_score_val = F1Score()
311 f1_score_test = F1Score()
312
313 # Update the state of the F1Score metric with training data
314 f1_score_train.update_state(train_true_classes,
315                             train_pred_classes)
315 # Update the state of the F1Score metric with validation data
316 f1_score_val.update_state(validation_true_classes,
317                             validation_pred_classes)
317 # Update the state of the F1Score metric with test data
318 f1_score_test.update_state(test_true_classes,
319                             test_pred_classes)
320
320 # Calculate and print the F1 scores
321 print("Calculated F1 Score for Training Set:", f1_score_train.
322       result().numpy())
322 print("Calculated F1 Score for Validation Set:", f1_score_val.
323       result().numpy())
323 print("Calculated F1 Score for Test Set:", f1_score_test.
324       result().numpy())
324
325 import os
326 import matplotlib.pyplot as plt
327 from tensorflow.keras.preprocessing.image import ImageDataGenerator
328 from PIL import Image, ImageDraw
329

```

```

330 # Assuming `model` and `test_generator` have already been ←
      defined as in your original code.
331
332 # Create a directory to save the annotated images
333 save_dir = '/content/drive/MyDrive/Deteksi/Eks2'
334 os.makedirs(save_dir, exist_ok=True)
335
336 # Iterate over each batch in the test generator
337 for i in range(len(test_generator)):
338     images, labels = test_generator.next()
339     predictions = model.predict(images)
340     pred_classes = (predictions > 0.5).astype(int)
341
342     # Process each image in the batch
343     for j in range(len(images)):
344         # Correctly handle images based on channel ←
            information
345         if images[j].shape[-1] == 3: # Color images
346             img = Image.fromarray((images[j] * 255).astype('←
                uint8')).convert('L')
347         else: # Grayscale images, potentially with a channel←
            dimension of 1
348             # Ensure the image is 2D
349             img_array = (images[j] * 255).squeeze() # Remove←
                singleton dimensions
350             img = Image.fromarray(img_array.astype('uint8'), ←
                'L')
351
352         # Determine predicted class name
353         predicted_label = pred_classes[j]
354         predicted_class_name = "noairgap" if predicted_label ←
            == 1 else "airgap"
355
356         # Prepare text to draw on the image
357         annotation = f'Predicted class: {predicted_class_name}←
            }'
358
359         # Draw text on the image using PIL (default font)
360         draw = ImageDraw.Draw(img)
361         draw.text((10, 10), annotation, fill='white')
362
363         # Add title to the image
364         plt.imshow(img, cmap='gray') # Ensure the image is ←
            displayed as grayscale
365         plt.title(f'Predicted class: {predicted_class_name}')←
            # Display the name of the predicted class
366         plt.axis('off')
367
368         # Get the original filename

```

```

369     original_filename = test_generator.filenames[←
370         test_generator.batch_index * test_generator.←
371         batch_size + j]
370     original_filename = os.path.basename(←
371         original_filename) # Get just the file name
372
372     # Generate new filename based on predicted class
373     predicted_filename = original_filename.split('.')[0] ←
373         + '_' + predicted_class_name + '.' + ←
373         original_filename.split('.')[1]
374
375     # Save the image with the new filename
376     plt.savefig(os.path.join(save_dir, predicted_filename←
376         ), bbox_inches='tight', pad_inches=0)
377     plt.close()
378
379 print("Images have been annotated and saved.")

```

Program CNN 2-Dimensi Percobaan Ketiga

Program 5.10: Program CNN 2-Dimensi Percobaan Ketiga

```

1 # -*- coding: utf-8 -*-
2 """CNN TA v1 d3.ipynb
3
4 Automatically generated by Colab.
5
6 Original file is located at
7     https://colab.research.google.com/drive/1←
7         REU_uabiXleR9J2P306nMpuyenmJobka
8 """
9
10 import tensorflow as tf
11 tf.__version__
12
13 from tensorflow.keras.models import Sequential
14 from tensorflow.keras.layers import Conv2D, MaxPooling2D, ←
14     Flatten, Dense, Dropout
15 from tensorflow.keras.preprocessing.image import ←
15     ImageDataGenerator
16 from tensorflow.keras.metrics import Precision, Recall
17 import os
18 import time
19 import time
20 from sklearn.metrics import confusion_matrix
21 import seaborn as sns
22 import matplotlib.pyplot as plt
23 import numpy as np
24
25 # Set the seed for TensorFlow's random number

```

```

26 np.random.seed(42)
27 tf.random.set_seed(42)
28
29 from google.colab import drive
30 drive.mount('/content/drive')
31
32 datasetDir = "/content/drive/MyDrive/KeperluanTA/out"
33
34 train_datagen = ImageDataGenerator(
35     rescale=1.0/255,
36     shear_range=0.2,
37     zoom_range=0.2,
38     fill_mode="nearest",
39     validation_split=0.25
40 )
41
42 val_datagen = ImageDataGenerator(
43     rescale=1.0/255,
44     validation_split=0.25
45 )
46
47 train_generator = train_datagen.flow_from_directory(
48     datasetDir,
49     target_size=(330, 540),
50     batch_size=10,
51     class_mode="binary", # Set class_mode to "categorical" ←
52     # for multi-class classification
53     color_mode="grayscale", # Generate grayscale images
54     subset="training"
55 )
56
57 validation_generator = val_datagen.flow_from_directory(
58     datasetDir,
59     target_size=(330, 540),
60     batch_size=10,
61     class_mode="binary", # Set class_mode to "categorical" ←
62     # for multi-class classification
63     color_mode="grayscale", # Generate grayscale images
64     subset="validation"
65 )
66
67 print(train_generator.class_indices)
68
69 # Now your model definition follows
70 model = Sequential()
71
72 # First Convolutional Block

```

```

73 model.add(Conv2D(32, (3, 3), activation="relu", input_shape=→
    =(330, 540, 1)))
74 model.add(MaxPooling2D((2, 2)))
75
76 # Second Convolutional Block
77 model.add(Conv2D(64, (3, 3), activation="relu"))
78 model.add(MaxPooling2D((2, 2)))
79
80 # Third Convolutional Block
81 model.add(Conv2D(128, (3, 3), activation="relu"))
82 model.add(MaxPooling2D((2, 2)))
83
84 # Fourth Convolutional Block
85 model.add(Conv2D(256, (3, 3), activation="relu"))
86 model.add(MaxPooling2D((2, 2)))
87
88 # Flattening and Fully Connected Layers
89 model.add(Flatten())
90 model.add(Dense(256, activation="relu"))
91 model.add(Dropout(0.5)) # Dropout layer to reduce ←
    overfitting
92 model.add(Dense(128, activation="relu"))
93 model.add(Dropout(0.5)) # Another dropout layer
94
95 # Output Layer for binary classification
96 model.add(Dense(1, activation="sigmoid"))
97
98 model.summary()
99
100 model.compile(
101     optimizer="adam",
102     loss="binary_crossentropy",
103     metrics=["accuracy"])
104 )
105
106 from tensorflow.keras.callbacks import EarlyStopping
107
108 # Define EarlyStopping callback
109 early_stopping = EarlyStopping(
110     monitor='val_loss',      # Metric to monitor
111     patience=10,            # Number of epochs with no ←
        improvement after which training will be stopped
112     verbose=1,              # To log when training is being ←
        stopped
113     mode='min',             # Stops training when the ←
        quantity monitored has stopped decreasing
114     restore_best_weights=True # Restores model weights from ←
        the epoch with the best value of the monitored ←
        quantity

```

```

115 )
116
117 # Fit the model with the EarlyStopping callback
118 history = model.fit(
119     train_generator,
120     validation_data=validation_generator,
121     epochs=100,
122     validation_steps=80,
123     verbose=1,
124     callbacks=[early_stopping] # Include the callback in the←
125         list
126
127 model.evaluate(validation_generator)
128
129 model.evaluate(train_generator)
130
131 # Specify the directory path
132 save_dir = '/content/drive/MyDrive/KeperluanTA/3rev'
133
134 # Create the directory if it doesn't exist
135 os.makedirs(save_dir, exist_ok=True)
136
137 # Save the model to the specified directory
138 model.save(save_dir + 'model.h5')
139
140 model.save("model.h5")
141
142 import matplotlib.pyplot as plt
143
144 # Plot training & validation accuracy values
145 plt.figure(figsize=(12, 4))
146
147 plt.subplot(1, 2, 1)
148 plt.plot(history.history['accuracy'])
149 plt.plot(history.history['val_accuracy'])
150 plt.title('Model Accuracy')
151 plt.ylabel('Accuracy')
152 plt.xlabel('Epoch')
153 plt.legend(['Train', 'Validation'], loc='upper left')
154
155 # Plot training & validation loss values
156 plt.subplot(1, 2, 2)
157 plt.plot(history.history['loss'])
158 plt.plot(history.history['val_loss'])
159 plt.title('Model Loss')
160 plt.ylabel('Loss')
161 plt.xlabel('Epoch')
162 plt.legend(['Train', 'Validation'], loc='upper left')

```

```

163
164 plt.show()
165
166 import time
167
168 from sklearn.metrics import confusion_matrix
169 import seaborn as sns
170 import matplotlib.pyplot as plt
171 import numpy as np
172
173 # Create new generators without shuffling for evaluating ←
174     confusion matrix
175 eval_train_generator = train_datagen.flow_from_directory(
176     datasetDir,
177     target_size=(330, 540),
178     batch_size=10, # Adjust this based on your setup
179     class_mode="binary", # Adjusted for binary ←
180         classification
181     color_mode="grayscale",
182     shuffle=False, # Important for matching predictions to ←
183         labels
184     subset="training"
185 )
186
187 eval_validation_generator = val_datagen.flow_from_directory(
188     datasetDir,
189     target_size=(330, 540),
190     batch_size=10, # Match the batch size used for training/←
191         validation
192     class_mode="binary", # Adjusted for binary ←
193         classification
194     color_mode="grayscale",
195     shuffle=False, # Important for matching predictions to ←
196         labels
197     subset="validation"
198 )
199
200 # Start timer for inference
201 start_time = time.time()
202
203 # Predict the data
204 train_predictions = model.predict(eval_train_generator, ←
205     verbose=1)
206 validation_predictions = model.predict(←
207     eval_validation_generator, verbose=1)
208
209 # End timer and calculate inference time
210 inference_time = time.time() - start_time
211 print("Inference time for the test set: {:.2f} seconds".←
212     format(inference_time))

```

