OPTIMASI DETEKSI DINI RETINOPATI DIABETIK MENGGUNAKAN PREPROCESSING DAN AUGMENTASI CITRA FUNDUS

TUGAS AKHIR

ALVIN PRATIKTA WIDAJAT

NIM: 311910003

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

MALANG

2024

LEMBAR PENGESAHAN
TUGAS AKHIR

OPTIMASI DETEKSI RETINOPATI DIABETIK MENGGUNAKAN PREPROCESSING DAN AUGMENTASI CITRA FUNDUS

Oleh :

ALVIN PRATIKTA WIDAJAT

NIM: 311910003

dari :

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

Dosen Pembimbing 1,

Windra Swastika, S.Kom., MT., Ph.D.
NIP. 20070039

Dosen Pembimbing 2,

Paulus Lucky Tirma Irawan, S.Kom., MT.
NIP. 20100005

Dekan Fakultas Teknologi dan Desain

Prof. Dr. Eng. Romy Budhi Widodo

NIP.20070035

OPTIMASI DETEKSI DINI RETINOPATI DIABETIK MENGGUNAKAN PREPROCESSING DAN AUGMENTASI CITRA FUNDUS

Alvin Pratikta Widajat, Windra Swastika, Paulus Lucky Tirma Irawan
Universitas Ma Chung

Abstrak

Retinopati Diabetik (RD) merupakan komplikasi umum pada diabetes melitus yang seringkali menyebabkan kebutaan pada usia produktif. Deteksi dini RD sangat penting, namun metode manual memakan waktu dan diagnosis para ahli dapat berbeda berdasarkan asumsi pribadi. Oleh karena itu, penelitian ini mengusulkan metode deep learning guna menghindari kesalahan diagnosis. Penelitian ini melibatkan empat arsitektur (Inception-v4, ResNet-50, VGG-19, YOLO v5Nano) dengan teknik *preprocessing* (*cropping*, histogram *equalization*) dan augmentasi *flipping*. Dataset diunduh dari laman Kaggle lalu di-*sampling* untuk menciptakan dataset asli. Selanjutnya, dilakukan *preprocessing* citra dan augmentasi data. Empat arsitektur dilatih untuk menghasilkan empat model, *confusion matrix*, dan nilai akurasi yang dievaluasi menggunakan uji normalitas dan uji statistik. Selanjutnya, pengaruh *preprocessing* dan augmentasi data terhadap performa model dianalisis dengan melatih model pada berbagai kombinasi dataset. Hasil penelitian menunjukkan bahwa YOLO v5Nano unggul, mencapai akurasi pelatihan sebesar 0,93 dan akurasi pengujian rata-rata sebesar 0,73. Augmentasi *flipping* terbukti efektif, dengan akurasi pelatihan sebesar 0,91 dan akurasi pengujian sebesar 0,93. Secara keseluruhan, penerapan arsitektur deep learning dengan preprocessing dan augmentasi data berhasil meningkatkan deteksi dini RD. Arsitektur YOLO v5Nano dianggap optimal, sementara augmentasi *flipping* menunjukkan kinerja superior. Temuan ini berkontribusi pada pengurangan risiko kebutaan melalui deteksi dini retinopati diabetik.

Kata kunci: Retinopati Diabetik, *Deep Learning*, *Preprocessing* Citra, Augmentasi Data

OPTIMIZATION OF EARLY DETECTION OF DIABETIC RETINOPATHY USING FUNDUS IMAGE PREPROCESSING AND AUGMENTATION

Alvin Pratikta Widajat, Windra Swastika, Paulus Lucky Tirma Irawan
Universitas Ma Chung

Abstract

Diabetic Retinopathy (DR) is a complication of diabetes mellitus that often leads to blindness at a productive age. Early detection of DR is crucial, but manual methods are time-consuming, and expert diagnoses can vary based on personal assumptions. Therefore, this research proposes a deep learning method to avoid diagnostic errors. This study involves four architectures (Inception-v4, ResNet-50, VGG-19, YOLO v5Nano) with preprocessing techniques (cropping, histogram equalization) and flipping augmentation. The dataset was downloaded from Kaggle and then sampled to create an original dataset. Next, image preprocessing and data augmentation were performed. Four architectures were trained to produce four models, confusion matrices, and accuracy values evaluated using normality tests and statistical tests. Furthermore, the influence of preprocessing and data augmentation on model performance was analyzed by training models on various dataset combinations. The results show that YOLO v5Nano excels, achieving a training accuracy of 0.93 and an average testing accuracy of 0.73. Flipping augmentation proved effective, with a training accuracy of 0.91 and a testing accuracy of 0.93. Overall, the implementation of deep learning architectures with preprocessing and data augmentation successfully improves early detection of DR. The YOLO v5Nano architecture is considered optimal, while flipping augmentation demonstrates superior performance. These findings contribute to reducing the risk of blindness through early detection of diabetic retinopathy.

Keywords: Diabetic Retinopathy, Deep Learning, Image Preprocessing, Data Augmentation

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas anugerah-Nya tugas dan laporan Tugas Akhir ini dapat selesai dengan baik. Laporan ini ditulis untuk melaporkan hasil pencapaian dari proses pengerjaan Tugas Akhir yang telah dilaksanakan.

Pada kesempatan ini, penulis ingin berterima kasih kepada seluruh pihak yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini. Ucapan terima kasih ini penulis sampaikan kepada:

1. Bapak Windra Swastika, S.Kom., MT., Ph.D., selaku Pembimbing I,
2. Bapak Paulus Lucky Tirma Irawan, S.Kom., MT., selaku Pembimbing II,
3. Orang tua penulis yang memberikan dukungan dan semangat sehingga proyek Tugas Akhir ini bisa selesai,
4. Teman-teman penulis yang sudah memberikan bantuan dan kontribusinya dalam pengerjaan proyek Tugas Akhir ini.

Laporan ini ditulis berdasarkan hasil dari Tugas Akhir dengan judul "Optimasi Deteksi Dini Retinopati Diabetik Menggunakan Preprocessing dan Augmentasi Citra Fundus". Tugas Akhir ini adalah mata kuliah yang wajib diselesaikan oleh mahasiswa Teknik Informatika Universitas Ma Chung Malang sebagai salah satu prasyarat kelulusan.

Malang, 03 Maret 2024

Alvin Pratikta Widajat

311910003

DAFTAR ISI

LEMBAR PENGESAHAN		i
Abstrak		ii
Abstract		iii
KATA PENGANTAR		iv
DAFTAR ISI		v
DAFTAR GAMBAR		x
DAFTAR TABEL		xiii
BAB I PENDAHULUAN		1
1.1	Latar Belakang	1
1.2	Identifikasi Masalah	2
1.3	Batasan Masalah	3
1.4	Rumusan Masalah	3
1.5	Tujuan Penelitian	3
1.6	Manfaat Penelitian	3
1.7	Luaran Penelitian	4
1.8	Sistematika Penulisan	4
BAB II TINJAUAN PUSTAKA		6
2.1	Diabetes Melitus	6
2.2	Retinopati Diabetik	7
2.3	Citra	9
2.4	Intensitas Citra	10
2.5	*Preprocessing* Citra	11
2.5.1	*Cropping*	13
2.5.2	*Contrast Adjustment*	14

2.5.3 *Histogram Equalization* 16
2.5.4 *Brightness Adjustment* 20
2.6 Augmentasi Data 23
2.7 *Artificial Intelligence* 24
2.8 *Machine Learning* 25
2.9 *Deep Learning* 27
2.10 *Convolutional Neural Network* (CNN) 28
2.11 Inception-v4 29
2.12 ResNet-50 30
2.13 VGG-19 31
2.14 YOLO V5 31
2.15 *Confusion Matrix* 33
2.16 Python 34
2.17 Kaggle 34
2.18 Google Colaboratory 35
2.19 Microsoft Visual Studio Code 35
2.20 Uji Normalitas 36
2.20.1 Uji Shapiro Wilk 36
2.21 Uji Rata-Rata (*Mean*) 37
2.21.1 Uji Anova 37
2.21.2 Uji Tukey HSD 39
2.21.3 Uji Kruskal-Wallis 39
2.21.4 Uji Mann-Whitney U 40
2.22 Formulasi Hipotesis 41
2.23 Penelitian Terdahulu 42
BAB III RANCANGAN PENELITIAN 44

3.1 Tahapan Penelitian 44
3.2 Analisis Kebutuhan Penelitian 44
3.3 Desain Penelitian 45
3.3.1 Pengunduhan Data 46
3.3.2 *Sampling* Data 48
3.3.3 *Preprocessing* dan Augmentasi Data 48
3.3.4 Pelatihan Model 50
3.3.5 Pengujian Model 50
3.3.5.1 *Testing* Model 52
3.3.5.2 Uji Normalitas 52
3.3.5.3 Uji ANOVA 52
3.3.5.4 Uji Tukey HSD 52
3.3.5.5 Uji Kruskal-Wallis 53
3.3.5.6 Uji Mann-Whitney U 53
3.3.6 Pelatihan Model Analisis *Preprocessing* dan Augmentasi Data 53
3.3.7 Pengujian Model Analisis *Preprocessing* dan Augmentasi Data 55
3.3.7.1 *Testing* Model 55
3.3.7.2 Uji Normalitas 56
3.3.7.3 Uji ANOVA 56
3.3.7.4 Uji Tukey HSD 56
3.3.7.5 Uji Kruskal-Wallis 57
3.3.7.6 Uji Mann-Whitney U 57
3.4 Interpretasi Hasil Penelitian 57
BAB IV HASIL DAN PEMBAHASAN 58

4.1	Hasil Pelatihan Model	58
4.1.1	Hasil Pelatihan ResNet-50	59
4.1.2	Hasil Pelatihan Inception-v4	61
4.1.3	Hasil Pelatihan VGG-19	63
4.1.4	Hasil Pelatihan YOLO v5Nano	65
4.2	Hasil Pengujian Model	66
4.2.1	Hasil Uji Shapiro-Wilk Pengujian Model	67
4.2.2	Hasil Uji Kruskal-Wallis Pengujian Model	69
4.2.3	Hasil Uji Mann-Whitney U Pengujian Model	70
4.3	Hasil Pelatihan Model Analisis *Preprocessing* dan Augmentasi Data	73
4.3.1	Hasil Pelatihan Dataset Asli	74
4.3.2	Hasil Pelatihan Dataset *Cropping*	76
4.3.3	Hasil Pelatihan Dataset Histeq	78
4.3.4	Hasil Pelatihan Dataset *Flipping*	80
4.3.5	Hasil Pelatihan Dataset *Cropping* dan Histeq	82
4.3.6	Hasil Pelatihan Dataset *Cropping* dan *Flipping*	84
4.3.7	Hasil Pelatihan Dataset Histeq dan *Flipping*	86
4.3.8	Hasil Pelatihan Dataset Setelah *Preprocessing* dan Augmentasi	88
4.4	Hasil Pengujian Model Analisis *Preprocessing* dan Augmentasi Data	90
4.4.1	Hasil Uji Shapiro-Wilk Pengujian Model Analisis *Preprocessing* dan Augmentasi Data	91
4.4.2	Hasil Uji Kruskall-Wallis Pengujian Model Analisis *Preprocessing* dan Augmentasi Data	93

4.4.3 Hasil Uji Mann-Whitney U Pengujian Model Analisis *Preprocessing* dan Augmentasi Data 95

BAB V KESIMPULAN DAN SARAN 100

5.1 Kesimpulan 100

5.2 Saran 100

DAFTAR PUSTAKA 101

LAMPIRAN 104

DAFTAR GAMBAR

Gambar 2.1 Citra Fundus, (A) Mata Normal, (B) Mata NPRD Ringan, (C) Mata NPRD Sedang, (D) Mata NPRD Berat, (E) Mata PRD 8

Gambar 2.2 Contoh Intensitas Citra *greyscale* pada matriks 2D 10

Gambar 2.3 Contoh Intensitas Citra RGB pada Matriks 2D 11

Gambar 2.4 Contoh Citra (A) Sebelum Diterapkan *Cropping*, (B) Setelah Diterapkan *Cropping* 13

Gambar 2.5 Contoh Citra (A) Sebelum Diterapkan *Contrast Adjustment*, (B) Setelah Diterapkan *Contrast Adjustment* 15