```

203
204 # Convert probabilities to binary predictions based on a 0.5 ←
     threshold
205 train_pred_classes = (train_predictions > 0.5).astype(int).←
     reshape(-1)
206 validation_pred_classes = (validation_predictions > 0.5).←
     astype(int).reshape(-1)
207
208 # True labels (already in binary format)
209 train_true_classes = eval_train_generator.classes
210 validation_true_classes = eval_validation_generator.classes
211
212 # Compute confusion matrices
213 train_cm = confusion_matrix(train_true_classes, ←
     train_pred_classes)
214 validation_cm = confusion_matrix(validation_true_classes, ←
     validation_pred_classes)
215
216 # Plotting the confusion matrices
217 fig, ax = plt.subplots(1, 2, figsize=(12, 5))
218
219 sns.heatmap(train_cm, annot=True, fmt='d', cmap='Blues', ax=←
     ax[0])
220 ax[0].set_title('Training Confusion Matrix')
221 ax[0].set_xlabel('Predicted Labels')
222 ax[0].set_ylabel('True Labels')
223 ax[0].set_xticklabels(['Negative', 'Positive'])
224 ax[0].set_yticklabels(['Negative', 'Positive'])
225
226 sns.heatmap(validation_cm, annot=True, fmt='d', cmap='Greens'←
     , ax=ax[1])
227 ax[1].set_title('Validation Confusion Matrix')
228 ax[1].set_xlabel('Predicted Labels')
229 ax[1].set_ylabel('True Labels')
230 ax[1].set_xticklabels(['Negative', 'Positive'])
231 ax[1].set_yticklabels(['Negative', 'Positive'], va='center')
232
233 plt.tight_layout()
234 plt.show()
235
236 import time
237 from sklearn.metrics import confusion_matrix
238 import seaborn as sns
239 import matplotlib.pyplot as plt
240 import numpy as np
241
242 testDir = "/content/drive/MyDrive/KeperluanTA/testout"
243
244 # Set up the test data generator

```

```

245 test_datagen = ImageDataGenerator(rescale=1.0/255)
246
247 test_generator = test_datagen.flow_from_directory(
248     testDir, # Directory with test images
249     target_size=(330, 540),
250     batch_size=10, # Adjust based on your setup
251     class_mode="binary", # Ensure this matches your label ↪
252         setup
253     color_mode="grayscale",
254     shuffle=False # Important for matching predictions to ↪
255         labels
256 )
257
258 # Start timer for inference
259 start_time = time.time()
260
261 # Predict the data
262 test_predictions = model.predict(test_generator, verbose=1)
263
264 # End timer and calculate inference time
265 inference_time = time.time() - start_time
266 print("Inference time for the test set: {:.2f} seconds".format(inference_time))
267
268 # Convert probabilities to binary predictions based on a 0.5 ↪
269 # threshold
270 test_pred_classes = (test_predictions > 0.5).astype(int).reshape(-1)
271
272 # True labels (already in binary format)
273 test_true_classes = test_generator.classes
274
275 # Compute the confusion matrix
276 test_cm = confusion_matrix(test_true_classes, ↪
277     test_pred_classes)
278
279 # Plotting the confusion matrix
280 plt.figure(figsize=(6, 5))
281 sns.heatmap(test_cm, annot=True, fmt='d', cmap='Purples')
282 plt.title('Test Confusion Matrix')
283 plt.xlabel('Predicted Labels')
284 plt.ylabel('True Labels')
285 plt.xticks([0.5, 1.5], ['Negative', 'Positive'])
286 plt.yticks([0.5, 1.5], ['Negative', 'Positive'], va='center')
287 plt.show()

import tensorflow as tf
from tensorflow.keras.metrics import Precision, Recall

```

```

288 # Define a custom F1 score metric
289 class F1Score(tf.keras.metrics.Metric):
290     def __init__(self, name='f1_score', **kwargs):
291         super(F1Score, self).__init__(name=name, **kwargs)
292         self.precision = Precision()
293         self.recall = Recall()
294
295     def update_state(self, y_true, y_pred, sample_weight=None):
296         self.precision.update_state(y_true, y_pred,
297                                     sample_weight)
297         self.recall.update_state(y_true, y_pred,
298                                     sample_weight)
298
299     def result(self):
300         p = self.precision.result()
301         r = self.recall.result()
302         return 2 * ((p * r) / (p + r + tf.keras.backend.epsilon()))
303
304     def reset_state(self):
305         self.precision.reset_state()
306         self.recall.reset_state()
307
308 # Instantiate the F1Score metric for training, validation, and test
309 f1_score_train = F1Score()
310 f1_score_val = F1Score()
311 f1_score_test = F1Score()
312
313 # Update the state of the F1Score metric with training data
314 f1_score_train.update_state(train_true_classes,
315                             train_pred_classes)
315 # Update the state of the F1Score metric with validation data
316 f1_score_val.update_state(validation_true_classes,
317                             validation_pred_classes)
317 # Update the state of the F1Score metric with test data
318 f1_score_test.update_state(test_true_classes,
319                           test_pred_classes)
320
321 # Calculate and print the F1 scores
322 print("Calculated F1 Score for Training Set:", f1_score_train.result().numpy())
323 print("Calculated F1 Score for Validation Set:", f1_score_val.result().numpy())
324 print("Calculated F1 Score for Test Set:", f1_score_test.result().numpy())
325 import os

```

```

326 import matplotlib.pyplot as plt
327 from tensorflow.keras.preprocessing.image import ←
    ImageDataGenerator
328 from PIL import Image, ImageDraw
329
330 # Assuming `model` and `test_generator` have already been ←
    defined as in your original code.
331
332 # Create a directory to save the annotated images
333 save_dir = '/content/drive/MyDrive/Deteksi/Eks3'
334 os.makedirs(save_dir, exist_ok=True)
335
336 # Iterate over each batch in the test generator
337 for i in range(len(test_generator)):
338     images, labels = test_generator.next()
339     predictions = model.predict(images)
340     pred_classes = (predictions > 0.5).astype(int)
341
342     # Process each image in the batch
343     for j in range(len(images)):
344         # Correctly handle images based on channel ←
            information
345         if images[j].shape[-1] == 3: # Color images
346             img = Image.fromarray((images[j] * 255).astype('←
                uint8')).convert('L')
347         else: # Grayscale images, potentially with a channel←
            dimension of 1
348             # Ensure the image is 2D
349             img_array = (images[j] * 255).squeeze() # Remove←
                singleton dimensions
350             img = Image.fromarray(img_array.astype('uint8'), ←
                'L')
351
352         # Determine predicted class name
353         predicted_label = pred_classes[j]
354         predicted_class_name = "noairgap" if predicted_label ←
            == 1 else "airgap"
355
356         # Prepare text to draw on the image
357         annotation = f'Predicted class: {predicted_class_name←
            }'
358
359         # Draw text on the image using PIL (default font)
360         draw = ImageDraw.Draw(img)
361         draw.text((10, 10), annotation, fill='white')
362
363         # Add title to the image
364         plt.imshow(img, cmap='gray') # Ensure the image is ←
            displayed as grayscale

```

```

365     plt.title(f'Predicted class: {predicted_class_name}') ←
366         # Display the name of the predicted class
367     plt.axis('off')
368
369     # Get the original filename
370     original_filename = test_generator.filenames[←
371         test_generator.batch_index * test_generator.←
372         batch_size + j]
373     original_filename = os.path.basename(←
374         original_filename) # Get just the file name
375
376     # Generate new filename based on predicted class
377     predicted_filename = original_filename.split('.')[0] ←
378         + '_' + predicted_class_name + '.' + ←
379         original_filename.split('.')[1]
380
381     # Save the image with the new filename
382     plt.savefig(os.path.join(save_dir, predicted_filename) ←
383         ), bbox_inches='tight', pad_inches=0)
384     plt.close()
385
386 print("Images have been annotated and saved.")

```

Program CNN 2-Dimensi Percobaan Keempat

Program 5.11: Program CNN 2-Dimensi Percobaan Keempat

```

1 # -*- coding: utf-8 -*-
2 """"CNN TA v2 d2.ipynb
3
4 Automatically generated by Colab.
5
6 Original file is located at
7     https://colab.research.google.com/drive/14←
7         j8vm7ZgSmFuK882nIG5Jnml11XZ73hg
8 """
9
10 import tensorflow as tf
11 tf.__version__
12
13 from tensorflow.keras.models import Sequential
14 from tensorflow.keras.layers import Conv2D, MaxPooling2D, ←
14     Flatten, Dense, Dropout, BatchNormalization
15 from tensorflow.keras.preprocessing.image import ←
15     ImageDataGenerator
16 from tensorflow.keras.metrics import Precision, Recall
17 import os
18 import time
19 import time
20 from sklearn.metrics import confusion_matrix

```

```

21 import seaborn as sns
22 import matplotlib.pyplot as plt
23 import numpy as np
24
25 # Set the seed for TensorFlow's random number generator
26 np.random.seed(42)
27 tf.random.set_seed(42)
28
29 from google.colab import drive
30 drive.mount('/content/drive')
31
32 datasetDir = "/content/drive/MyDrive/KeperluanTA/another1/out"
33
34 train_datagen = ImageDataGenerator(
35     rescale=1.0/255,
36     shear_range=0.2,
37     zoom_range=0.2,
38     fill_mode="nearest",
39     validation_split=0.1765
40 )
41
42 val_datagen = ImageDataGenerator(
43     rescale=1.0/255,
44     validation_split=0.1765
45 )
46
47 train_generator = train_datagen.flow_from_directory(
48     datasetDir,
49     target_size=(330, 540),
50     batch_size=10,
51     class_mode="binary", # Set class_mode to "categorical" ←
52     # for multi-class classification
53     color_mode="grayscale", # Generate grayscale images
54     subset="training"
55 )
56
57 validation_generator = val_datagen.flow_from_directory(
58     datasetDir,
59     target_size=(330, 540),
60     batch_size=10,
61     class_mode="binary", # Set class_mode to "categorical" ←
62     # for multi-class classification
63     color_mode="grayscale", # Generate grayscale images
64     subset="validation"
65 )
66
67 print(train_generator.class_indices)

```