Gambar 2.6 Contoh Citra (A) Sebelum Diterapkan *Histogram Equalization*, (B) Setelah Diterapkan *Histogram Equalization* 17

Gambar 2.7 Contoh Matriks 2D Nilai Intensitas Citra yang Berukuran 10x10 18

Gambar 2.8 Matriks 2D Intensitas Citra Hasil *Histogram Equalization* dari Gambar 2.7 20

Gambar 2.9 Contoh Citra (A) Sebelum Diterapkan *Brightness Adjustment*, (B) Setelah Diterapkan *Brightness Adjustment* 22

Gambar 2.10 Contoh *Brightness Adjustment* Pada Citra Berbentuk Matriks 2D 23

Gambar 2.11 Contoh Arsitektur CNN 28

Gambar 2.12 Contoh Arsitektur Inception-v4 30

Gambar 2.13 Arsitektur ResNet-50 31

Gambar 2.14 Contoh Arsitektur VGGNet 32

Gambar 2.15 Contoh Arsitektur YOLO 33

Gambar 2.16 Contoh *Confusion Matrix* Ukuran 2x2 33

Gambar 3.1 Tahapan Penelitian 44

Gambar 3.2 Desain Penelitian 45

Gambar 3.3 Contoh Citra Fundus (A) Normal, (B) RD Ringan, (C) RD Sedang, (D) RD Berat, (E) RD Proliferatif 47

Gambar 3.4 Contoh Citra Fundus (A) Sebelum *Cropping*, (B) Setelah *Cropping* 48

Gambar 3.5 Contoh Citra Fundus (A) Sebelum *Histogram Equalization*, (B) Setelah *Histogram Equalization* 49
Gambar 3.6 Contoh Citra (A) Sebelum *Flipping*, (B) Setelah *Horizontal Flipping*, (C) Setelah *Vertical Flipping*, (D) Setelah *Horizontal* dan *Vertical Flipping* 49
Gambar 3.7 *Flowchart* Pengujian Model 50
Gambar 3.8 *Flowchart* Pengujian Model Analisis *Preprocessing* dan Augmentasi Data 55
Gambar 4.1 Grafik Akurasi *Training* dan Validasi ResNet-50 59
Gambar 4.2 Grafik *Loss Training* dan Validasi ResNet-50 59
Gambar 4.3 *Confusion Matrix Testing* ResNet-50 60
Gambar 4.4 Grafik Akurasi *Training* dan Validasi Inception-v4 61
Gambar 4.5 Grafik *Loss Training* dan Validasi Inception-v4 61
Gambar 4.6 *Confusion Matrix Testing* Inception-v4 62
Gambar 4.7 Grafik Akurasi *Training* dan Validasi VGG-19 63
Gambar 4.8 Grafik *Loss Training* dan Validasi VGG-19 63
Gambar 4.9 *Confusion Matrix Testing* VGG-19 64
Gambar 4.10 Grafik Akurasi *Training* dan Validasi YOLO v5Nano 65
Gambar 4.11 Grafik *Loss Training* dan Validasi YOLO v5Nano 65
Gambar 4.12 *Confusion Matrix* YOLO v5Nano 66
Gambar 4.13 Grafik Akurasi *Training* Dataset Asli 74
Gambar 4.14 Grafik *Loss Training* dan Validasi Dataset Asli 75
Gambar 4.15 *Confusion Matrix* Dataset Asli 75
Gambar 4.16 Grafik Akurasi *Training* Dataset *Cropping* 76
Gambar 4.17 Grafik *Loss Training* dan Validasi Dataset *Cropping* 77
Gambar 4.18 *Confusion Matrix* Dataset *Cropping* 77
Gambar 4.19 Grafik Akurasi *Training* Dataset Histeq 78
Gambar 4.20 Grafik *Loss Training* dan Validasi Dataset Histeq 79
Gambar 4.21 *Confusion Matrix* Dataset Histeq 79
Gambar 4.22 Grafik Akurasi *Training* Dataset *Flipping* 80
Gambar 4.23 Grafik *Loss Training* dan Validasi Dataset *Flipping* 81
Gambar 4.24 *Confusion Matrix* Dataset *Flipping* 81

Gambar 4.25 Grafik Akurasi *Training* Dataset *Cropping* dan Histeq 82
Gambar 4.26 Grafik *Loss Training* dan Validasi Dataset *Cropping* dan Histeq 83
Gambar 4.27 *Confusion Matrix* Dataset *Cropping* dan Histeq 83
Gambar 4.28 Grafik Akurasi *Training* Dataset *Cropping* dan *Flipping* 84
Gambar 4.29 Grafik *Loss Training* dan Validasi Dataset *Cropping* dan *Flipping* 85
Gambar 4.30 *Confusion Matrix* Dataset *Cropping* dan *Flipping* 85
Gambar 4.31 Grafik Akurasi *Training* Dataset Histeq dan *Flipping* 86
Gambar 4.32 Grafik *Loss Training* dan Validasi Dataset Histeq dan *Flipping* 87
Gambar 4.33 *Confusion* Matrix Dataset Histeq dan *Flipping* 87
Gambar 4.34 Grafik Akurasi *Training* Dataset Setelah *Preprocessing* dan Augmentasi 88
Gambar 4.35 Grafik *Loss Training* dan Validasi Dataset Setelah *Preprocessing* dan Augmentasi 89
Gambar 4.36 *Confusion Matrix* Dataset Setelah *Preprocessing* dan Augmentasi 90

DAFTAR TABEL

Tabel 2.1 Tabel Intensitas Citra Pada Gambar 2.7 19
Tabel 2.2 Tabel Hasil Transformasi Intensitas Citra Pada Gambar 2.7 19
Tabel 3.1 Contoh *Confusion Matrix Multiclass* 51
Tabel 3.2 Rincian Dataset RD Untuk Analisis Pengaruh *Preprocessing* dan Augmentasi Data 54
Tabel 4.1 Hasil *Training* dan *Testing* Model 58
Tabel 4.2 Hasil Uji Shapiro-Wilk Pengujian Model 69
Tabel 4.3 Hasil Uji Mann-Whitney U Pengujian Model 73
Tabel 4.4 Akurasi Tes Analisis *Preprocessing* dan Augmentasi Data 73
Tabel 4.5 Hasil Uji Shapiro-Wilk Pengujian Model Analisis *Preprocessing* dan Augmentasi Data 93
Tabel 4.6 Hasil Uji Mann-Whitney U Pengujian Model Analisis *Preprocessing* dan Augmentasi Data 97

BAB I
PENDAHULUAN

1.1 Latar Belakang

Retinopati Diabetik adalah komplikasi dari penyakit diabetes melitus. Retinopati Diabetik merupakan salah satu penyebab utama kebutaan pada penduduk usia kerja. Retinopati Diabetik terbagi menjadi dua tingkat, yaitu *non-proliferative diabetic retinopathy* (NPRD) dan *proliferative diabetic retinopathy* (PRD) (Wang & Lo, 2018). NPRD terbagi menjadi 3 tahap, yaitu NPRD ringan (*mild*), NPRD sedang (*moderate*), dan NPRD berat (*severe*) (Fajariyanti, 2017). Dalam penelitian yang dilakukan di beberapa daerah di Indonesia, prevalensi retinopati diabetik diperkirakan sebesar 42,6%, dari jumlah tersebut, sekitar 10% mengalami kebutaan (Bayer, 2019).

Untuk mencegah kebutaan, diperlukan metode deteksi dini agar penyakit retinopati diabetik dapat segera diobati. Namun, cara mengklasifikasi penyakit retinopati diabetik saat ini yang biasanya dilakukan adalah secara manual. Diagnosis manual membutuhkan waktu yang lama dan memerlukan tenaga ahli (Qummar et al., 2019). Selain itu, cara ini dapat menimbulkan pendapat yang berbeda-beda dari masing-masing tenaga ahli (Valverde et al., 2016). Oleh karena itu, dibutuhkan sebuah metode deteksi penyakit retinopati diabetik untuk menghindari kemungkinan diagnosis yang salah.

Metode deteksi yang dapat dikembangkan menggunakan metode *deep learning*. Pengembangan metode tersebut dapat meningkatkan akurasi deteksi retinopati diabetik dalam setiap tahapan. Hal ini disebabkan karena metode *deep learning* membuat sebuah model khusus untuk mengidentifikasi objek tersebut. Beberapa penelitian terdahulu terkait dengan deteksi citra menggunakan CNN yang menjadi landasan penggunaan CNN adalah, penelitian Lin dan Wu di tahun 2023, digunakan arsitektur ResNet-50 untuk mengklasifikasi penyakit retinopati diabetik. Akurasi arsitektur ResNet-50 pada penelitian tersebut mencapai 75% (Lin & Wu, 2023). Pada tahun 2020, Shankar dkk. berhasil menerapkan *Hyper Parameter Tuning Inception-v4* dan *Contrast Limited Adaptive Histogram Equalization* untuk mengklasifikasi penyakit retinopati diabetik dengan nilai *f1-score* 98,72% (Shankar

et al., 2020). Pada penelitian Sudha dan Ganeshbabu (2021), *preprocessing* citra digunakan untuk membedakan abnormalitas pada citra fundus penderita retinopati diabetik. Metode tersebut diikuti dengan penggunaan arsitektur VGG-19 terbukti dapat memberikan tingkat akurasi hingga 96% (Sudha & Ganeshbabu, 2021). Beberapa penelitian terdahulu terkait dengan optimasi model *deep learning* menggunakan *preprocessing* citra dan augmentasi data yang menjadi landasan penggunaan *preprocessing* citra dan augmentasi data adalah, penelitian Mishra dkk., (2020) menggunakan *cropping* untuk mengoptimasi performa model *deep learning* pada kasus pengawasan bencana banjir. Proses *cropping* meningkatkan akurasi *training* dari 55% menjadi 76% (Mishra et al., 2020). Penelitian yang dilakukan Yang dkk., (2022) menerapkan berbagai macam augmentasi data pada berbagai macam model dan dataset. Pada penelitian tersebut disimpulkan bahwa augmentasi data merupakan solusi efektif meningkatkan akurasi model (Yang et al., 2022).

Oleh karena itu, penelitian ini akan menerapkan *preprocessing* citra dan augmentasi data sebagai upaya meningkatkan akurasi dari empat model CNN, yaitu Inception-v4, Resnet-50, VGG-19, dan YOLO v5Nano. *Preprocessing* citra yang akan digunakan pada penelitian ini antara lain *cropping* dan *histogram equalization*. Augmentasi data yang akan digunakan adalah transformasi *flipping.*

1.2 Identifikasi Masalah

Berdasarkan latar belakang masalah yang telah dipaparkan, dapat diidentifikasi masalah berupa proses klasifikasi penyakit RD secara manual membutuhkan waktu yang lama dan hasil klasifikasi bisa berbeda. Hingga saat ini, metode diagnosis menggunakan *deep learning* yang sudah diciptakan hanya dapat mengidentifikasi penyakit RD. Sehingga belum dapat melakukan diagnosis tingkat keparahan dari penyakit RD tersebut.

Penyakit RD Tingkat keparahan berbeda-beda, namun masih belum bisa mengklasifikasi Tingkat keparahan RD

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut.

a. Dataset yang digunakan diambil dari *website* Kaggle (Dugas et al., 2015).
b. Program dibuat menggunakan Visual Studio Code dan *Google Colab*.
c. Arsitektur *deep learning* yang digunakan adalah Inception-v4, ResNet-50, VGG-19, dan YOLO V5 Nano.
d. *Preprocessing* citra yang digunakan adalah *cropping* dan *histogram equalization*.

1.4 Rumusan Masalah

Berdasarkan identifikasi masalah yang dipaparkan, dapat dirumuskan masalah dari penelitian ini sebagai berikut.

a. Apa arsitektur CNN yang optimal untuk mengklasifikasi penyakit retinopati diabetik?
b. Apakah *preprocessing* citra dan augmentasi data mampu meningkatkan akurasi dari model *deep learning* untuk mengklasifikasi penyakit retinopati diabetik?

1.5 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut.

a. Mengetahui arsitektur CNN yang optimal untuk mengklasifikasi penyakit retinopati diabetik.
b. Mengetahui kemampuan *preprocessing* citra dan augmentasi data dalam meningkatkan akurasi dari model *deep learning* untuk mengklasifikasi penyakit retinopati diabetik.

1.6 Manfaat Penelitian

Manfaat dari pengerjaan penelitian ini sebagai berikut.

a. Bagi penulis, dapat menambah wawasan dan keterampilan baru dalam membuat model *Deep Learning* yang diperuntukkan untuk mengklasifikasi penyakit retinopati diabetik.
b. Bagi Universitas khususnya Program Studi Teknik Informatika, dapat membantu mempersiapkan lulusan yang siap kerja dan kompeten dengan memberikan bekal kepada mahasiswa berupa proses pembelajaran yang intens selama kegiatan penelitian Tugas Akhir.
c. Bagi Masyarakat khususnya ahli medis, model *Deep Learning* yang dihasilkan pada penelitian Tugas Akhir dapat dimanfaatkan dan dipergunakan untuk melakukan klasifikasi penyakit retinopati diabetik.

1.7 Luaran Penelitian

Luaran hasil penelitian ini berupa model yang optimal untuk mengklasifikasi RD melalui citra fundus dan artikel ilmiah yang akan dipublikasikan di jurnal.

1.8 Sistematika Penulisan

Sistematika dalam penulisan proposal Tugas Akhir ini akan dibagi menjadi lima bab seperti berikut.

Bab I Pendahuluan

Pada bab pendahuluan ini berisi latar belakang, identifikasi masalah, batasan masalah, tujuan penelitian, manfaat penelitian, luaran tugas akhir dan sistematika penulisan.

Bab II Tinjauan Pustaka

Bab tinjauan pustaka ini berisi uraian sistematis terkait dengan literatur yang digunakan dalam proses penyusunan Tugas Akhir sehingga diperoleh landasan teori terkait *deep learning*, *preprocessing* citra, augmentasi data, dan penyakit retinopati diabetik.

Bab III Metodologi Penelitian

Bab ini menjelaskan tahapan pengerjaan serta analisis perancangan awal sistem yang akan dibuat. Tahapan ini sendiri terdiri atas identifikasi masalah, studi pustaka, pengumpulan data, desain sistem dan pengujian.

Bab IV Hasil dan Pembahasan

Bab ini menyajikan hasil-hasil dari tahapan pengerjaan serta analisis sistem yang telah dilakukan. Dalam bab ini, terdapat penjelasan mengenai identifikasi masalah, studi pustaka, pengumpulan data, *profiling*, desain sistem, serta pengembangan dan pengujian aplikasi.

Bab V Kesimpulan dan Saran

Bab ini merupakan rangkuman simpulan dari hasil penelitian Tugas Akhir yang telah dilakukan, serta menyajikan saran-saran yang dapat diterapkan untuk meningkatkan sistem aplikasi dalam penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Diabetes Melitus

Diabetes melitus adalah sebuah penyakit metabolik yang ditandai dengan kadar gula dalam darah yang tinggi. Penderita diabetes melitus tidak dapat memproduksi insulin yang cukup atau sel-sel tubuh penderita menjadi tidak responsif kepada insulin. Insulin diproduksi di dalam sel-sel pankreas dan berfungsi untuk mengatur kadar gula dalam tubuh dan memfasilitasi masuknya glukosa ke dalam sel-sel tubuh untuk digunakan sebagai sumber energi. Terdapat tiga jenis diabetes melitus, yaitu:

1. Diabetes Tipe 1

   Diabetes tipe 1 disebabkan karena sel-sel pankreas tidak dapat memproduksi insulin. Hal ini dapat disebabkan karena gangguan autoimun yang menyebabkan sistem kekebalan tubuh menghancurkan sel-sel penghasil insulin di pankreas.

2. Diabetes Tipe 2

   Diabetes tipe 2 merupakan jenis diabetes yang paling umum terjadi. Diabetes tipe ini dapat terjadi ketika tubuh menjadi kurang responsif terhadap insulin atau tidak memproduksi insulin dengan cukup. Hal tersebut dapat dipengaruhi oleh pola makan yang buruk, berat badan berlebih, kurangnya aktivitas fisik, dan faktor genetik.

3. Diabetes Gestasional

   Diabetes gestasional terjadi pada wanita hamil yang sebelumnya tidak terkena penyakit diabetes tipe apapun. Hal ini terjadi karena meningkatnya resistensi tubuh terhadap insulin atau kurangnya produksi insulin pada masa kehamilan.

Penderita diabetes melitus perlu menjaga kadar glukosa dalam tubuh agar tidak terjadi komplikasi. Komplikasi jangka Panjang dari penyakit diabetes melitus adalah retinopati diabetik (mata), nefropati diabetik(ginjal), neuropati diabetik (saraf), dan penyakit jantung. Cara menjaga kadar glukosa dalam tubuh dapat dilakukan dengan menerapkan pola makan yang sehat, olahraga teratur, tidak stress,

dan mengonsumsi obat-obatan antidiabetes. Apabila diperlukan, dapat menggunakan suntikan insulin (International Diabetes Federation & The Fred Hollows Foundation, 2015).

2.2 Retinopati Diabetik

Retinopati diabetik adalah salah satu komplikasi jangka panjang dari penyakit diabetes melitus. Kondisi ini disebabkan oleh rusaknya pembuluh darah pada retina. Pada awalnya, penderita penyakit ini bisa tidak merasakan gejala apapun. Namun, seiring waktu penglihatan akan menjadi kabur dan dapat menyebabkan kebutaan jika tidak segera diobati dengan benar. Retinopati diabetik dapat menyebabkan perubahan-perubahan berikut pada mata penderita (International Diabetes Federation & The Fred Hollows Foundation, 2015).

- Mikroaneurisma, yaitu pembengkakan kecil pada pembuluh darah yang berada di mata. Hal ini dapat menyebabkan bocornya cairan ke dalam retina.
- Pendarahan pada retina, disebabkan oleh bocornya darah dari pembuluh darah ke dalam retina.
- Eksudat keras, yaitu lemak yang tersimpan di dalam mata yang menjadi keras.
- *Cotton wool spots*, yaitu bercak di mata yang disebabkan oleh pembengkakan akson iskemik di lapisan saraf.
- Pelebaran dan perlekatan pembuluh darah vena
- *Intraretinal microvascular abnormalities*, percabangan atau pelebaran abnormal pembuluh darah dalam retina yang sudah ada.
- Munculnya pembuluh darah baru pada tempat yang abnormal, bergantung pada lokasi munculnya bisa disebut *neovascularisation of the disc* atau *neovascularisation elsewhere*.

Penyakit retinopati diabetik terbagi menjadi 2 tahap, yaitu *non-proliferative diabetic retinopathy* dan *proliferative diabetic retinopathy*. *Non-proliferative diabetic retinopathy* terbagi menjadi 3, yaitu NPRD ringan (*mild*), NPRD sedang (*moderate*), dan NPRD berat (*severe*). Pada tahap NPRD ringan, gejala yang

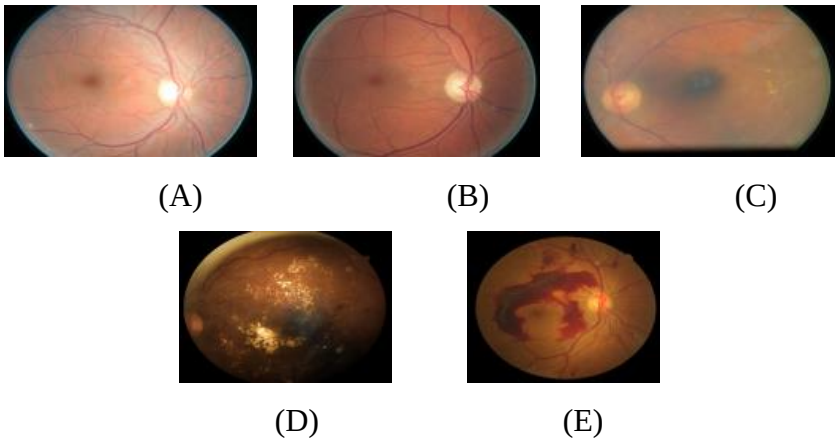
dialami terdapat minimal satu mikroaneurisma. Pada tahap NPRD sedang, terdapat gejala mikroaneurisma yang meluas, pendarahan di dalam retina, dan *cotton wool spots*. Pada tahap NPRD berat, terdapat *intraretinal microvascular abnormalities*. Pada tahap NPRD ini dapat memengaruhi fungsi visual dari mata penderita.

Proliferative diabetic retinopathy merupakan komplikasi paling parah dari retinopati diabetik. Pada tahap ini, terdapat gejala *neovascularisation*. Pembuluh darah baru yang terbentuk dari *neovascularisation* rentan bocor. Jika tidak diatasi, hal tersebut dapat menyebabkan kebutaan (Kementerian Kesehatan Republik Indonesia, 2018).

Salah satu cara untuk mengklasifikasi penyakit retinopati diabetik adalah menggunakan pemeriksaan citra fundus. Citra fundus hanya dapat diambil menggunakan kamera khusus yaitu kamera fundus. Kamera fundus dapat mengambil citra mata dengan sangat detail. Pemeriksaan ini biasa digunakan untuk penderita penyakit retinopati diabetik, glaukoma, degenerasi makula, dan lain-lain. Pemeriksaan citra fundus mempermudah dokter untuk melihat kondisi dari mata penderita, contohnya adalah seperti berikut:

a. Ada atau hilangnya pembuluh darah sekitar retina.
b. Ada atau tidaknya pendarahan di sekitar retina.
c. Ada atau tidaknya pembuluh darah yang tidak seharusnya ada.

Berikut adalah contoh dari citra fundus untuk masing-masing tahapan penyakit retinopati diabetik.



Gambar 2.1 Citra Fundus, (A) Mata Normal, (B) Mata NPRD Ringan, (C) Mata NPRD Sedang, (D) Mata NPRD Berat, (E) Mata PRD

(Qummar et al., 2019)

Gambar 2.1 (A) merupakan hasil pengambilan citra fundus mata pada pasien yang tidak terkena komplikasi retinopati diabetik. Pada gambar 2.1 (A) dapat dilihat bahwa belum muncul gejala-gejala awal retinopati diabetik. Gambar 2.1 (B) merupakan hasil pengambilan citra fundus mata pada pasien yang baru terkena NPRD ringan. Pada gambar 2.1 (B) dapat dilihat mata pasien sudah mulai muncul gejala-gejala retinopati diabetik seperti mikroaneurisma. Pada gambar 2.1 (C) merupakan hasil pengambilan citra fundus mata pada pasien yang terkena NPRD sedang. Pada gambar 2.1 (C) dapat dilihat gejala-gejala pada mata pasien sudah bertambah parah. Pada gambar 2.1 (C), *cotton wool spots* pada mata pasien semakin jelas terlihat.

Gambar 2.1 (D) merupakan hasil pengambilan citra fundus mata pada pasien yang terkena NPRD berat. Pada gambar 2.1 (D) terlihat *cotton wool spots* semakin bertambah parah, pembuluh darah juga semakin menebal dan muncul pada tempat yang tidak seharusnya. Gambar 2.1 (E) merupakan hasil pengambilan citra fundus mata pada pasien yang terkena PRD. Pada gambar 2.1 (E) dapat terlihat mata pasien sudah mengalami pendarahan akibat bocornya pembululuh darah. Pembuluh darah pada mata pasien juga terlihat lebih banyak dibandingkan tahap-tahap retinopati diabetik sebelumnya.

2.3 Citra

Citra digital adalah representasi visual dari sebuah informasi yang disimpan dalam format digital, terdiri dari titik-titik yang biasa disebut dengan “piksel”. Setiap piksel adalah unit dasar yang digunakan untuk membuat gambar digital, dan setiap piksel adalah warna yang terdiri dari tiga saluran warna (merah, hijau, dan biru) yang dapat digabungkan untuk menghasilkan berbagai warna. Citra digital dapat dibuat dengan berbagai cara, seperti mengambil foto dengan kamera digital, memindai dokumen atau gambar menggunakan mesin pemindai, atau menggambar atau menggunakan perangkat lunak grafik untuk membuat gambar dari awal.