```

67 train_generator.image_shape
68
69 # Now your model definition follows
70 model = Sequential()
71
72 # First Convolutional Block
73 model.add(Conv2D(32, (3, 3), activation="relu", input_shape=(330, 540, 1)))
74 model.add(BatchNormalization())
75 model.add(MaxPooling2D((2, 2)))
76
77 # Second Convolutional Block
78 model.add(Conv2D(64, (3, 3), activation="relu"))
79 model.add(BatchNormalization())
80 model.add(MaxPooling2D((2, 2)))
81
82 # Third Convolutional Block
83 model.add(Conv2D(128, (3, 3), activation="relu"))
84 model.add(BatchNormalization())
85 model.add(MaxPooling2D((2, 2)))
86
87 # Fourth Convolutional Block
88 model.add(Conv2D(256, (3, 3), activation="relu"))
89 model.add(BatchNormalization())
90 model.add(MaxPooling2D((2, 2)))
91
92 # Flattening and Fully Connected Layers
93 model.add(Flatten())
94 model.add(Dense(256, activation="relu"))
95 model.add(Dropout(0.5)) # Dropout layer to reduce overfitting
96 model.add(Dense(128, activation="relu"))
97 model.add(Dropout(0.5)) # Another dropout layer
98
99 # Output Layer for binary classification
100 model.add(Dense(1, activation="sigmoid"))
101
102 model.summary()
103
104 model.compile(
105     optimizer="adam",
106     loss="binary_crossentropy",
107     metrics=["accuracy"])
108 )
109
110 from tensorflow.keras.callbacks import EarlyStopping
111
112 # Define EarlyStopping callback
113 early_stopping = EarlyStopping(

```

```

114     monitor='val_loss',      # Metric to monitor
115     patience=10,           # Number of epochs with no ←
116     improvement after which training will be stopped
117     verbose=1,             # To log when training is being ←
118     stopped
119     mode='min',            # Stops training when the ←
120     quantity monitored has stopped decreasing
121     restore_best_weights=True # Restores model weights from ←
122     the epoch with the best value of the monitored ←
123     quantity
124 )
125
126 # Fit the model with the EarlyStopping callback
127 history = model.fit(
128     train_generator,
129     validation_data=validation_generator,
130     epochs=100,
131     validation_steps=60,
132     verbose=1,
133     callbacks=[early_stopping] # Include the callback in the←
134     list
135 )
136
137 model.evaluate(validation_generator)
138
139 model.evaluate(train_generator)
140
141 # Specify the directory path
142 save_dir = '/content/drive/MyDrive/KeperluanTA/4rev'
143
144 # Create the directory if it doesn't exist
145 os.makedirs(save_dir, exist_ok=True)
146
147 # Save the model to the specified directory
148 model.save(save_dir + 'model.h5')
149
150 model.save("model.h5")
151
152 import matplotlib.pyplot as plt
153
154 # Plot training & validation accuracy values
155 plt.figure(figsize=(12, 4))
156
157 plt.subplot(1, 2, 1)
158 plt.plot(history.history['accuracy'])
159 plt.plot(history.history['val_accuracy'])
160 plt.title('Model Accuracy')
161 plt.ylabel('Accuracy')
162 plt.xlabel('Epoch')

```

```

157 plt.legend(['Train', 'Validation'], loc='upper left')
158
159 # Plot training & validation loss values
160 plt.subplot(1, 2, 2)
161 plt.plot(history.history['loss'])
162 plt.plot(history.history['val_loss'])
163 plt.title('Model Loss')
164 plt.ylabel('Loss')
165 plt.xlabel('Epoch')
166 plt.legend(['Train', 'Validation'], loc='upper left')
167
168 plt.show()
169
170 import time
171
172 from sklearn.metrics import confusion_matrix
173 import seaborn as sns
174 import matplotlib.pyplot as plt
175 import numpy as np
176
177 # Create new generators without shuffling for evaluating ←
178 # confusion matrix
179 eval_train_generator = train_datagen.flow_from_directory(
180     datasetDir,
181     target_size=(330, 540),
182     batch_size=10, # Adjust this based on your setup
183     class_mode="binary", # Adjusted for binary ←
184     classification
185     color_mode="grayscale",
186     shuffle=False, # Important for matching predictions to ←
187     labels
188     subset="training"
189 )
190
191 eval_validation_generator = val_datagen.flow_from_directory(
192     datasetDir,
193     target_size=(330, 540),
194     batch_size=10, # Match the batch size used for training/←
195     validation
196     class_mode="binary", # Adjusted for binary ←
197     classification
198     color_mode="grayscale",
199     shuffle=False, # Important for matching predictions to ←
200     labels
201     subset="validation"
202 )
203
204 # Start timer for inference
205 start_time = time.time()
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999

```

```

200 # Predict the data
201 train_predictions = model.predict(eval_train_generator, ←
    verbose=1)
202 validation_predictions = model.predict(←
    eval_validation_generator, verbose=1)
203
204 # End timer and calculate inference time
205 inference_time = time.time() - start_time
206 print("Inference time for the test set: {:.2f} seconds".←
    format(inference_time))
207
208 # Convert probabilities to binary predictions based on a 0.5 ←
    threshold
209 train_pred_classes = (train_predictions > 0.5).astype(int).←
    reshape(-1)
210 validation_pred_classes = (validation_predictions > 0.5).←
    astype(int).reshape(-1)
211
212 # True labels (already in binary format)
213 train_true_classes = eval_train_generator.classes
214 validation_true_classes = eval_validation_generator.classes
215
216 # Compute confusion matrices
217 train_cm = confusion_matrix(train_true_classes, ←
    train_pred_classes)
218 validation_cm = confusion_matrix(validation_true_classes, ←
    validation_pred_classes)
219
220 # Plotting the confusion matrices
221 fig, ax = plt.subplots(1, 2, figsize=(12, 5))
222
223 sns.heatmap(train_cm, annot=True, fmt='d', cmap='Blues', ax=←
    ax[0])
224 ax[0].set_title('Training Confusion Matrix')
225 ax[0].set_xlabel('Predicted Labels')
226 ax[0].set_ylabel('True Labels')
227 ax[0].set_xticklabels(['Negative', 'Positive'])
228 ax[0].set_yticklabels(['Negative', 'Positive'])
229
230 sns.heatmap(validation_cm, annot=True, fmt='d', cmap='Greens'←
    , ax=ax[1])
231 ax[1].set_title('Validation Confusion Matrix')
232 ax[1].set_xlabel('Predicted Labels')
233 ax[1].set_ylabel('True Labels')
234 ax[1].set_xticklabels(['Negative', 'Positive'])
235 ax[1].set_yticklabels(['Negative', 'Positive'], va='center')
236
237 plt.tight_layout()
238 plt.show()

```

```

239
240 import time
241 from sklearn.metrics import confusion_matrix
242 import seaborn as sns
243 import matplotlib.pyplot as plt
244 import numpy as np
245
246 testDir = "/content/drive/MyDrive/KeperluanTA/another1/←
247           testout"
248
249 # Set up the test data generator
250 test_datagen = ImageDataGenerator(rescale=1.0/255)
251
252 test_generator = test_datagen.flow_from_directory(
253     testDir, # Directory with test images
254     target_size=(330, 540),
255     batch_size=10, # Adjust based on your setup
256     class_mode="binary", # Ensure this matches your label ←
257     color_mode="grayscale",
258     shuffle=False # Important for matching predictions to ←
259     labels
260 )
261
262 # Start timer for inference
263 start_time = time.time()
264
265 # Predict the data
266 test_predictions = model.predict(test_generator, verbose=1)
267
268 # End timer and calculate inference time
269 inference_time = time.time() - start_time
270 print("Inference time for the test set: {:.2f} seconds".←
271       format(inference_time))
272
273 # Convert probabilities to binary predictions based on a 0.5 ←
274 # threshold
275 test_pred_classes = (test_predictions > 0.5).astype(int).←
276       reshape(-1)
277
278 # True labels (already in binary format)
279 test_true_classes = test_generator.classes
280
281 # Compute the confusion matrix
282 test_cm = confusion_matrix(test_true_classes, ←
283     test_pred_classes)
284
285 # Plotting the confusion matrix
286 plt.figure(figsize=(6, 5))

```

```

281 sns.heatmap(test_cm, annot=True, fmt='d', cmap='Purples')
282 plt.title('Test Confusion Matrix')
283 plt.xlabel('Predicted Labels')
284 plt.ylabel('True Labels')
285 plt.xticks([0.5, 1.5], ['Negative', 'Positive'])
286 plt.yticks([0.5, 1.5], ['Negative', 'Positive'], va='center')
287 plt.show()
288
289 import tensorflow as tf
290 from tensorflow.keras.metrics import Precision, Recall
291
292 # Define a custom F1 score metric
293 class F1Score(tf.keras.metrics.Metric):
294     def __init__(self, name='f1_score', **kwargs):
295         super(F1Score, self).__init__(name=name, **kwargs)
296         self.precision = Precision()
297         self.recall = Recall()
298
299     def update_state(self, y_true, y_pred, sample_weight=None):
300         self.precision.update_state(y_true, y_pred,
301                                     sample_weight)
302         self.recall.update_state(y_true, y_pred,
303                                 sample_weight)
304
305     def result(self):
306         p = self.precision.result()
307         r = self.recall.result()
308         return 2 * ((p * r) / (p + r + tf.keras.backend.epsilon()))
309
310     def reset_state(self):
311         self.precision.reset_state()
312         self.recall.reset_state()
313
314 # Instantiate the F1Score metric for training, validation, and test
315 f1_score_train = F1Score()
316 f1_score_val = F1Score()
317 f1_score_test = F1Score()
318
319 # Update the state of the F1Score metric with training data
320 f1_score_train.update_state(train_true_classes,
321                             train_pred_classes)
322 # Update the state of the F1Score metric with validation data
323 f1_score_val.update_state(validation_true_classes,
324                            validation_pred_classes)
325 # Update the state of the F1Score metric with test data

```

```

322 f1_score_test.update_state(test_true_classes, ←
    test_pred_classes)
323
324 # Calculate and print the F1 scores
325 print("Calculated F1 Score for Training Set:", f1_score_train←
    .result().numpy())
326 print("Calculated F1 Score for Validation Set:", f1_score_val←
    .result().numpy())
327 print("Calculated F1 Score for Test Set:", f1_score_test.←
    result().numpy())
328
329 import os
330 import matplotlib.pyplot as plt
331 from tensorflow.keras.preprocessing.image import ←
    ImageDataGenerator
332 from PIL import Image, ImageDraw
333
334 # Assuming `model` and `test_generator` have already been ←
    defined as in your original code.
335
336 # Create a directory to save the annotated images
337 save_dir = '/content/drive/MyDrive/Deteksi/Eks4'
338 os.makedirs(save_dir, exist_ok=True)
339
340 # Iterate over each batch in the test generator
341 for i in range(len(test_generator)):
342     images, labels = test_generator.next()
343     predictions = model.predict(images)
344     pred_classes = (predictions > 0.5).astype(int)
345
346     # Process each image in the batch
347     for j in range(len(images)):
348         # Correctly handle images based on channel ←
            information
349         if images[j].shape[-1] == 3: # Color images
350             img = Image.fromarray((images[j] * 255).astype('←
                uint8')).convert('L')
351         else: # Grayscale images, potentially with a channel←
            dimension of 1
352         # Ensure the image is 2D
353         img_array = (images[j] * 255).squeeze() # Remove←
            singleton dimensions
354         img = Image.fromarray(img_array.astype('uint8'), ←
            'L')
355
356         # Determine predicted class name
357         predicted_label = pred_classes[j]
358         predicted_class_name = "noairgap" if predicted_label ←
            == 1 else "airgap"

```