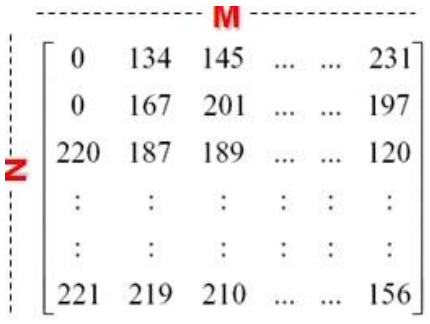
Setelah dibuat, gambar digital dapat disimpan dalam berbagai format file, seperti JPEG, PNG, atau GIF. Citra digital memiliki beberapa keunggulan dibandingkan gambar analog (seperti foto pada film atau kertas). Misalnya, gambar digital dapat dengan mudah disalin, dikirim, dan diedit menggunakan komputer.

Selain itu, gambar digital juga dapat dengan mudah dicetak kembali dengan kualitas yang sama seperti aslinya. Namun citra digital juga memiliki kekurangan, seperti kerusakan akibat proses kompresi atau penurunan kualitas jika terus menerus disimpan dan dimodifikasi.

Citra digital memiliki banyak aplikasi dalam berbagai bidang, seperti fotografi, desain grafis, ilmu komputer, dan lain-lain. Teknologi citra digital juga terus berkembang dan memungkinkan penggunaan citra digital dalam berbagai aplikasi yang semakin luas, seperti pengenalan wajah, pemrosesan citra satelit, dan lainnya. Oleh karena itu, pengetahuan tentang citra digital sangat penting bagi para profesional di bidang terkait (Pramunendar et al., 2020).

2.4 Intensitas Citra

Citra digital pada umumnya memiliki bentuk persegi panjang dengan dimensi tinggi x lebar (N x M). N menyatakan jumlah baris sedangkan M menyatakan jumlah kolom pada matriks citra. Masing-masing elemen pada baris dan kolom tersebut dapat disebut sebagai elemen citra, elemen gambar, atau piksel. Informasi dalam piksel bergantung pada tipe data yang digunakan. Nilai piksel selalu merupakan bilangan biner dengan panjang k, sehingga piksel dapat mewakili nilai beda 2^k. Nilai k disebut juga sebagai kedalam bit (*bit depth*) dari citra. Susunan skala bit yang tepat dari sebuah piksel bergantung pada jenis citra seperti citra biner, citra aras keabuan (*greyscale*), atau warna RGB (*red green blue*).

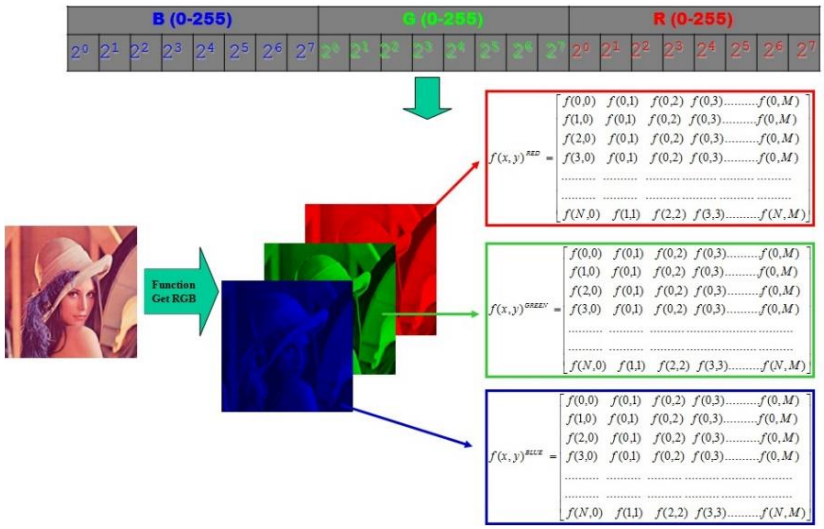


Gambar 2.2 Contoh Intensitas Citra *greyscale* pada matriks 2D

(Sumber: https://yusronrijal.wordpress.com/2012/03/24/pengolahan-citra-digital/)

Gambar 2.2 merupakan contoh intensitas citra *greyscale* pada matriks 2D. Data citra dalam citra *greyscale* terdiri dari kanal tunggal yang mewakili intensitas,

kecerahan atau densitas citra. Nilainya berupa bilangan bulat positif antara 0 hingga $2^k - 1$. Citra *greyscale* memiliki nilai kedalaman bit delapan, maka nilai intensitas pikselnya bernilai dari 0 hingga 255. Nilai 0 mewakili hitam (kecerahan minimum) dan 255 mewakili warna putih (kecerahan maksimum).



Gambar 2.3 Contoh Intensitas Citra RGB pada Matriks 2D

(Sumber: https://yusronrijal.wordpress.com/2012/03/24/pengolahan-citra-digital/)

Gambar 2.3 memiliki tiga kanal yang dimana masing-masing kanal memiliki kedalaman 8 bit. Oleh karena itu, masing-masing piksel dalam citra RGB dikodekan dalam 24 bit. Sedangkan dalam citra biner, masing-masing piksel hanya memiliki kedalaman 1 bit. Sehingga masing-masing piksel dalam citra biner hanya dapat bernilai 0 (berwarna hitam) atau 1 (berwarna putih) (Sulistiyanti et al., 2016).

2.5 *Preprocessing* Citra

Preprocessing citra adalah langkah-langkah atau teknik yang digunakan untuk mempersiapkan data citra sebelum dilakukan analisis atau pengolahan lebih lanjut kepada data citra tersebut. Tujuan dari *preprocessing* citra adalah meningkatkan kualitas citra, menghilangkan gangguan atau *noise*, meningkatkan kontras, mengurangi dimensi data, dan memperjelas fitur penting dalam citra. Berikut adalah macam-macam teknik yang umum digunakan dalam *preprocessing* citra:

1. *Cropping* atau pemotongan

   Memotong bagian yang tidak diperlukan dari citra, bertujuan untuk memfokuskan perhatian kepada area yang diinginkan.

2. Normalisasi

   Mengubah rentang nilai piksel dalam citra agar sesuai dengan rentang piksel yang diinginkan, bertujuan untuk menghilangkan perbedaan skala yang ada di antara citra-citra yang berbeda-beda.

3. *Contrast Adjustment*

   Meningkatkan perbedaan intensitas antara piksel-piksel dalam citra, dapat dilakukan dengan mengaplikasikan teknik transformasi histogram, seperti ekualisasi histogram.

4. *Noise Reduction*

   Mengurangi atau menghilangkan *noise* yang mungkin terdapat dalam citra. Salah satu teknik yang biasa digunakan adalah penghalusan citra (*smoothing*) dengan *filter* median atau *filter* Gaussian.

5. Pemisahan Warna

   Jika citra berwarna, citra dapat dipisahkan menjadi saluran warna terpisah, seperti merah, hijau, dan biru. Hal ini dapat memungkinkan analisis yang lebih spesifik pada masing-masing saluran warna.

6. *Resize* atau *Resampling*

   Mengubah ukuran citra secara proporsional dengan memperbesar atau memperkecil resolusi citra. Hal ini sering dilakukan untuk menyesuaikan ukuran citra agar sesuai dengan kebutuhan analisis atau mempercepat proses pengolahan citra.

7. Segmentasi

   Memisahkan citra menjadi wilayah-wilayah yang berbeda berdasarkan perbedaan intensitas atau fitur-fitur lainnya. Segmentasi dapat membantu dalam mengidentifikasi objek atau fitur yang penting dalam sebuah citra.

8. *Brightness Adjustment*

   Mengubah kecerahan citra menjadi lebih terang atau lebih gelap bergantung pada kebutuhan analisis citra tersebut. Biasa digunakan

untuk menyesuaikan kecerahan citra untuk membantu mengidentifikasi fitur yang penting dalam sebuah citra.

Preprocessing citra merupakan langkah penting dalam analisis citra dan sering kali menjadi langkah awal sebelum melakukan ekstraksi fitur atau pengenalan objek dalam citra. Teknik yang digunakan dalam *preprocessing* citra dapat bervariasi bergantung pada jenis citra yang digunakan dan tujuan analisis yang ingin dicapai (Sulistiyanti et al., 2016).

2.5.1 *Cropping*

Cropping adalah sebuah teknik pemrosesan citra dengan cara menghapus bagian-bagian yang tidak diinginkan dari sebuah citra. Teknik ini dilakukan untuk memperoleh elemen yang diinginkan dari citra tersebut dan dapat fokus pada elemen tersebut. Biasanya, teknik *cropping* ini dilakukan secara manual melalui perangkat lunak pengolah citra seperti Adobe Photoshop, Corel Draw, dan lain-lain. Dalam proses *cropping*, citra dapat dipotong-potong untuk memperbaiki komposisi, menyesuaikan ukuran, mengubah rasio aspek, atau menghilangkan unsur-unsur yang tidak relevan. Berikut adalah contoh citra yang diterapkan proses *cropping*:

(A) (B)

Gambar 2.4 Contoh Citra (A) Sebelum Diterapkan *Cropping*, (B) Setelah Diterapkan *Cropping*

(Sumber: *online pictures* di aplikasi Microsoft Word)

2.5.2 *Contrast Adjustment*

Contrast adjustment adalah proses mengubah tingkat kontras dalam sebuah citra untuk meningkatkan perbedaan antara piksel-piksel yang berdekatan. Kontras menggambarkan perbadaan kecerahan antara bagian gelap dengan bagian cerah dalam sebuah citra. Dalam citra dengan kontras yang rendah, perbedaan antara bagian gelap dengan bagian cerah menjadi kurang jelas. Dalam citra dengan kontras yang tinggi, perbedaan ini menjadi semakin jelas.

Terdapat berbagai metode dan teknik yang dapat digunakan untuk menyesuaikan kontras dalam citra, termasuk:

1. *Brightness/Contrast Adjustment*

   Brightness/Contrast Adjustment merupakan metode sederhana yang memungkinkan pengguna untuk mengatur tingkat kecerahan dan kontras secara langsung.
2. *Histogram Stretching*

   Histogram Stretching merupakan metode yang menggunakan histogram citra untuk mengubah rentang intensitas piksel. Dengan merentangkan histogram ke seluruh rentang intensitas yang tersedia, perbedaan kontras dalam gambar diperbesar.
3. *Histogram Equalization*

   Histogram Equalization merupakan metode yang menyetarakan nilai rentang intensitas piksel. Hal ini berguna untuk meratakan distribusi intensitas piksel citra dan meningkatkan kontras citra secara keseluruhan
4. *Tone Mapping*

   Tone Mapping biasa digunakan dalam pemrosesan citra *High Dynamic Range* (HDR) untuk mengubah tingkat kontras antara area terang dan area gelap dalam citra, sehingga detail dapat dipertahankan dengan baik.
5. Pengolahan Lokal

   Kontras sebuah citra dapat diatur berdasarkan konteks regional atau berdasarkan fitur-fitur tertentu dalam citra. Oleh karena itu, bagian citra yang berubah hanya bagian yang diinginkan saja dan tidak seluruh citra.

Contrast adjustment biasa dilakukan di perangkat lunak pengedit citra seperti Adobe Photoshop, Lightroom, atau GIMP. Pengaturan kontras yang tepat akan bergantung pada preferensi pengguna dan sifat citra yang sedang diproses. Berikut adalah contoh citra yang diterapkan *contrast adjustment*:

(A) (B)

Gambar 2.5 Contoh Citra (A) Sebelum Diterapkan *Contrast Adjustment*, (B) Setelah Diterapkan *Contrast Adjustment*

(Sumber: https://www.mathworks.com/help/images/contrast-enhancement-techniques.html)

Untuk melakukan *contrast adjustment*, perlu menghitung faktor perubahan *contrast* yang diinginkan. Berikut adalah persamaan operasi *contrast adjustment*:

$$F = \frac{259 * (255 + c)}{255 * (259 - c)} \quad (2\text{-}1)$$

F = Faktor *contrast adjustment*

c = nilai *input* kontras

Kemudian langkah selanjutnya adalah menghitung nilai intensitas piksel terbaru.

$$f(x,y)' = F(f(x,y) - 128) + 128 \quad (2\text{-}2)$$

F = Faktor *contrast adjustment*

$f(x,y)$ = nilai intensitas piksel pada koordinat x, y

$f(x,y)'$ = nilai intensitas piksel baru pada koordinat x, y

Nilai intensitas piksel tidak boleh melebihi batas intensitas piksel tersebut. Oleh karena itu diperlukan persamaan untuk membatasi nilai intensitas piksel tersebut.

$$f(x,y)\begin{cases} 255, f(x,y) > 255 \\ (x,y), 0 < f(x,y) < 255 \\ 0, f(x,y) < 0 \end{cases} \quad (2\text{-}3)$$

Dengan menggunakan persamaan (2-3), nilai intensitas piksel tersebut tidak akan melebihi batas atas maupun bawah intensitas piksel tersebut (Sulistiyanti et al., 2016).

2.5.3 *Histogram Equalization*

Histogram Equalization adalah salah satu teknik *preprocessing* citra yang digunakan untuk meningkatkan kontras citra dengan cara mendistribusikan ulang intensitas piksel secara merata di seluruh rentang nilai intensitas yang tersedia. Pada dasarnya, teknik *histogram equalization* memanfaatkan histogram dari sebuah citra. Histogram sendiri adalah grafik distribusi frekuensi kemunculan nilai intensitas piksel sebuah citra.

Proses *histogram equalization* terdiri dari beberapa langkah, yaitu:

1. Menghitung histogram

   Histogram citra awal dihitung untuk memperoleh informasi mengenai distribusi intensitas piksel citra tersebut.
2. Menghitung *Cumulative Distribution Function* (CDF)

   Fungsi ini merupakan akumulasi dari histogram dan berfungsi untuk menunjukkan jumlah piksel dengan intensitas yang kurang dari atau sama dengan nilai intensitas tertentu.
3. Normalisasi CDF

   Fungsi CDF dinormalisasikan agar rentang nilai dari fungsi CDF adalah 0 hingga 1.
4. Transformasi intensitas

Setiap piksel dalam citra awal ditransformasikan dengan mengganti nilai intensitasnya dengan nilai intensitas baru yang dihasilkan dari fungsi CDF yang dinormalisasi.

5. Pembuatan histogram baru

   Setelah piksel di citra awal ditransformasi, histogram citra baru akan dihitung untuk memastikan distribusi intensitas di citra baru sudah didistribusikan secara merata di seluruh rentang nilai intensitas yang tersedia.

Hasil dari *histogram equalization* adalah sebuah citra dengan kontras yang lebih baik, di mana piksel intensitas rendah dan tinggi didistribusikan secara merata di seluruh rentang nilai intensitas. Teknik *histogram equalization* akan memperjelas detail dan memperbaiki visualisasi citra dengan intensitas yang tidak seimbang sebelumnya. *Histogram equalization* biasa digunakan dalam aplikasi pengolahan citra seperti perbaikan gambar, deteksi tepi, segmentasi, dan aplikasi analisis citra lainnya. Berikut adalah contoh citra yang diterapkan *histogram equalization*:

(A) (B)

Gambar 2.6 Contoh Citra (A) Sebelum Diterapkan *Histogram Equalization*, (B) Setelah Diterapkan *Histogram Equalization*

(Sumber: https://www.mathworks.com/help/images/contrast-enhancement-techniques.html)

Histogram equalization dilakukan dengan cara meratakan nilai perseberan dari *histogram* sebuah citra. Berikut adalah persamaan matematika dari *histogram equalization*:

$$p_r(r_k) = \frac{n_k}{k} \qquad (2\text{-}4)$$

dimana

$$r_k = \frac{k}{L-1}, 0 \leq k < L-1 \quad (2\text{-}5)$$

k = nilai kedalaman bit pada sebuah piksel

p_r = peluang kemunculan intensitas piksel bernilai r

r_k = nilai intensitas piksel yang sudah dinormalkan

L = nilai kedalaman bit pada sebuah citra

n_k = jumlah piksel dengan nilai kedalaman bit k

Histogram Equalization merupakan proses transformasi (T) intensitas sebuah piksel (r) menjadi intensitas yang baru (s). Persamaan transformasi ini dapat dituliskan menjadi persamaan berikut:

$$S_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j) \quad (2\text{-}6)$$

Persamaan (2-6) merupakan persamaan yang akan digunakan untuk mencari nilai intensitas baru dari sebuah piksel (Sulistiyanti et al., 2016). Berikut adalah contoh *histogram equalization* pada sebuah citra dengan ukuran 10x10 dengan intensitas maksimal (L) 8:

0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
2	2	2	2	2	1	1	1	1	1
2	2	2	2	2	3	3	3	3	3
2	2	2	2	2	4	4	4	4	4
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	6	6	6	7	7

Gambar 2.7 Contoh Matriks 2D Nilai Intensitas Citra yang Berukuran 10x10

Tabel 2.1 Tabel Intensitas Citra Pada Gambar 2.7

k	r_k	n_k	$P_r(r_k)$	$s_k = P_r(r_j)$
0	0/7 = 0	20	0,2	0,2
1	1/7 = 0,14	25	0,25	0,45
2	2/7 = 0,29	15	0,15	0,6
3	3/7 = 0,43	5	0,05	0,65
4	4/7 = 0,57	15	0,25	0,8
5	5/7 = 0,71	15	0,25	0,95
6	6/7 = 0,86	3	0,03	0,98
7	7/7 = 1	2	0,02	1

Nilai dari s_k dibulatkan ke nilai r terdekat.

$s_0 =$ 0,2 lebih dekat ke nilai 1/7 = 0,14, maka $s_0 = \frac{1}{7}$

$s_1 =$ 0,45 lebih dekat ke nilai 3/7 = 0,43, maka $s_1 = \frac{3}{7}$

$s_2 =$ 0,60 lebih dekat ke nilai 4/7 = 0,57, maka $s_2 = \frac{4}{7}$

$s_3 =$ 0,65 lebih dekat ke nilai 5/7 = 0,71, maka $s_3 = \frac{5}{7}$

$s_4 =$ 0,80 lebih dekat ke nilai 6/7 = 0,86, maka $s_4 = \frac{6}{7}$

$s_5 =$ 0,95 lebih dekat ke nilai 7/7 = 1, maka $s_5 = \frac{7}{7}$

$s_6 =$ 0,98 lebih dekat ke nilai 7/7 = 1, maka $s_6 = \frac{7}{7}$

$s_7 =$ 1 lebih dekat ke nilai 7/7 = 1, maka $s_7 = \frac{7}{7}$

Tabel 2.2 Tabel Hasil Transformasi Intensitas Citra Pada Gambar 2.7

k	r_k	$s_k = P_r(r_j)$	**s**
0	0/7 = 0	0,2	1
1	1/7 = 0,14	0,45	3
2	2/7 = 0,29	0,6	4
3	3/7 = 0,43	0,65	5
4	4/7 = 0,57	0,8	6

5	5/7 = 0,71	0,95	7
6	6/7 = 0,86	0,98	7
7	7/7 = 1	1	7

Berikut adalah matriks 2D nilai intensitas citra yang telah diterapkan *histogram equalization*:

1	1	1	1	1	3	3	3	3	3
1	1	1	1	1	3	3	3	3	3
1	1	1	1	1	3	3	3	3	3
1	1	1	1	1	3	3	3	3	3
4	4	4	4	4	3	3	3	3	3
4	4	4	4	4	5	5	5	5	5
4	4	4	4	4	6	6	6	6	6
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7	7

Gambar 2.8 Matriks 2D Intensitas Citra Hasil *Histogram Equalization* dari Gambar 2.7

Gambar 2.8 merupakan matriks 2D dari intensitas gambar 2.7 yang telah diterapkan *histogram equalization*. Hasil dari *histogram equalization* di atas tidak terlalu merata disebabkan oleh nilai intensitas dan jumlah piksel yang terbatas. Hal ini juga disebabkan oleh hasil perataan adalah pembuatan ke intensitas yang terdekat.

2.5.4 *Brightness Adjustment*

Brightness adjustment adalah proses mengubah tingkat kecerahan dalam citra untuk mengatur kecerahan piksel-piksel dalam citra tersebut. Penyesuaian kecerahan memengaruhi sejauh mana citra terlihat terang atau gelap secara keseluruhan. Dalam proses *brightness adjustment*, kecerahan gambar dapat ditingkatkan atau dikurangi dengan cara mengubah nilai intensitas piksel secara proporsional. Penyesuaian kecerahan ini dapat dilakukan menggunakan perangkat lunak pengolahan citra seperti Adobe Photoshop, Lightroom, GIMP, atau alat pengolahan citra lainnya. Dalam beberapa aplikasi, *brightness adjustment* sering digunakan bersamaan dengan *contrast adjustment* untuk mengoptimalkan penampilan visual secara keseluruhan. *Brightness adjustment* dapat membantu meningkatkan detail dalam area citra yang terlalu gelap atau terlalu terang, atau

untuk mencapai pencahayaan yang lebih baik secara umum. Berikut adalah beberapa metode dan alat yang umum digunakan untuk *brightness adjustment*:

1. *Brightness Slider*

   Alat ini memungkinkan pengguna untuk secara langsung menyesuaikan tingkat kecerahan dengan cara menggeser *slider* ke kanan atau ke kiri. Geser ke kanan untuk meningkatkan kecerahan dan geser ke kiri untuk mengurangi kecerahan.

2. *Levels Adjustment*

   Pengaturan ini memungkinkan pengguna untuk secara manual menyesuaikan tingkat kecerahan dalam berbagai rentang intensitas piksel. Pengguna dapat mengatur titik hitam, putih, dan titik abu-abu tengah untuk mencapai pencahayaan yang diinginkan.

3. *Curves Adjustment*

   Pengaturan ini memungkinkan pengguna untuk mengubah distribusi kecerahan dalam gambar dengan lebih rinci. Pengguna dapat menyesuaikan kurva luminositas untuk mengatur kecerahan pada berbagai tingkat intensitas.

4. *Histogram Adjustment*

   Pengaturan ini menggunakan *histogram* citra, pengguna dapat melihat sebaran intensitas piksel dalam gambar dan menyesuaikan tingkat kecerahan berdasarkan informasi *histogram* tersebut.

Brightness adjustment harus dilakukan dengan hati-hati, agar penyesuaian yang dilakukan tidak berlebihan dan tidak menyebabkan gambar terlihat terlalu terang atau terlalu gelap dan kehilangan detailnya.

(A) (B)

Gambar 2.9 Contoh Citra (A) Sebelum Diterapkan *Brightness Adjustment*, (B) Setelah Diterapkan *Brightness Adjustment*

(Sumber: https://www.mathworks.com/help/images/low-light-image-enhancement.html)

Brightness adjustment dapat dilakukan dengan cara menambah atau mengurangkan sebuah konstanta ke intensitas masing-masing piksel di dalam citra. Berikut adalah persamaan matematika dari *brightness adjustment*:

$$f(x,y)' = f(x,y) + c \tag{2-7}$$

$f(x,y)$ = nilai intensitas piksel pada koordinat x, y

$f(x,y)'$ = nilai intensitas piksel pada koordinat x, y

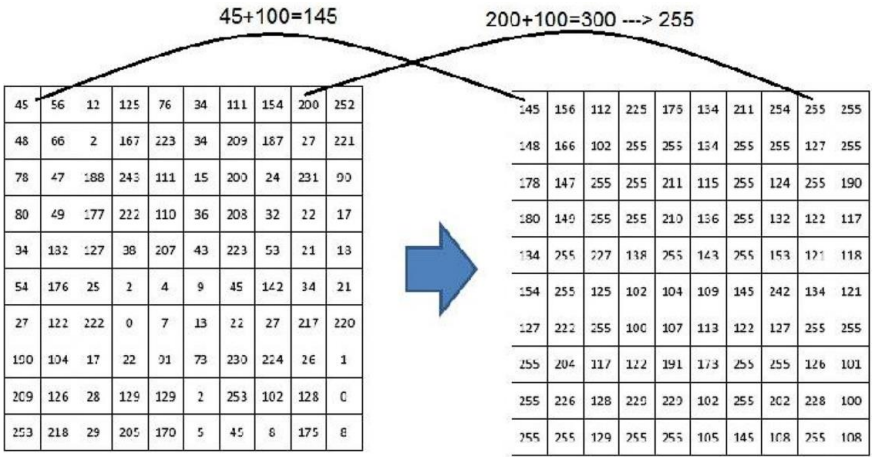
c = konstanta

Dalam persamaan (2-7) dimungkinkan untuk hasil dari persamaan tersebut melebihi batas dari intensitas piksel. Oleh karena itu diperlukan operasi pemotongan (*clipping*) agar hasil dari persamaan (2-7) tidak melebihi batas nilai intentsitas piksel. Berikut adalah persamaan operasi pemotongan:

$$f(x,y)\begin{cases} 2^k - 1, f(x,y) > 2^k - 1 \\ (x,y), 0 < f(x,y) < 2^k - 1 \\ 0, f(x,y) < 0 \end{cases} \tag{2-8}$$

k = nilai kedalaman bit

Berikut adalah contoh *brightness adjustment* pada citra:



Gambar 2.10 Contoh *Brightness Adjustment* Pada Citra Berbentuk Matriks 2D (Sulistiyanti et al., 2016)

Gambar 2.10 merupakan contoh *brightness adjustment* pada citra yang berbentuk matriks 2D. Pada piksel (0,0) nilai intensitas citra tersebut adalah 45, setelah diterapkan *brightness adjustment* (+100) nilai intensitas piksel tersebut berubah menjadi 145. Begitu juga dengan nilai piksel (8,0) memiliki nilai intensitas 200, setelah diterapkan *brightness adjustment* (+100) nilai intensitas piksel tersebut berubah menjadi 255. Hal itu disebabkan karena nilai intensitas citra memiliki batas atas 255, sehingga nilai intensitas piksel (8,0) tidak dapat melebihi nilai 255 (Sulistiyanti et al., 2016).