```

359
360     # Prepare text to draw on the image
361     annotation = f'Predicted class: {predicted_class_name} '
362
363     # Draw text on the image using PIL (default font)
364     draw = ImageDraw.Draw(img)
365     draw.text((10, 10), annotation, fill='white')
366
367     # Add title to the image
368     plt.imshow(img, cmap='gray') # Ensure the image is displayed as grayscale
369     plt.title(f'Predicted class: {predicted_class_name}') # Display the name of the predicted class
370     plt.axis('off')
371
372     # Get the original filename
373     original_filename = test_generator.filenames[
374         test_generator.batch_index * test_generator.batch_size + j]
374     original_filename = os.path.basename(
375         original_filename) # Get just the file name
376
377     # Generate new filename based on predicted class
378     predicted_filename = original_filename.split('.')[0] +
379         '_' + predicted_class_name + '.' + original_filename.split('.')[1]
380
381     # Save the image with the new filename
382     plt.savefig(os.path.join(save_dir, predicted_filename),
383                 bbox_inches='tight', pad_inches=0)
384     plt.close()
385
386 print("Images have been annotated and saved.")

```

Program CNN 2-Dimensi Percobaan Kelima

Program 5.12: Program CNN 2-Dimensi Percobaan Kelima

```

1 # -*- coding: utf-8 -*-
2 """CNN TA v3 d2.ipynb
3
4 Automatically generated by Colab.
5
6 Original file is located at
7     https://colab.research.google.com/drive/1
8         PlukUiRafS8ZGE8S26rNjQ3WD0SEQ-tP
9
10 import tensorflow as tf

```

```

11 tf.__version__
12
13 from tensorflow.keras.models import Sequential
14 from tensorflow.keras.layers import Conv2D, MaxPooling2D, ←
    Flatten, Dense, Dropout, BatchNormalization
15 from tensorflow.keras.preprocessing.image import ←
    ImageDataGenerator
16 from tensorflow.keras.metrics import Precision, Recall
17 import os
18 import time
19 import time
20 from sklearn.metrics import confusion_matrix
21 import seaborn as sns
22 import matplotlib.pyplot as plt
23 import numpy as np
24
25 # Set the seed for TensorFlow's random number generator
26 np.random.seed(42)
27 tf.random.set_seed(42)
28
29 from google.colab import drive
30 drive.mount('/content/drive')
31
32 datasetDir = "/content/drive/MyDrive/KeperluanTA/another1/out←
    "
33
34 train_datagen = ImageDataGenerator(
35     rescale=1.0/255,
36     shear_range=0.2,
37     zoom_range=0.2,
38     fill_mode="nearest",
39     validation_split=0.1765
40 )
41
42 val_datagen = ImageDataGenerator(
43     rescale=1.0/255,
44     validation_split=0.1765
45 )
46
47 train_generator = train_datagen.flow_from_directory(
48     datasetDir,
49     target_size=(330, 540),
50     batch_size=10,
51     class_mode="binary", # Set class_mode to "categorical" ←
        for multi-class classification
52     color_mode="grayscale", # Generate grayscale images
53     subset="training"
54 )
55

```

```

56 validation_generator = val_datagen.flow_from_directory(
57     datasetDir,
58     target_size=(330, 540),
59     batch_size=10,
60     class_mode="binary", # Set class_mode to "categorical" ←
61     # for multi-class classification
62     color_mode="grayscale", # Generate grayscale images
63     subset="validation"
64 )
65 print(train_generator.class_indices)
66
67 train_generator.image_shape
68
69 # Now your model definition follows
70 model = Sequential()
71
72 # First Convolutional Block
73 model.add(Conv2D(32, (3, 3), activation="relu", input_shape=←
74     =(330, 540, 1)))
75 model.add(BatchNormalization())
76 model.add(MaxPooling2D((2, 2)))
77
78 # Second Convolutional Block
79 model.add(Conv2D(64, (3, 3), activation="relu"))
80 model.add(BatchNormalization())
81 model.add(MaxPooling2D((2, 2)))
82
83 # Flattening and Fully Connected Layers
84 model.add(Flatten())
85 model.add(Dense(128, activation="relu"))
86 model.add(Dropout(0.5)) # Dropout layer to reduce ←
87     overfitting
88
89 # Output Layer for binary classification
90 model.add(Dense(1, activation="sigmoid"))
91
92 model.compile(
93     optimizer="adam",
94     loss="binary_crossentropy",
95     metrics=["accuracy"])
96 )
97
98 from tensorflow.keras.callbacks import EarlyStopping
99
100 # Define EarlyStopping callback
101 early_stopping = EarlyStopping(

```

```

102     monitor='val_loss',      # Metric to monitor
103     patience=10,           # Number of epochs with no ←
104         improvement after which training will be stopped
104     verbose=1,             # To log when training is being ←
105         stopped
105     mode='min',            # Stops training when the ←
106         quantity monitored has stopped decreasing
106     restore_best_weights=True # Restores model weights from ←
107         the epoch with the best value of the monitored ←
107         quantity
108
109 # Fit the model with the EarlyStopping callback
110 history = model.fit(
111     train_generator,
112     validation_data=validation_generator,
113     epochs=100,
114     validation_steps=60,
115     verbose=1,
116     callbacks=[early_stopping] # Include the callback in the←
116         list
117 )
118
119 model.evaluate(validation_generator)
120
121 model.evaluate(train_generator)
122
123 # Specify the directory path
124 save_dir = '/content/drive/MyDrive/KeperluanTA/5rev'
125
126 # Create the directory if it doesn't exist
127 os.makedirs(save_dir, exist_ok=True)
128
129 # Save the model to the specified directory
130 model.save(save_dir + 'model.h5')
131
132 model.save("model.h5")
133
134 import matplotlib.pyplot as plt
135
136 # Plot training & validation accuracy values
137 plt.figure(figsize=(12, 4))
138
139 plt.subplot(1, 2, 1)
140 plt.plot(history.history['accuracy'])
141 plt.plot(history.history['val_accuracy'])
142 plt.title('Model Accuracy')
143 plt.ylabel('Accuracy')
144 plt.xlabel('Epoch')

```

```

145 plt.legend(['Train', 'Validation'], loc='upper left')
146
147 # Plot training & validation loss values
148 plt.subplot(1, 2, 2)
149 plt.plot(history.history['loss'])
150 plt.plot(history.history['val_loss'])
151 plt.title('Model Loss')
152 plt.ylabel('Loss')
153 plt.xlabel('Epoch')
154 plt.legend(['Train', 'Validation'], loc='upper left')
155
156 plt.show()
157
158 import time
159
160 from sklearn.metrics import confusion_matrix
161 import seaborn as sns
162 import matplotlib.pyplot as plt
163 import numpy as np
164
165 # Create new generators without shuffling for evaluating ←
166 # confusion matrix
167 eval_train_generator = train_datagen.flow_from_directory(
168     datasetDir,
169     target_size=(330, 540),
170     batch_size=10, # Adjust this based on your setup
171     class_mode="binary", # Adjusted for binary ←
172     classification
173     color_mode="grayscale",
174     shuffle=False, # Important for matching predictions to ←
175     labels
176     subset="training"
177 )
178
179 eval_validation_generator = val_datagen.flow_from_directory(
180     datasetDir,
181     target_size=(330, 540),
182     batch_size=10, # Match the batch size used for training/←
183     validation
184     class_mode="binary", # Adjusted for binary ←
185     classification
186     color_mode="grayscale",
187     shuffle=False, # Important for matching predictions to ←
188     labels
189     subset="validation"
190
191 # Start timer for inference
192 start_time = time.time()
193
194

```

```

188 # Predict the data
189 train_predictions = model.predict(eval_train_generator, ←
    verbose=1)
190 validation_predictions = model.predict(←
    eval_validation_generator, verbose=1)
191
192 # End timer and calculate inference time
193 inference_time = time.time() - start_time
194 print("Inference time for the test set: {:.2f} seconds".←
    format(inference_time))
195
196 # Convert probabilities to binary predictions based on a 0.5 ←
    threshold
197 train_pred_classes = (train_predictions > 0.5).astype(int).←
    reshape(-1)
198 validation_pred_classes = (validation_predictions > 0.5).←
    astype(int).reshape(-1)
199
200 # True labels (already in binary format)
201 train_true_classes = eval_train_generator.classes
202 validation_true_classes = eval_validation_generator.classes
203
204 # Compute confusion matrices
205 train_cm = confusion_matrix(train_true_classes, ←
    train_pred_classes)
206 validation_cm = confusion_matrix(validation_true_classes, ←
    validation_pred_classes)
207
208 # Plotting the confusion matrices
209 fig, ax = plt.subplots(1, 2, figsize=(12, 5))
210
211 sns.heatmap(train_cm, annot=True, fmt='d', cmap='Blues', ax=←
    ax[0])
212 ax[0].set_title('Training Confusion Matrix')
213 ax[0].set_xlabel('Predicted Labels')
214 ax[0].set_ylabel('True Labels')
215 ax[0].set_xticklabels(['Negative', 'Positive'])
216 ax[0].set_yticklabels(['Negative', 'Positive'])
217
218 sns.heatmap(validation_cm, annot=True, fmt='d', cmap='Greens'←
    , ax=ax[1])
219 ax[1].set_title('Validation Confusion Matrix')
220 ax[1].set_xlabel('Predicted Labels')
221 ax[1].set_ylabel('True Labels')
222 ax[1].set_xticklabels(['Negative', 'Positive'])
223 ax[1].set_yticklabels(['Negative', 'Positive'], va='center')
224
225 plt.tight_layout()
226 plt.show()

```