2.6 Augmentasi Data

Augmentasi data atau *data augmentation* merupakan cara untuk mengurangi *over-fitting*. Augmentasi data akan menghasilkan data baru dengan cara melakukan transformasi pada data asli. Augmentasi data memungkinkan untuk meningkatkan generalisasi data. Berikut merupakan beberapa alasan utama untuk menggunakan augmentasi data:

1. Data terlalu sedikit

   Ketika data terlalu sedikit dan ingin membuat model *machine learning* yang kompleks, maka model akan cenderung memiliki akurasi yang rendah. Oleh karena itu, augmentasi data akan digunakan untuk menambah jumlah data agar bisa digunakan dalam model *machine*

learning tersebut. Dengan melakukan augmentasi data, jumlah data akan menjadi semakin banyak dan meningkatkan akurasi dari model tersebut.

2. Meningkatkan akurasi model

   Ketika data yang digunakan sudah cukup banyak, namun akurasi dari model kurang memuaskan, maka dapat melakukan augmentasi data untuk meningkatkan akurasi model tersebut. Augmentasi data dapat mencegah model mengalami *over-fitting* atau *under-fitting*.

Pada data citra, *tools* yang biasa digunakan adalah *OpenCV* (*python library*), *Pillow* (*python library*), dan *imgaug* (*python library*). Augmentasi data yang dilakukan pada data citra adalah sebagai berikut (Mumuni & Mumuni, 2022).

1. Transformasi geometri

   Memutar, memotong, membalik, meregangkan, dan memperbesar citra secara acak.

2. Transformasi spasi warna

   Mengubah saluran warna RGB, kontras, dan kecerahan citra secara acak.

3. Penghapusan acak

   Menghapus citra secara acak.

4. Pencampuran gambar

   Mencampurkan beberapa citra secara acak.

2.7 *Artificial Intelligence*

Artificial Intelligence (AI) atau dalam bahasa Indonesia disebut kecerdasan buatan merupakan simulasi kecerdasan manusia pada mesin yang diprogram untuk melakukan tugas-tugas yang pada umumnya memerlukan kecerdasan manusia. AI harus mampu berpikir seperti manusia dan secara rasional, begitu juga berperilaku seperti manusia dan secara rasional. Pada tahun 1950, Alan Turing menciptakan sebuah tes yang disebut *Turing Test*. Tes ini digunakan untuk menguji tingkat kepintaran dari sebuah komputer. Komputer akan dinilai lulus tes tersebut apabila penguji tidak mampu membedakan apakah jawaban tersebut merupakan jawaban manusia atau komputer (Russell & Norvig, 2010).

AI dapat terbagi menjadi dua jenis seperti berikut.

1. *Machine Learning*

   Machine learning merupakan pendekatan *artificial intelligence* yang menggunakan algoritma dan model statistik untuk melatih mesin agar dapat belajar dari data dan membuat prediksi atau keputusan. Ini termasuk sub-bidang *machine learning* seperti *supervised learning* (pembelajaran terawasi), *unsupervised learning* (pembelajaran tanpa pengawasan), dan *reinforcement learning* (pembelajaran dengan penguatan).

2. *Deep Learning*

   Deep learning merupakan sub-bidang pembelajaran mesin yang menggunakan jaringan saraf tiruan dengan banyak lapisan untuk menganalisis data dengan representasi hierarkis. *Deep learning* telah berhasil dalam pengenalan gambar, pemrosesan bahasa alami (*Natural language processing*), dan bidang lainnya.

Artificial Intelligence dapat diterapkan kedalam berbagai bidang kehidupan, seperti kesehatan, transportasi, manufaktur, keuangan, sistem layanan, dan lain-lain. Namun *Artificial Intelligence* tidak selalu berefek baik, AI dapat menggantikan tenaga manusia sehingga menyebabkan berkurangnya lapangan pekerjaan dan lain-lain. Oleh karena itu, dalam pengembangan AI perlu diperhatikan manfaat dan efek dari AI tersebut agar memberi positif kepada manusia dan tidak merugikan manusia (Russell & Norvig, 2010).

2.8 *Machine Learning*

Machine Learning (ML) merupakan cabang dari *artificial intelligence* yang berkaitan dengan pengembangan algoritma dan model statistik. *Machine Learning* bertujuan untuk memungkinkan sebuah komputer dapat belajar secara mandiri dari data yang ada dan dapat menghasilkan sebuah prediksi tanpa deprogram secara eksplisit. Sebuah komputer dapat belajar dengan cara memperbaiki performa dalam mengerjakan sebuah tugas berkali-kali melalui pengalaman sebelumnya. Pengalaman yang dimaksud merupakan bagaimana hasil dari pengerjaan tugas

tersebut cocok dengan dataset yang digunakan atau biasa disebut *fit* (Bi et al., 2019). *Machine Learning* terbagi menjadi tiga jenis seperti berikut.

1. *Supervised Learning*

   Supervised Learning atau pembelajaran terbimbing ini merupakan jenis *machine learning* yang dilatih menggunakan data yang sudah memiliki label atau kelas yang sudah diketahui sebelumnya. Tujuan dari *supervised learning* adalah mengembangkan sebuah model yang dapat memelajari pola dari data training dan melakukan prediksi yang akurat pada data baru yang belum diketahui. Contoh dari algoritma *supervised learning* adalah regresi linier, Naïve Bayes, *decision tree*, SVM, dan lain-lain.

2. *Unsupervised Learning*

   Unspervised Learning atau pembelajaran tanpa pengawasan merupakan jenis *machine learning* yang dilatih menggunakan data yang belum memiliki label atau kelas yang sudah diketahui sebelumnya. Tujuan dari *unsupervised learning* adalah mengidentifikasi pola, struktur, atau kelompok pada sebuah data. Algoritma *unsupervised learning* akan menemukan hubungan dan pola yang tersembunyi dalam data tersebut tanpa memerlukan bimbingan eksternal. Contoh algoritma *unsupervised learning* adalah *k-means clustering*, analisis faktor, algoritma asosiasi, dan lain-lain.

3. *Reinforcement Learning*

   Reinforcement Learning atau pembelajaran penguatan merupakan jenis *machine learning* yang dilatih dengan tujuan menemukan tindakan atau langkah mencapai tujuan yang paling menguntungkan. Algoritma belajar dari tindakan yang diambil dengan tujuan memaksimalkan *reward* yang diterima. Contoh dari algoritma *reinforcement learning* adalah algoritma permainan catur seperti Stockfish, AlphaZero, dan lain-lain.

Selain jenis-jenis *machine learning* di atas, terdapat beberapa sub-bidang *machine learning* lainnya seperti *semi-supervised learning* atau pembelajaran semi-

terbimbing. *Semi-supervised Learning* menggabungkan unsur-unsur dari *supervised learning* dan *unsupervised learning* (Bi et al., 2019). Cabang dari *machine learning* lainnya adalah *deep learning*. *Deep Learning* menggunakan arsitektur jaringan saraf tiruan (*artificial neural network*) yang dalam untuk memelajari data yang lebih kompleks.

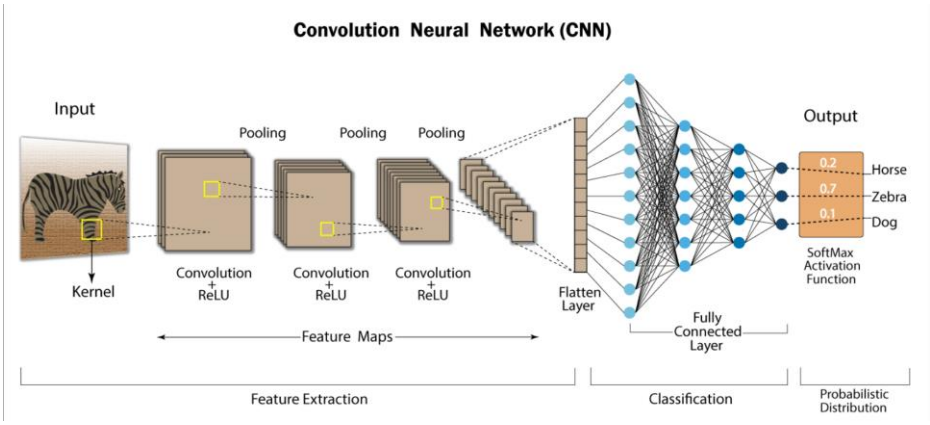
2.9 *Deep Learning*

Deep Learning merupakan salah satu cabang dari *machine learning*. *Deep Learning* menggunakan arsitektur jaringan saraf tiruan (*artificial neural network*) untuk memroses data yang diberikan. Jaringan saraf tiruan ini terdiri dari banyak lapisan (*layer*). Dengan menggunakan *deep learning*, mesin dapat belajar dari data yang diberikan dan memperbaiki performa seiring dengan proses pembelajaran. *Deep Learning* dapat menganalisis data yang lebih kompleks dan abstrak dibandingkan dengan *machine learning*. Namun, *deep learning* membutuhkan data yang besar untuk mampu memberikan hasil yang akurat dan membutuhkan perangkat keras yang bagus untuk memroses data-data tersebut. Selain itu, *deep learning* memerlukan waktu yang lama untuk melatih komputer terhadap dataset yang besar dan kompleks (Fan et al., 2021).

Proses pelatihan *deep learning* menggunakan pembaruan parameter di setiap lapisan berdasarkan perbedaan antara *output* yang dihasilkan oleh model dan *output* yang diharapkan. Ketika dilatih menggunakan data dengan jumlah yang besar, arsitektur jaringan saraf tiruan dalam *deep learning* dapat mengekstraksi fitur-fitur yang berguna secara otomatis dari data yang kompleks tanpa memerlukan pemrograman eksplisit. *Deep Learning* telah mencapai kemajuan yang signifikan dalam berbagai bidang kehidupan seperti pengenalan wajah, pengenalan suara, *natural language processing*, pemodelan bahasa, dan lain-lain. Beberapa contoh arsitektur jaringan saraf tiruan dalam deep learning adalah jaringan saraf konvolusi (*convolutional neural networks*/CNN) biasa digunakan untuk pengenalan gambar, jaringan saraf rekurensi (*recurrent neural networks*/RNN) biasa digunakan untuk pemrosesan urutan data, dan jaringan saraf generatif (*generative neural networks*) seperti jaringan generative berlawanan (*generative adversarial networks*/GAN) untuk menghasilkan data baru yang realistis (Fan et al., 2021).

2.10 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan perkembangan dari konsep *Multi Layer Perceptron* (MLP) yang dikembangkan khusus untuk mengolah data dua dimensi yang berbentuk citra. Secara umum, CNN memiliki beberapa lapisan (*layer*) yang memiliki fungsi yang berbeda-beda. Berikut adalah contoh arsitektur dari CNN secara umum.