```

227
228 import time
229 from sklearn.metrics import confusion_matrix
230 import seaborn as sns
231 import matplotlib.pyplot as plt
232 import numpy as np
233
234 testDir = "/content/drive/MyDrive/KeperluanTA/another1/←
235     testout"
236 # Set up the test data generator
237 test_datagen = ImageDataGenerator(rescale=1.0/255)
238
239 test_generator = test_datagen.flow_from_directory(
240     testDir, # Directory with test images
241     target_size=(330, 540),
242     batch_size=10, # Adjust based on your setup
243     class_mode="binary", # Ensure this matches your label ←
244         setup
245     color_mode="grayscale",
246     shuffle=False # Important for matching predictions to ←
247         labels
248 )
249
250 # Start timer for inference
251 start_time = time.time()
252
253 # Predict the data
254 test_predictions = model.predict(test_generator, verbose=1)
255
256 # End timer and calculate inference time
257 inference_time = time.time() - start_time
258 print("Inference time for the test set: {:.2f} seconds".←
259     format(inference_time))
260
261 # Convert probabilities to binary predictions based on a 0.5 ←
262     threshold
263 test_pred_classes = (test_predictions > 0.5).astype(int).←
264         reshape(-1)
265
266 # True labels (already in binary format)
267 test_true_classes = test_generator.classes
268
269 # Compute the confusion matrix
270 test_cm = confusion_matrix(test_true_classes, ←
271     test_pred_classes)
272
273 # Plotting the confusion matrix
274 plt.figure(figsize=(6, 5))

```

```

269 sns.heatmap(test_cm, annot=True, fmt='d', cmap='Purples')
270 plt.title('Test Confusion Matrix')
271 plt.xlabel('Predicted Labels')
272 plt.ylabel('True Labels')
273 plt.xticks([0.5, 1.5], ['Negative', 'Positive'])
274 plt.yticks([0.5, 1.5], ['Negative', 'Positive'], va='center')
275 plt.show()
276
277 import tensorflow as tf
278 from tensorflow.keras.metrics import Precision, Recall
279
280 # Define a custom F1 score metric
281 class F1Score(tf.keras.metrics.Metric):
282     def __init__(self, name='f1_score', **kwargs):
283         super(F1Score, self).__init__(name=name, **kwargs)
284         self.precision = Precision()
285         self.recall = Recall()
286
287     def update_state(self, y_true, y_pred, sample_weight=None):
288         self.precision.update_state(y_true, y_pred, sample_weight)
289         self.recall.update_state(y_true, y_pred, sample_weight)
290
291     def result(self):
292         p = self.precision.result()
293         r = self.recall.result()
294         return 2 * ((p * r) / (p + r + tf.keras.backend.epsilon()))
295
296     def reset_state(self):
297         self.precision.reset_state()
298         self.recall.reset_state()
299
300 # Instantiate the F1Score metric for training, validation, and test
301 f1_score_train = F1Score()
302 f1_score_val = F1Score()
303 f1_score_test = F1Score()
304
305 # Update the state of the F1Score metric with training data
306 f1_score_train.update_state(train_true_classes, train_pred_classes)
307 # Update the state of the F1Score metric with validation data
308 f1_score_val.update_state(validation_true_classes, validation_pred_classes)
309 # Update the state of the F1Score metric with test data

```

```

310 f1_score_test.update_state(test_true_classes, ←
    test_pred_classes)
311
312 # Calculate and print the F1 scores
313 print("Calculated F1 Score for Training Set:", f1_score_train←
    .result().numpy())
314 print("Calculated F1 Score for Validation Set:", f1_score_val←
    .result().numpy())
315 print("Calculated F1 Score for Test Set:", f1_score_test.←
    result().numpy())
316
317 import os
318 import matplotlib.pyplot as plt
319 from tensorflow.keras.preprocessing.image import ←
    ImageDataGenerator
320 from PIL import Image, ImageDraw
321
322 # Assuming `model` and `test_generator` have already been ←
    defined as in your original code.
323
324 # Create a directory to save the annotated images
325 save_dir = '/content/drive/MyDrive/Deteksi/Eks5'
326 os.makedirs(save_dir, exist_ok=True)
327
328 # Iterate over each batch in the test generator
329 for i in range(len(test_generator)):
330     images, labels = test_generator.next()
331     predictions = model.predict(images)
332     pred_classes = (predictions > 0.5).astype(int)
333
334     # Process each image in the batch
335     for j in range(len(images)):
336         # Correctly handle images based on channel ←
            information
            if images[j].shape[-1] == 3: # Color images
                img = Image.fromarray((images[j] * 255).astype('←
                    uint8')).convert('L')
            else: # Grayscale images, potentially with a channel←
                dimension of 1
                # Ensure the image is 2D
                img_array = (images[j] * 255).squeeze() # Remove←
                    singleton dimensions
                img = Image.fromarray(img_array.astype('uint8'), ←
                    'L')
337
338         # Determine predicted class name
339         predicted_label = pred_classes[j]
340         predicted_class_name = "noairgap" if predicted_label ←
            == 1 else "airgap"

```

```

347
348     # Prepare text to draw on the image
349     annotation = f'Predicted class: {predicted_class_name} '
350
351     # Draw text on the image using PIL (default font)
352     draw = ImageDraw.Draw(img)
353     draw.text((10, 10), annotation, fill='white')
354
355     # Add title to the image
356     plt.imshow(img, cmap='gray')    # Ensure the image is displayed as grayscale
357     plt.title(f'Predicted class: {predicted_class_name}')    # Display the name of the predicted class
358     plt.axis('off')
359
360     # Get the original filename
361     original_filename = test_generator.filenames[ test_generator.batch_index * test_generator.batch_size + j]
362     original_filename = os.path.basename( original_filename)    # Get just the file name
363
364     # Generate new filename based on predicted class
365     predicted_filename = original_filename.split('.')[0] +
366                         '_'+ predicted_class_name + '.' + original_filename.split('.')[1]
367
368     # Save the image with the new filename
369     plt.savefig(os.path.join(save_dir, predicted_filename),
370                 bbox_inches='tight', pad_inches=0)
371     plt.close()
372
373 print("Images have been annotated and saved.")

```

Program CNN 2-Dimensi Percobaan Keenam

Program 5.13: Program CNN 2-Dimensi Percobaan Keenam

```

1 # -*- coding: utf-8 -*-
2 """CNN TA v4 d3.ipynb
3
4 Automatically generated by Colab.
5
6 Original file is located at
7     https://colab.research.google.com/drive/1-
8         J9nJXK3vrTZfJ6JgwF8-ERq4FC6YQ5lh
9
10 import tensorflow as tf

```

```

11 tf.__version__
12
13 from tensorflow.keras.models import Sequential
14 from tensorflow.keras.layers import Conv2D, MaxPooling2D, ←
    Flatten, Dense, Dropout, BatchNormalization
15 from tensorflow.keras.preprocessing.image import ←
    ImageDataGenerator
16 from tensorflow.keras.metrics import Precision, Recall
17 import os
18 import time
19 import time
20 from sklearn.metrics import confusion_matrix
21 import seaborn as sns
22 import matplotlib.pyplot as plt
23 import numpy as np
24
25 # Set the seed for TensorFlow's random number generator
26 np.random.seed(42)
27 tf.random.set_seed(42)
28
29 from google.colab import drive
30 drive.mount('/content/drive')
31
32 datasetDir = "/content/drive/MyDrive/KeperluanTA/another1/out←
"
33
34 train_datagen = ImageDataGenerator(
35     rescale=1.0/255,
36     shear_range=0.2,
37     zoom_range=0.2,
38     fill_mode="nearest",
39     validation_split=0.1765
40 )
41
42 val_datagen = ImageDataGenerator(
43     rescale=1.0/255,
44     validation_split=0.1765
45 )
46
47 train_generator = train_datagen.flow_from_directory(
48     datasetDir,
49     target_size=(330, 540),
50     batch_size=10,
51     class_mode="binary", # Set class_mode to "categorical" ←
        for multi-class classification
52     color_mode="grayscale", # Generate grayscale images
53     subset="training"
54 )
55

```

```

56 validation_generator = val_datagen.flow_from_directory(
57     datasetDir,
58     target_size=(330, 540),
59     batch_size=10,
60     class_mode="binary", # Set class_mode to "categorical" ←
61     # for multi-class classification
62     color_mode="grayscale", # Generate grayscale images
63     subset="validation"
64 )
65 print(train_generator.class_indices)
66
67 train_generator.image_shape
68
69 # Now your model definition follows
70 model = Sequential()
71
72 # First Convolutional Block
73 model.add(Conv2D(32, (3, 3), activation='relu', input_shape←
74     =(330, 540, 1), padding='same'))
75 model.add(MaxPooling2D((2, 2)))
76
77 # Second Convolutional Block
78 model.add(Conv2D(64, (3, 3), activation='relu', padding='same←
79     '))
80 model.add(MaxPooling2D((2, 2)))
81
82 # Third Convolutional Block
83 model.add(Conv2D(128, (3, 3), activation='relu', padding='←
84     same'))
85 model.add(MaxPooling2D((2, 2)))
86
87 model.add(Flatten())
88 model.add(Dense(256, activation='relu'))
89 model.add(Dropout(0.5))
90
91 model.add(Dense(1, activation="sigmoid"))
92
93 model.summary()
94
95 model.compile(
96     optimizer="adam",
97     loss="binary_crossentropy",
98     metrics=["accuracy"])
99 )
100

```

```

101 from tensorflow.keras.callbacks import EarlyStopping
102
103 # Define EarlyStopping callback
104 early_stopping = EarlyStopping(
105     monitor='val_loss',          # Metric to monitor
106     patience=10,                # Number of epochs with no ←
107     verbose=1,                  # To log when training is being ←
108     stopped                    # Stopped
109     mode='min',                # Stops training when the ←
110     quantity monitored has stopped decreasing
111     restore_best_weights=True  # Restores model weights from ←
112     the epoch with the best value of the monitored ←
113     quantity
114 )
115
116 # Fit the model with the EarlyStopping callback
117 history = model.fit(
118     train_generator,
119     validation_data=validation_generator,
120     epochs=100,
121     validation_steps=60,
122     verbose=1,
123     callbacks=[early_stopping]  # Include the callback in the←
124     list
125 )
126
127 model.evaluate(validation_generator)
128
129 model.evaluate(train_generator)
130
131 # Specify the directory path
132 save_dir = '/content/drive/MyDrive/KeperluanTA/6rev'
133
134 # Create the directory if it doesn't exist
135 os.makedirs(save_dir, exist_ok=True)
136
137 # Save the model to the specified directory
138 model.save(save_dir + 'model.h5')
139
140 model.save("model.h5")
141
142 import matplotlib.pyplot as plt
143
144 # Plot training & validation accuracy values
145 plt.figure(figsize=(12, 4))
146
147 plt.subplot(1, 2, 1)
148 plt.plot(history.history['accuracy'])

```

```

144 plt.plot(history.history['val_accuracy'])
145 plt.title('Model Accuracy')
146 plt.ylabel('Accuracy')
147 plt.xlabel('Epoch')
148 plt.legend(['Train', 'Validation'], loc='upper left')
149
150 # Plot training & validation loss values
151 plt.subplot(1, 2, 2)
152 plt.plot(history.history['loss'])
153 plt.plot(history.history['val_loss'])
154 plt.title('Model Loss')
155 plt.ylabel('Loss')
156 plt.xlabel('Epoch')
157 plt.legend(['Train', 'Validation'], loc='upper left')
158
159 plt.show()
160
161 import time
162
163 from sklearn.metrics import confusion_matrix
164 import seaborn as sns
165 import matplotlib.pyplot as plt
166 import numpy as np
167
168 # Create new generators without shuffling for evaluating ←
169 # confusion matrix
170 eval_train_generator = train_datagen.flow_from_directory(
171     datasetDir,
172     target_size=(330, 540),
173     batch_size=10, # Adjust this based on your setup
174     class_mode="binary", # Adjusted for binary ←
175     classification
176     color_mode="grayscale",
177     shuffle=False, # Important for matching predictions to ←
178     labels
179     subset="training"
180 )
181
182 eval_validation_generator = val_datagen.flow_from_directory(
183     datasetDir,
184     target_size=(330, 540),
185     batch_size=10, # Match the batch size used for training/←
186     validation
187     class_mode="binary", # Adjusted for binary ←
188     classification
189     color_mode="grayscale",
190     shuffle=False, # Important for matching predictions to ←
191     labels
192     subset="validation"

```

```

187 )
188 # Start timer for inference
189 start_time = time.time()
190
191 # Predict the data
192 train_predictions = model.predict(eval_train_generator, ←
    verbose=1)
193 validation_predictions = model.predict(←
    eval_validation_generator, verbose=1)
194
195 # End timer and calculate inference time
196 inference_time = time.time() - start_time
197 print("Inference time for the test set: {:.2f} seconds".←
    format(inference_time))
198
199 # Convert probabilities to binary predictions based on a 0.5 ←
    threshold
200 train_pred_classes = (train_predictions > 0.5).astype(int).←
    reshape(-1)
201 validation_pred_classes = (validation_predictions > 0.5).←
    astype(int).reshape(-1)
202
203 # True labels (already in binary format)
204 train_true_classes = eval_train_generator.classes
205 validation_true_classes = eval_validation_generator.classes
206
207 # Compute confusion matrices
208 train_cm = confusion_matrix(train_true_classes, ←
    train_pred_classes)
209 validation_cm = confusion_matrix(validation_true_classes, ←
    validation_pred_classes)
210
211 # Plotting the confusion matrices
212 fig, ax = plt.subplots(1, 2, figsize=(12, 5))
213
214 sns.heatmap(train_cm, annot=True, fmt='d', cmap='Blues', ax=←
    ax[0])
215 ax[0].set_title('Training Confusion Matrix')
216 ax[0].set_xlabel('Predicted Labels')
217 ax[0].set_ylabel('True Labels')
218 ax[0].set_xticklabels(['Negative', 'Positive'])
219 ax[0].set_yticklabels(['Negative', 'Positive'])
220
221 sns.heatmap(validation_cm, annot=True, fmt='d', cmap='Greens'←
    , ax=ax[1])
222 ax[1].set_title('Validation Confusion Matrix')
223 ax[1].set_xlabel('Predicted Labels')
224 ax[1].set_ylabel('True Labels')
225 ax[1].set_xticklabels(['Negative', 'Positive'])

```

```

226 ax[1].set_yticklabels(['Negative', 'Positive'], va='center')
227
228 plt.tight_layout()
229 plt.show()
230
231 import time
232 from sklearn.metrics import confusion_matrix
233 import seaborn as sns
234 import matplotlib.pyplot as plt
235 import numpy as np
236
237 testDir = "/content/drive/MyDrive/KeperluanTA/another1/←
    testout"
238
239 # Set up the test data generator
240 test_datagen = ImageDataGenerator(rescale=1.0/255)
241
242 test_generator = test_datagen.flow_from_directory(
243     testDir, # Directory with test images
244     target_size=(330, 540),
245     batch_size=10, # Adjust based on your setup
246     class_mode="binary", # Ensure this matches your label ←
        setup
247     color_mode="grayscale",
248     shuffle=False # Important for matching predictions to ←
        labels
249 )
250
251 # Start timer for inference
252 start_time = time.time()
253
254 # Predict the data
255 test_predictions = model.predict(test_generator, verbose=1)
256
257 # End timer and calculate inference time
258 inference_time = time.time() - start_time
259 print("Inference time for the test set: {:.2f} seconds".←
    format(inference_time))
260
261 # Convert probabilities to binary predictions based on a 0.5 ←
    threshold
262 test_pred_classes = (test_predictions > 0.5).astype(int).←
    reshape(-1)
263
264 # True labels (already in binary format)
265 test_true_classes = test_generator.classes
266
267 # Compute the confusion matrix

```

```

268 test_cm = confusion_matrix(test_true_classes, ←
269     test_pred_classes)
270 # Plotting the confusion matrix
271 plt.figure(figsize=(6, 5))
272 sns.heatmap(test_cm, annot=True, fmt='d', cmap='Purples')
273 plt.title('Test Confusion Matrix')
274 plt.xlabel('Predicted Labels')
275 plt.ylabel('True Labels')
276 plt.xticks([0.5, 1.5], ['Negative', 'Positive'])
277 plt.yticks([0.5, 1.5], ['Negative', 'Positive'], va='center')
278 plt.show()
279
280 import tensorflow as tf
281 from tensorflow.keras.metrics import Precision, Recall
282
283 # Define a custom F1 score metric
284 class F1Score(tf.keras.metrics.Metric):
285     def __init__(self, name='f1_score', **kwargs):
286         super(F1Score, self).__init__(name=name, **kwargs)
287         self.precision = Precision()
288         self.recall = Recall()
289
290     def update_state(self, y_true, y_pred, sample_weight=None):
291         self.precision.update_state(y_true, y_pred, ←
292             sample_weight)
293         self.recall.update_state(y_true, y_pred, ←
294             sample_weight)
295
296     def result(self):
297         p = self.precision.result()
298         r = self.recall.result()
299         return 2 * ((p * r) / (p + r + tf.keras.backend.←
300             epsilon()))
301
302     def reset_state(self):
303         self.precision.reset_state()
304         self.recall.reset_state()
305
306 # Instantiate the F1Score metric for training, validation, ←
307 # and test
308 f1_score_train = F1Score()
309 f1_score_val = F1Score()
310 f1_score_test = F1Score()
311
312 # Update the state of the F1Score metric with training data
313 f1_score_train.update_state(train_true_classes, ←
314     train_pred_classes)

```

```

310 # Update the state of the F1Score metric with validation data
311 f1_score_val.update_state(validation_true_classes, ←
    validation_pred_classes)
312 # Update the state of the F1Score metric with test data
313 f1_score_test.update_state(test_true_classes, ←
    test_pred_classes)
314
315 # Calculate and print the F1 scores
316 print("Calculated F1 Score for Training Set:", f1_score_train←
    .result().numpy())
317 print("Calculated F1 Score for Validation Set:", f1_score_val←
    .result().numpy())
318 print("Calculated F1 Score for Test Set:", f1_score_test.←
    result().numpy())
319
320 import os
321 import matplotlib.pyplot as plt
322 from tensorflow.keras.preprocessing.image import ←
    ImageDataGenerator
323 from PIL import Image, ImageDraw
324
325 # Assuming `model` and `test_generator` have already been ←
    defined as in your original code.
326
327 # Create a directory to save the annotated images
328 save_dir = '/content/drive/MyDrive/Deteksi/Eks6'
329 os.makedirs(save_dir, exist_ok=True)
330
331 # Iterate over each batch in the test generator
332 for i in range(len(test_generator)):
333     images, labels = test_generator.next()
334     predictions = model.predict(images)
335     pred_classes = (predictions > 0.5).astype(int)
336
337     # Process each image in the batch
338     for j in range(len(images)):
339         # Correctly handle images based on channel ←
            information
340         if images[j].shape[-1] == 3: # Color images
341             img = Image.fromarray((images[j] * 255).astype('←
                uint8')).convert('L')
342         else: # Grayscale images, potentially with a channel←
            dimension of 1
343             # Ensure the image is 2D
344             img_array = (images[j] * 255).squeeze() # Remove←
                singleton dimensions
345             img = Image.fromarray(img_array.astype('uint8'), ←
                'L')
346

```

```

347     # Determine predicted class name
348     predicted_label = pred_classes[j]
349     predicted_class_name = "noairgap" if predicted_label ←
350         == 1 else "airgap"
351
352     # Prepare text to draw on the image
353     annotation = f'Predicted class: {predicted_class_name}←
354         }'
355
356     # Draw text on the image using PIL (default font)
357     draw = ImageDraw.Draw(img)
358     draw.text((10, 10), annotation, fill='white')
359
360     # Add title to the image
361     plt.imshow(img, cmap='gray') # Ensure the image is ←
362         displayed as grayscale
363     plt.title(f'Predicted class: {predicted_class_name}')←
364         # Display the name of the predicted class
365     plt.axis('off')
366
367     # Get the original filename
368     original_filename = test_generator.filenames[←
369         test_generator.batch_index * test_generator.←
370         batch_size + j]
371     original_filename = os.path.basename(←
372         original_filename) # Get just the file name
373
374     # Generate new filename based on predicted class
375     predicted_filename = original_filename.split('.')[0] ←
376         + '_' + predicted_class_name + '.' + ←
377         original_filename.split('.')[1]
378
379     # Save the image with the new filename
380     plt.savefig(os.path.join(save_dir, predicted_filename)←
381         ), bbox_inches='tight', pad_inches=0)
382     plt.close()
383
384 print("Images have been annotated and saved.")

```

Program CNN 2-Dimensi Percobaan Ketujuh

Program 5.14: Program CNN 2-Dimensi Percobaan Ketujuh

```

1 # -*- coding: utf-8 -*-
2 """CNN TA v5 d3.ipynb
3
4 Automatically generated by Colab.
5
6 Original file is located at

```

```

7     https://colab.research.google.com/drive/14YJiY7-->
8     QgkdhxnLSej0SxlxH0txi2uiL
9
10    import tensorflow as tf
11    tf.__version__
12
13    from tensorflow.keras.models import Sequential
14    from tensorflow.keras.layers import Conv2D, MaxPooling2D, ←
15        Flatten, Dense, Dropout, BatchNormalization
15    from tensorflow.keras.preprocessing.image import ←
16        ImageDataGenerator
16    from tensorflow.keras.metrics import Precision, Recall
17    import os
18    import time
19    import time
20    from sklearn.metrics import confusion_matrix
21    import seaborn as sns
22    import matplotlib.pyplot as plt
23    import numpy as np
24
25    # Set the seed for TensorFlow's random number generator
26    np.random.seed(42)
27    tf.random.set_seed(42)
28
29    from google.colab import drive
30    drive.mount('/content/drive')
31
32    datasetDir = "/content/drive/MyDrive/KeperluanTA/another1/out←
33
34    train_datagen = ImageDataGenerator(
35        rescale=1.0/255,
36        shear_range=0.2,
37        zoom_range=0.2,
38        fill_mode="nearest",
39        validation_split=0.1765
40    )
41
42    val_datagen = ImageDataGenerator(
43        rescale=1.0/255,
44        validation_split=0.1765
45    )
46
47    train_generator = train_datagen.flow_from_directory(
48        datasetDir,
49        target_size=(330, 540),
50        batch_size=10,

```