Gambar 2.11 Contoh Arsitektur CNN

(Sumber: https://developersbreach.com/convolution-neural-network-deep-learning/)

Gambar 2.11 merupakan contoh arsitektur CNN secara umum. CNN secara umum terbagi menjadi tiga *layer*, yaitu *input layer*, *hidden layer*, dan *output layer*. *Input layer* merupakan *layer* dimana citra dimasukkan ke dalam CNN. Dalam *hidden layer* terdapat lapisan konvolusi, lapisan aktivasi, lapisan *pooling*, dan *fully connected layer*. Lapisan konvolusi berfungsi untuk mengekstrak fitur-fitur yang terdapat dalam sebuah citra dengan cara menerapkan filter-filter terhadap citra tersebut. Lapisan aktivasi berfungsi untuk mengaktifkan fitur-fitur yang telah ditemukan oleh filter-filter tersebut.

Lapisan *pooling* berfungsi untuk memperkecil ukuran volume output. Lapisan *pooling* diterapkan diantara lapisan konvolusi dalam arsitektur CNN. Lapisan *pooling* digunakan untuk mengurangi jumlah parameter yang digunakan sehingga dapat mempercepat komputasi. *Fully connected layer* berfungsi untuk menggabungkan fitur-fitur yang telah diekstrak dari citra. Kemudian CNN

menggunakan algoritma *backpropagation* untuk memperbarui bobot dan bias pada masing-masing *layer* agar hasil output menjadi lebih akurat (Alzubaidi et al., 2021).

2.11 Inception-v4

Inception atau yang bernama lain *GoogleNet*, merupakan sebuah model jaringan saraf yang dikembangkan oleh peneliti-peneliti di Google. Versi terbaru dari arsitektur tersebut adalah Inception-v4. Inception-v4 merupakan pengembangan lanjutan dari model-model *Inception* sebelumnya, yaitu Inception-v1, Inception-v2, dan Inception-v3. Model arsitektur Inception dikembangkan dengan tujuan mengoptimalkan efisiensi komputasi dan meningkatkan akurasi dalam pengenalan citra. Arsitektur ini menggunakan konsep yang dikenal sebagai "*Inception module*" atau "*Inception block*" yang dirancang untuk mengekstraksi fitur-fitur dengan berbagai ukuran dan tingkat kompleksitas dari citra (Al Husaini et al., 2022). Berikut adalah beberapa fitur utama dari arsitektur Inception-v4, antara lain:

1. *Inception Module*

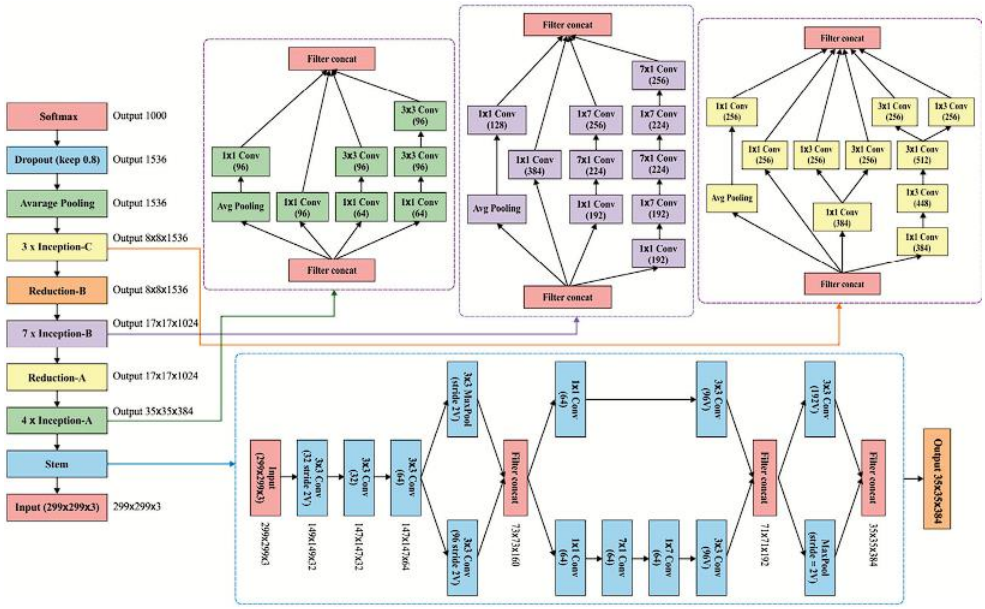
   Inception-v4 menggunakan beberapa modul *Inception* yang berisi beberapa jalur paralel untuk melakukan ekstraksi fitur. *Inception module* menggabungkan konvolusi dengan kernel yang memiliki ukuran berbeda, yaitu 1x1, 3x3, dan 5x5, serta pooling berukuran 3x3, untuk menangkap fitur-fitur pada berbagai skala spasial.
2. *Residual Connections*

   Arsitektur Inception-v4 juga menggunakan koneksi residu. Hal ini memungkinkan aliran informasi langsung melalui lapisan-lapisan dalam jaringan. Koneksi residu dapat membantu mencegah terjadinya masalah penurunan gradien dan membuat waktu pelatihan menjadi lebih cepat.
3. *Factorization into smaller convolutions*

   Arsitektur Inception-v4 menggunakan faktorisasi konvolusi yang lebih kecil untuk mengurangi tingginya kompleksitas komputasi. Contohnya, konvolusi 5x5 dapat digantikan dengan menggunakan dua konvolusi 3x3 berturut-turut.
4. *Average Pooling*

Sebagai pengganti *fully connected layers* di bagian akhir jaringan saraf, Inception-v4 menggunakan *average pooling global* untuk menghasilkan vektor fitur akhir. Pendekatan ini membantu mengurangi jumlah parameter dalam jaringan saraf dan dapat mencegah terjadinya *overfitting*

Berikut merupakan gambar contoh arsitektur Inception-v4.



Gambar 2.12 Contoh Arsitektur Inception-v4

(Shankar et al., 2020)

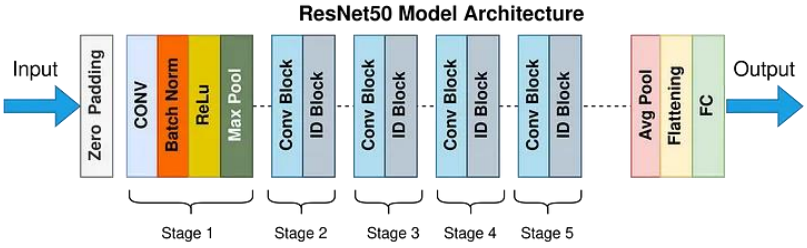
2.12 ResNet-50

Residual Network-50 atau biasa disebut ResNet-50 merupakan salah satu macam model arsitektur dari *Convolutional Neural Network* (CNN) yang populer digunakan dalam *deep learning*. ResNet-50 dikembangkan oleh tim *Microsoft Research* pada tahun 2015. Model arsitektur ini diberikan nama ResNet-50 dikarenakan model ini menambahkan 50 *layer* tambahan pada CNN.

Pada model arsitektur ResNet-50, terdapat konsep baru yaitu *shortcut connection* atau *skip connection*. Hal ini menyebabkan model arsitektur tersebut dapat melakukan *skip* atau melompati beberapa *layer* dalam pemrosesan citra dan menggabungkan informasi yang didapat dari *layer* sebelumnya dengan *layer* saat ini. *Skip connection* ini dikembangkan untuk menjadi solusi atas permasalahan

menghilangnya gradien (*vanishing gradient*) yang sering terjadi pada model arsitektur CNN yang berukuran besar. Hal ini diakibatkan karena adanya kekeliruan dalam stabilitas nilai parameter yang digunakan. Dengan menggunakan konsep *skip connection*, ResNet-50 dapat mempelajari fitur-fitur yang lebih kompleks dan abstrak pada citra. Hal ini menyebabkan meningkatnya kinerja dan akurasi pada tugas-tugas seperti klasifikasi gambar, segmentasi objek, deteksi objek dan lain-lain.

ResNet50 terdiri dari beberapa blok, di mana setiap blok terdiri dari beberapa *layer* konvolusi dan aktivasi, diikuti dengan sebuah skip connection. Blok-blok tersebut dapat diulang beberapa kali untuk meningkatkan kedalaman jaringan. Selain itu, ResNet-50 juga menggunakan lapisan *pooling* dan lapisan *fully connected* pada bagian akhir jaringan. Berikut merupakan gambar contoh arsitektur dari ResNet-50.



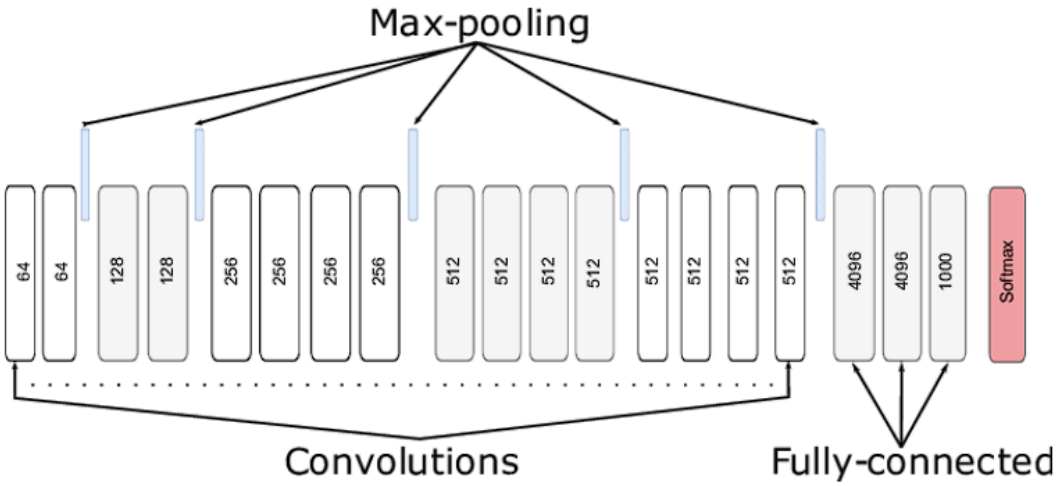
Gambar 2.13 Arsitektur ResNet-50

(Sumber: https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758)

2.13 VGG-19

VGG-19 merupakan pengembangan dari VGGNet (*Visual Geometry Group Network*) adalah sebuah arsitektur jaringan saraf konvolusi (*convolutional neural network*) atau biasa disebut CNN. VGGNet dikembangkan oleh tim *Visual Geometry Group* di Universitas Oxford. Arsitektur VGGNet pertama kali diperkenalkan oleh Karen Simonyan dan Andrew Zisserman pada tahun 2014 dalam sebuah makalah yang berjudul "Very Deep Convolutional Networks for Large-Scale Image Recognition"(Simonyan & Zisserman, 2014). VGGNet dikembangkan untuk meningkatkan performa dari arsitektur CNN dengan cara

menambah kedalaman dari arsitektur tersebut. VGG-19 merupakan arsitektur VGGnet yang menggunakan lapisan konvolusi sejumlah 19. Arsitektur ini memanfaatkan konvolusi dengan jarak langkah (*stride*) dan penyaring konvolusi kecil dengan ukuran 3x3 yang dijalankan secara berulang-ulang. Arsitektur VGG-19 juga menggunakan *max pooling* dengan jarak langkah (*stride*) 2 untuk mengurangi dimensi dari fitur yang akan dihasilkan. Secara keseluruhan, VGG-19 memberikan representasi fitur yang lebih jelas dan akurat dengan beban komputasi yang lebih tinggi jika dibandingkan dengan arsitektur-arsitektur CNN lainnya yang lebih dangkal (Simonyan & Zisserman, 2014).