```

51     class_mode="binary", # Set class_mode to "categorical" ←
52         for multi-class classification
53     color_mode="grayscale", # Generate grayscale images
54     subset="training"
55 )
56 validation_generator = val_datagen.flow_from_directory(
57     datasetDir,
58     target_size=(330, 540),
59     batch_size=10,
60     class_mode="binary", # Set class_mode to "categorical" ←
61         for multi-class classification
62     color_mode="grayscale", # Generate grayscale images
63     subset="validation"
64 )
65 print(train_generator.class_indices)
66
67 train_generator.image_shape
68
69 model = Sequential()
70
71 # Input layer and first Convolutional block
72 model.add(Conv2D(32, (3, 3), activation="relu", input_shape=
73     (330, 540, 1)))
74 model.add(MaxPooling2D((4, 4)))
75
76 # Second Convolutional block
77 model.add(Conv2D(34, (3, 3), activation="relu"))
78 model.add(MaxPooling2D((2, 2)))
79
80 # Third Convolutional block
81 model.add(Conv2D(32, (3, 3), activation="relu"))
82 model.add(MaxPooling2D((2, 2)))
83
84 # Fourth Convolutional block
85 model.add(Conv2D(32, (3, 3), activation="relu"))
86 model.add(MaxPooling2D((2, 2)))
87
88 # Fifth Convolutional block
89 model.add(Conv2D(32, (3, 3), activation="relu"))
90
91 # Sixth Convolutional block
92 model.add(Conv2D(32, (3, 3), activation="relu"))
93
94 # Flattening layer
95 model.add(Flatten())
96 model.add(Dense(1, activation="sigmoid"))

```

```

97
98 model.summary()
99
100 model.compile(
101     optimizer="adam",
102     loss="binary_crossentropy",
103     metrics=["accuracy"]
104 )
105
106 from tensorflow.keras.callbacks import EarlyStopping
107
108 # Define EarlyStopping callback
109 early_stopping = EarlyStopping(
110     monitor='val_loss',          # Metric to monitor
111     patience=10,                # Number of epochs with no ←
112     verbose=1,                  # To log when training is being ←
113     stopped                     # Stopped training when the ←
114     mode='min',                 # quantity monitored has stopped decreasing
115     restore_best_weights=True   # Restores model weights from ←
116     the epoch with the best value of the monitored ←
117     quantity
118 )
119
120 # Fit the model with the EarlyStopping callback
121 history = model.fit(
122     train_generator,
123     validation_data=validation_generator,
124     epochs=100,
125     validation_steps=60,
126     verbose=1,
127     callbacks=[early_stopping]    # Include the callback in the←
128     list
129 )
130
131 # Specify the directory path
132 save_dir = '/content/drive/MyDrive/KeperluanTA/7rev'
133
134 # Create the directory if it doesn't exist
135 os.makedirs(save_dir, exist_ok=True)
136
137 # Save the model to the specified directory
138 model.save(save_dir + 'model.h5')
139

```

```

140 model.save("model.h5")
141
142 import matplotlib.pyplot as plt
143
144 # Plot training & validation accuracy values
145 plt.figure(figsize=(12, 4))
146
147 plt.subplot(1, 2, 1)
148 plt.plot(history.history['accuracy'])
149 plt.plot(history.history['val_accuracy'])
150 plt.title('Model Accuracy')
151 plt.ylabel('Accuracy')
152 plt.xlabel('Epoch')
153 plt.legend(['Train', 'Validation'], loc='upper left')
154
155 # Plot training & validation loss values
156 plt.subplot(1, 2, 2)
157 plt.plot(history.history['loss'])
158 plt.plot(history.history['val_loss'])
159 plt.title('Model Loss')
160 plt.ylabel('Loss')
161 plt.xlabel('Epoch')
162 plt.legend(['Train', 'Validation'], loc='upper left')
163
164 plt.show()
165
166 import time
167
168 from sklearn.metrics import confusion_matrix
169 import seaborn as sns
170 import matplotlib.pyplot as plt
171 import numpy as np
172
173 # Create new generators without shuffling for evaluating ←
174 # confusion matrix
175 eval_train_generator = train_datagen.flow_from_directory(
176     datasetDir,
177     target_size=(330, 540),
178     batch_size=10, # Adjust this based on your setup
179     class_mode="binary", # Adjusted for binary ←
180     classification
181     color_mode="grayscale",
182     shuffle=False, # Important for matching predictions to ←
183     labels
184     subset="training"
185 )
186
187 eval_validation_generator = val_datagen.flow_from_directory(
188     datasetDir,

```

```

186     target_size=(330, 540),
187     batch_size=10,  # Match the batch size used for training/←
188         validation
189     class_mode="binary",  # Adjusted for binary ←
190         classification
191     color_mode="grayscale",
192     shuffle=False,  # Important for matching predictions to ←
193         labels
194     subset="validation"
195 )
196 # Start timer for inference
197 start_time = time.time()
198
199 # Predict the data
200 train_predictions = model.predict(eval_train_generator, ←
201     verbose=1)
202 validation_predictions = model.predict(←
203     eval_validation_generator, verbose=1)
204
205 # End timer and calculate inference time
206 inference_time = time.time() - start_time
207 print("Inference time for the test set: {:.2f} seconds".←
208     format(inference_time))
209
210 # Convert probabilities to binary predictions based on a 0.5 ←
211     threshold
212 train_pred_classes = (train_predictions > 0.5).astype(int).←
213     reshape(-1)
214 validation_pred_classes = (validation_predictions > 0.5).←
215     astype(int).reshape(-1)
216
217 # True labels (already in binary format)
218 train_true_classes = eval_train_generator.classes
219 validation_true_classes = eval_validation_generator.classes
220
221 # Compute confusion matrices
222 train_cm = confusion_matrix(train_true_classes, ←
223     train_pred_classes)
224 validation_cm = confusion_matrix(validation_true_classes, ←
225     validation_pred_classes)
226
227 # Plotting the confusion matrices
228 fig, ax = plt.subplots(1, 2, figsize=(12, 5))
229
230 sns.heatmap(train_cm, annot=True, fmt='d', cmap='Blues', ax=←
231     ax[0])
232 ax[0].set_title('Training Confusion Matrix')
233 ax[0].set_xlabel('Predicted Labels')
234 ax[0].set_ylabel('True Labels')

```

```

223 ax[0].set_xticklabels(['Negative', 'Positive'])
224 ax[0].set_yticklabels(['Negative', 'Positive'])
225
226 sns.heatmap(validation_cm, annot=True, fmt='d', cmap='Greens'←
    , ax=ax[1])
227 ax[1].set_title('Validation Confusion Matrix')
228 ax[1].set_xlabel('Predicted Labels')
229 ax[1].set_ylabel('True Labels')
230 ax[1].set_xticklabels(['Negative', 'Positive'])
231 ax[1].set_yticklabels(['Negative', 'Positive'], va='center')
232
233 plt.tight_layout()
234 plt.show()
235
236 import time
237 from sklearn.metrics import confusion_matrix
238 import seaborn as sns
239 import matplotlib.pyplot as plt
240 import numpy as np
241
242 testDir = "/content/drive/MyDrive/KeperluanTA/another1/←
    testout"
243
244 # Set up the test data generator
245 test_datagen = ImageDataGenerator(rescale=1.0/255)
246
247 test_generator = test_datagen.flow_from_directory(
248     testDir, # Directory with test images
249     target_size=(330, 540),
250     batch_size=10, # Adjust based on your setup
251     class_mode="binary", # Ensure this matches your label ←
        setup
252     color_mode="grayscale",
253     shuffle=False # Important for matching predictions to ←
        labels
254 )
255
256 # Start timer for inference
257 start_time = time.time()
258
259 # Predict the data
260 test_predictions = model.predict(test_generator, verbose=1)
261
262 # End timer and calculate inference time
263 inference_time = time.time() - start_time
264 print("Inference time for the test set: {:.2f} seconds".←
    format(inference_time))
265

```

```

266 # Convert probabilities to binary predictions based on a 0.5 ←
267 # threshold
268 test_pred_classes = (test_predictions > 0.5).astype(int).←
269 reshape(-1)
270
271 # True labels (already in binary format)
272 test_true_classes = test_generator.classes
273
274 # Compute the confusion matrix
275 test_cm = confusion_matrix(test_true_classes, ←
276     test_pred_classes)
277
278 # Plotting the confusion matrix
279 plt.figure(figsize=(6, 5))
280 sns.heatmap(test_cm, annot=True, fmt='d', cmap='Purples')
281 plt.title('Test Confusion Matrix')
282 plt.xlabel('Predicted Labels')
283 plt.ylabel('True Labels')
284 plt.xticks([0.5, 1.5], ['Negative', 'Positive'])
285 plt.yticks([0.5, 1.5], ['Negative', 'Positive'], va='center')
286 plt.show()
287
288 # Define a custom F1 score metric
289 class F1Score(tf.keras.metrics.Metric):
290     def __init__(self, name='f1_score', **kwargs):
291         super(F1Score, self).__init__(name=name, **kwargs)
292         self.precision = Precision()
293         self.recall = Recall()
294
295     def update_state(self, y_true, y_pred, sample_weight=None):
296         self.precision.update_state(y_true, y_pred, ←
297             sample_weight)
298         self.recall.update_state(y_true, y_pred, ←
299             sample_weight)
300
301     def result(self):
302         p = self.precision.result()
303         r = self.recall.result()
304         return 2 * ((p * r) / (p + r + tf.keras.backend.←
305             epsilon()))
306
307     def reset_state(self):
308         self.precision.reset_state()
309         self.recall.reset_state()

```

```

308 # Instantiate the F1Score metric for training, validation, ←
    and test
309 f1_score_train = F1Score()
310 f1_score_val = F1Score()
311 f1_score_test = F1Score()
312
313 # Update the state of the F1Score metric with training data
314 f1_score_train.update_state(train_true_classes, ←
    train_pred_classes)
315 # Update the state of the F1Score metric with validation data
316 f1_score_val.update_state(validation_true_classes, ←
    validation_pred_classes)
317 # Update the state of the F1Score metric with test data
318 f1_score_test.update_state(test_true_classes, ←
    test_pred_classes)
319
320 # Calculate and print the F1 scores
321 print("Calculated F1 Score for Training Set:", f1_score_train.←
    .result().numpy())
322 print("Calculated F1 Score for Validation Set:", f1_score_val.←
    .result().numpy())
323 print("Calculated F1 Score for Test Set:", f1_score_test.←
    .result().numpy())
324
325 import os
326 import matplotlib.pyplot as plt
327 from tensorflow.keras.preprocessing.image import ←
    ImageDataGenerator
328 from PIL import Image, ImageDraw
329
330 # Assuming `model` and `test_generator` have already been ←
    defined as in your original code.
331
332 # Create a directory to save the annotated images
333 save_dir = '/content/drive/MyDrive/Deteksi/Eks7'
334 os.makedirs(save_dir, exist_ok=True)
335
336 # Iterate over each batch in the test generator
337 for i in range(len(test_generator)):
338     images, labels = test_generator.next()
339     predictions = model.predict(images)
340     pred_classes = (predictions > 0.5).astype(int)
341
342     # Process each image in the batch
343     for j in range(len(images)):
344         # Correctly handle images based on channel ←
            information
345         if images[j].shape[-1] == 3: # Color images

```

```

346         img = Image.fromarray((images[j] * 255).astype('←
347             uint8')).convert('L')
348     else: # Grayscale images, potentially with a channel←
349         dimension of 1
350         # Ensure the image is 2D
351         img_array = (images[j] * 255).squeeze() # Remove←
352             singleton dimensions
353         img = Image.fromarray(img_array.astype('uint8'), ←
354             'L')
355
356     # Determine predicted class name
357     predicted_label = pred_classes[j]
358     predicted_class_name = "noairgap" if predicted_label ←
359         == 1 else "airgap"
360
361     # Prepare text to draw on the image
362     annotation = f'Predicted class: {predicted_class_name}←
363         }'
364
365     # Draw text on the image using PIL (default font)
366     draw = ImageDraw.Draw(img)
367     draw.text((10, 10), annotation, fill='white')
368
369     # Add title to the image
370     plt.imshow(img, cmap='gray') # Ensure the image is ←
371         displayed as grayscale
372     plt.title(f'Predicted class: {predicted_class_name}')←
373         # Display the name of the predicted class
374     plt.axis('off')
375
376     # Get the original filename
377     original_filename = test_generator.filenames[←
378         test_generator.batch_index * test_generator.←
379             batch_size + j]
380     original_filename = os.path.basename(←
381         original_filename) # Get just the file name
382
383     # Generate new filename based on predicted class
384     predicted_filename = original_filename.split('.')[0] ←
385         + '_' + predicted_class_name + '.' + ←
386         original_filename.split('.')[1]
387
388     # Save the image with the new filename
389     plt.savefig(os.path.join(save_dir, predicted_filename)←
390         ), bbox_inches='tight', pad_inches=0)
391     plt.close()
392
393 print("Images have been annotated and saved.")

```

Program Training YOLOv9

Program 5.15: Program Training YOLOv9

```
1 !nvidia-smi
2 import os
3 HOME = os.getcwd()
4 print(HOME)
5
6 !git clone https://github.com/WongKinYiu/yolov9.git
7 %cd yolov9
8 !pip install -r requirements.txt -q
9
10 !pip install -q roboflow
11 import roboflow
12 from IPython.display import Image
13
14 !mkdir -p {HOME}/weights
15 !wget -P {HOME}/weights -q https://github.com/WongKinYiu/←
    yolov9/releases/download/v0.1/yolov9-c.pt
16 !wget -P {HOME}/weights -q https://github.com/WongKinYiu/←
    yolov9/releases/download/v0.1/yolov9-e.pt
17 !wget -P {HOME}/weights -q https://github.com/WongKinYiu/←
    yolov9/releases/download/v0.1/gelan-c.pt
18 !wget -P {HOME}/weights -q https://github.com/WongKinYiu/←
    yolov9/releases/download/v0.1/gelan-e.pt
19 !ls -la {HOME}/weights
20
21 %cd {HOME}/yolov9
22 from roboflow import Roboflow
23 rf = Roboflow(api_key="5dDggxfyiXKXX0iHfrDv")
24 project = rf.workspace("fp-pcv-z6u2a").project("detect-cl54w"←
    )
25 version = project.version(1)
26 dataset = version.download("yolov9")
27
28 %cd {HOME}/yolov9
29 !python train.py \
30 --batch 8 --epochs 10 --img 640 --device 0 --min-items 0 --←
    close-mosaic 15 \
31 --data {dataset.location}/data.yaml \
32 --weights {HOME}/weights/gelan-c.pt \
33 --cfg models/detect/gelan-c.yaml \
34 --hyp hyp.scratch-high.yaml
35
36 !ls {HOME}/yolov9/runs/train/exp/
37 Image(filename=f"{HOME}/yolov9/runs/train/exp/results.png", ←
    width=1000)
38 Image(filename=f"{HOME}/yolov9/runs/train/exp/←
    confusion_matrix.png", width=1000)
```

```

39 Image(filename=f"{HOME}/yolov9/runs/train/exp/val_batch0_pred←
    .jpg", width=1000)
40
41 %cd {HOME}/yolov9
42 !python val.py \
43 --img 640 --batch 8 --conf 0.001 --iou 0.7 --device 0 \
44 --data {dataset.location}/data.yaml \
45 --weights {HOME}/yolov9/runs/train/exp/weights/best.pt
46
47 !python detect.py \
48 --img 640 --conf 0.1 --device 0 \
49 --weights {HOME}/yolov9/runs/train/exp/weights/best.pt \
50 --source {dataset.location}/valid/images

```

Program Pengujian Model CNN 2-Dimensi

Program 5.16: Pengujian Model CNN 2-Dimensi

```

1 from google.colab import drive
2 drive.mount('/content/drive')
3
4 import os
5 import matplotlib.pyplot as plt
6 from tensorflow.keras.preprocessing.image import ←
    ImageDataGenerator
7 from PIL import Image, ImageDraw
8
9 # Assuming `model` and `test_generator` have already been ←
    defined as in your original code.
10
11 # Create a directory to save the annotated images
12 save_dir = '/content/drive/MyDrive/Deteksi/Eks6'
13 os.makedirs(save_dir, exist_ok=True)
14
15 # Iterate over each batch in the test generator
16 for i in range(len(test_generator)):
17     images, labels = test_generator.next()
18     predictions = model.predict(images)
19     pred_classes = (predictions > 0.5).astype(int)
20
21     # Process each image in the batch
22     for j in range(len(images)):
23         # Correctly handle images based on channel ←
            information
24         if images[j].shape[-1] == 3: # Color images
25             img = Image.fromarray((images[j] * 255).astype('←
                uint8')).convert('L')
26         else: # Grayscale images, potentially with a channel←
            dimension of 1
27             # Ensure the image is 2D

```

```

28     img_array = (images[j] * 255).squeeze() # Remove←
29         singleton dimensions
30     img = Image.fromarray(img_array.astype('uint8'), ←
31         'L')
32
33     # Determine predicted class name
34     predicted_label = pred_classes[j]
35     predicted_class_name = "noairgap" if predicted_label ←
36         == 1 else "airgap"
37
38     # Prepare text to draw on the image
39     annotation = f'Predicted class: {predicted_class_name}←
40         }'
41
42     # Draw text on the image using PIL (default font)
43     draw = ImageDraw.Draw(img)
44     draw.text((10, 10), annotation, fill='white')
45
46     # Add title to the image
47     plt.imshow(img, cmap='gray') # Ensure the image is ←
48         displayed as grayscale
49     plt.title(f'Predicted class: {predicted_class_name}')←
50         # Display the name of the predicted class
51     plt.axis('off')
52
53     # Get the original filename
54     original_filename = test_generator.filenames[←
55         test_generator.batch_index * test_generator.←
56         batch_size + j]
57     original_filename = os.path.basename(←
58         original_filename) # Get just the file name
59
60     # Generate new filename based on predicted class
61     predicted_filename = original_filename.split('.')[0] ←
62         + '_' + predicted_class_name + '.' + ←
63         original_filename.split('.')[1]
64
65     # Save the image with the new filename
66     plt.savefig(os.path.join(save_dir, predicted_filename)←
67         ), bbox_inches='tight', pad_inches=0)
68     plt.close()
69
70 print("Images have been annotated and saved.")

```

Program Pengujian Model YOLOv9

Program 5.17: Program Pengujian Model YOLOv9

```

1 from google.colab import drive
2 drive.mount('/content/drive')

```

```
3
4 import os
5 base_dir = '/content/drive/MyDrive/KumpulanModel'
6 model_path = os.path.join(base_dir, 'best.pt')
7 image_path = os.path.join(base_dir, 'picture/wn.jpg')
8
9 !git clone https://github.com/WongKinYiu/yolov9.git
10 %cd yolov9
11 !pip install -r requirements.txt -q
12
13 !python detect.py --weights '{model_path}' --img 640 --conf ←
    0.25 --source '{image_path}'
14 from IPython.display import Image, display
15 for imageName in os.listdir('/content/yolov9/runs/detect/exp'←
    ):
16     display(Image(filename='/content/yolov9/runs/detect/exp/'←
        + imageName))
```

[Halaman ini sengaja dikosongkan]

BIOGRAFI PENULIS



Gilang Maulana, atau yang biasa dikenal dengan Gilang, lahir pada tanggal 30 Mei 2002 di kota Tuban. Penulis merupakan anak kedua dari dua bersaudara yang tinggal dan besar di Tuban, Jawa Timur. Setelah lulus dari SMA Negeri 1 Tuban, penulis kemudian melanjutkan pendidikan ke jenjang strata satu di Departemen Teknik Komputer, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember mulai tahun 2020.

Penulis merupakan orang yang aktif berorganisasi, dan memiliki berbagai *softskill* serta *hardskill*. Hal ini dibuktikan dengan rekam jejak organisasi dan kepanitiaan dari penulis seperti, Wakil Kepala Departemen Hubungan Luar Himpunan Mahasiswa Teknik Komputer ITS (HIMATEKKOM-ITS), Wakil Ketua Divisi Desain dan Dokumentasi *Multimedia and Game Event*, Ketua Capstone Project pada Bangkit Academy yang diselenggarakan oleh Google, Cloud Engineer pada Bangkit Academy Capstone project, dan masih banyak lagi.

[Halaman ini sengaja dikosongkan]