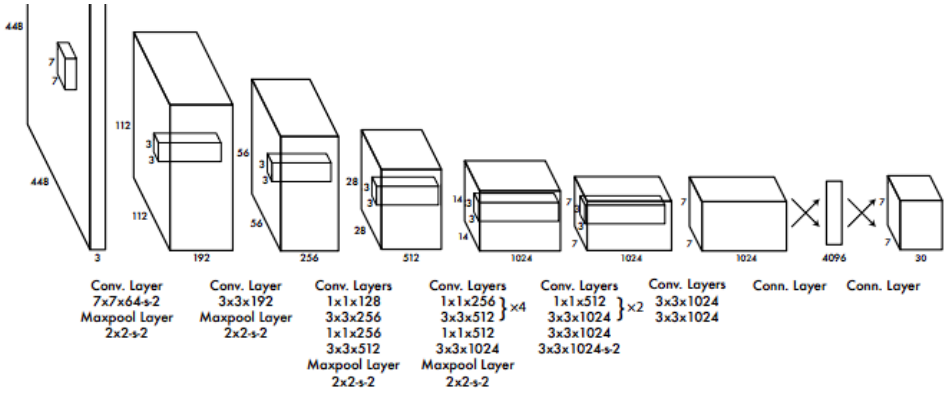
Gambar 2.14 Contoh Arsitektur VGG-19

(Kamal et al., 2023)

2.14 YOLO V5

Memiliki kepanjangan *You Only Look Once*, YOLO merupakan sistem deteksi objek yang memproses citra dan mengidentifikasi objek yang berada di dalamnya. Arsitektur ini diperkenalkan oleh Joseph Radmon, Santosh Divvala, Ross Girshick, dan Ali Farhadi dalam jurnal berjudul "*You Only Look Once: Unified, Real-Time Object Detection*" yang diterbitkan pada tahun 2016. Arsitektur YOLO telah mencapai versi ke-8, namun untuk penelitian ini yang akan digunakan adalah YOLO V5. YOLO V5 merupakan arsitektur YOLO pertama yang diterapkan di Pytorch. Kelebihan dari arsitektur YOLO V5 adalah cepatnya proses

training dan ringannya proses komputasi. Arsitektur YOLO V5 memiliki 5 ukuran yaitu nano (n), *small* (s), *medium* (m), *large* (l), dan *extra large* (x). Semakin besar ukuran arsitektur yang digunakan, maka semakin lama dan semakin membutuhkan memori yang lebih besar untuk menjalankan proses *training* dan prediksi. Berikut merupakan contoh gambar arsitektur YOLO.

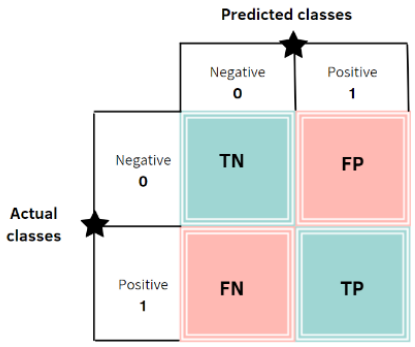


Gambar 2.15 Contoh Arsitektur YOLO

(Redmon et al., 2015)

2.15 *Confusion Matrix*

Confusion matrix adalah sebuah tabel yang digunakan untuk menampilkan visualisasi dari performa sebuah algoritma *supervised machine learning* atau *deep learning*. Kolom dari tabel *confusion matrix* menampilkan label kelas yang diprediksi, sedangkan baris menampilkan label kelas yang sebenarnya. Ukuran dari *confusion matrix* beragam menyesuaikan dengan jumlah label. Berikut adalah contoh *confusion matrix* berukuran 2x2:



Gambar 2.16 Contoh *Confusion Matrix* Ukuran 2x2

(Sumber: https://pub.towardsai.net/deep-understanding-of-confusion-matrix-6ab1f88a267e)

Pada Gambar 2.16 terdapat kelas hasil prediksi dan kelas sebenarnya. *True Negative* (TN) merupakan jumlah kelas negatif yang berhasil diklasifikasi sebagai negatif. *True Positive* (TP) merupakan jumlah kelas positif yang berhasil diklasifikasi sebagai positif. *False Positive* (FP) merupakan jumlah kelas negatif yang diklasifikasi sebagai positif. Sedangkan *False Negative* (FN) merupakan jumlah kelas positif yang diklasifikasi sebagai negatif. Dari empat elemen tersebut, dapat digunakan untuk menghitung nilai *accuracy*.

Accuracy adalah rasio prediksi benar terhadap total prediksi. *Accuracy* menunjukkan seberapa baik model dalam melakukan prediksi dengan benar secara keseluruhan. Berikut persamaan untuk menghitung nilai *accuracy*:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2\text{-}9)$$

TP = *True Positive*

TN = *True Negative*

FP = *False Positive*

FN = *False Negative*

2.16 Python

Python merupakan bahasa pemrograman tingkat tinggi bersifat *open source* yang dapat digunakan untuk banyak kasus (general purpose). Python diciptakan oleh Guido van Rosum dan pertama kali dirilis pada 20 Februari 1991. Nama python terinspirasi dari acara favorit Guido yang tayang di BBC dengan nama *Monty Python's Flying Circus*. Pada tahun 2021 berdasarkan survey pada *website stack overflow*, python digunakan oleh 48.24% developer di seluruh dunia dari 83,052 responden.

2.17 Kaggle

Kaggle adalah sebuah *platform online* berbasis *website* yang menyediakan data, sumber daya, dan kompetisi untuk pengembang, peneliti, dan ilmuwan data untuk berpartisipasi dalam menyelesaikan sebuah tantangan data. Kaggle juga

menyediakan berbagai sumber daya seperti *dataset*, forum diskusi, dan *tutorial* pemrosesan data dan *machine learning*.

Pada *platform* Kaggle, pengguna dapat mengikuti kompetisi data yang diadakan oleh perusahaan-perusahaan atau organisasi yang membutuhkan bantuan dalam menyelesaikan sebuah permasalahan terkait dengan *machine learning* atau *data science*. Kompetisi ini dapat memberikan penghargaan berupa uang maupun kesempatan kerja pada perusahaan yang mengadakan kompetisi tersebut. Akibat adanya kompetisi-kompetisi ini, Kaggle menjadi *platform* yang populer dan telah membantu memecahkan banyak permasalahan di bidang kesehatan, keuangan, teknologi, dan lain-lain.

2.18 Google Colaboratory

Google Colaboratory atau yang biasa disebut Colab adalah sebuah *platform* gratis berbasis *cloud* yang dikembangkan oleh tim *Google Research* untuk mempermudah pekerjaan yang berkaitan dengan *data science* dan *machine learning*. Colab menyediakan lingkungan (*environment*) pengembangan interaktif yang memungkinkan pengguna untuk membuat, menjalankan, dan membagikan kode python melalui *web*. Colab memiliki format *notebook* layaknya *Jupyter Notebook*, sehingga memungkinkan kita untuk menulis kode per bagian. Selain itu Colab juga menyediakan akses ke mesin *virtual google* yang dilengkapi dengan GPU dan TPU, yang memungkinkan pengguna untuk memproses data dengan lebih cepat.

2.19 Microsoft Visual Studio Code

Microsoft Visual Studio Code (VS Code) adalah salah satu editor kode teks yang dikembangkan oleh Microsoft. VS Code dapat digunakan pada berbagai sistem operasi seperti Windows, Linux, dan macOS. Dirilis pada 29 April 2015, VS Code mampu digunakan untuk 15 bahasa pemrograman. Beberapa fitur yang disediakan oleh VS Code adalah *debugging*, pengeditan kode cerdas, integrasi terminal, dan lain-lain.

2.20 Uji Normalitas

Uji Normalitas merupakan sebuah metode untuk mengevaluasi apakah data dalam suatu kelompok atau variabel mengikuti distribusi normal atau tidak. Tujuannya adalah untuk menentukan apakah data yang telah dikumpulkan berasal dari populasi yang memiliki distribusi normal. Meskipun secara empiris banyak ahli statistik menyatakan bahwa data dengan jumlah sampel lebih dari 30 dapat diasumsikan mengikuti distribusi normal, namun untuk memastikan, disarankan untuk menggunakan uji normalitas. Hal ini karena jumlah sampel yang besar tidak selalu menjamin distribusi normal, begitu pula sebaliknya untuk jumlah sampel yang kurang dari 30. Beberapa uji statistik yang umum digunakan untuk menguji normalitas antara lain Uji Chi-Square, Kolmogorov Smirnov, Lilliefors, Shapiro Wilk, dan Jarque Bera.

2.20.1 Uji Shapiro Wilk

Uji Shapiro Wilk adalah sebuah teknik atau formula untuk mengukur sebaran data yang dikembangkan oleh Shapiro dan Wilk. Metode ini efektif dan dapat diandalkan untuk menguji normalitas pada sampel dengan jumlah yang relatif kecil. Dalam praktiknya, para peneliti dapat menggunakan perangkat lunak statistik seperti SPSS dan STATA untuk menerapkan uji ini. Berikut adalah persamaan dari uji Shapiro Wilk.

$$W = \frac{(\sum_{i=1}^{m} a_i (X_{(n+1-i)} - X_i))^2}{\sum_{i=1}^{n} (X_i - \bar{X})^2} \quad (2\text{-}10)$$

W = Koefisien Uji Shapiro Wilk

$\bar{X}$ = Rata-rata sampel

X_i = Angka ke i pada sampel

m = jika genap $\frac{n}{2}$, jika ganjil $\frac{n-1}{2}$

Setelah menemukan nilai W, cari nilai W pada tabel Uji Shapiro Wilk untuk menemukan *p-value*. Nilai *p-value* akan dibandingkan dengan nilai alpha (α) untuk pengambilan kesimpulan.

2.21 Uji Rata-Rata (*Mean*)

Uji rata-rata (*mean*) adalah metode statistik yang digunakan untuk menguji rata-rata populasi atau kelompok data terdapat perbedaan secara signifikan. Uji rata-rata berguna untuk membandingkan rata-rata sampel dengan nilai teoritis atau referensi yang diberikan.

2.21.1 Uji Anova

Uji Anova *(Analysis of Variance)* adalah sebuah analisis statistik yang digunakan untuk menguji perbedaan rata-rata dari lebih dari dua populasi atau kelompok yang berbeda. Uji ini berfungsi untuk mengidentifikasi apakah ada perbedaan signifikan antar rata-rata populasi yang dibandingkan. Uji Anova mengukur varian (*variance*) data dalam populasi dan varian (*variance*) antar populasi. Jika varian antar populasi lebih besar dibandingkan varian dalam populasi, maka dapat dinyatakan bahwa terdapat perbedaan signifikan antara rata-rata populasi tersebut. Uji ANOVA digunakan jika data yang digunakan berdistribusi normal. Berikut merupakan contoh macam uji ANOVA yang sering digunakan, seperti:

a. *One-Way* ANOVA: Digunakan ketika ada satu faktor atau variabel bebas yang mempengaruhi satu variabel dependen. Berikut adalah persamaan *One-Way* ANOVA.

$$SST = \sum (X_{ij} - \bar{X})^2 \tag{2-11}$$

SST = Jumlah Kuadrat Total

$\bar{X}$ = Rata-rata seluruh kelompok

X_{ij} = Angka ke I pada kelompok j

$$SSB = \sum n_j (\bar{X}_j - \bar{X})^2 \tag{2-12}$$

SSB = Jumlah Kuadrat Antara

$\bar{X}_j$ = Rata-rata kelompok j

$$SSW = \sum\sum (X_{ij} - \bar{X}_j)^2 \quad (2\text{-}13)$$

SSW = Jumlah Kuadrat Dalam

$$df_a = j - 1 \quad (2\text{-}14)$$

df_a = *degree of freedom* (derajat kebebasan) kelompok

$$df_r = total\ data - j \quad (2\text{-}15)$$

df_r = *degree of freedom* (derajat kebebasan) residual

$$MS_a = \frac{SSW}{df_a} \quad (2\text{-}16)$$

MS_a = *mean square* (rata-rata Kuadrat) kelompok

$$MS_r = \frac{SSB}{df_r} \quad (2\text{-}16)$$

MS_r = *mean square* (rata-rata Kuadrat) residual

$$F = \frac{MS_a}{MS_r} \quad (2\text{-}16)$$

F = F statistik

Setelah mendapatkan nilai F statistik, bandingkan nilai F statistik dengan nilai kritis menggunakan tabel distribusi F dengan alpha dan derajat kebebasan yang sesuai. Hasil perbandingan tersebut digunakan untuk pengambilan kesimpulan dari hipotesis yang telah ditetapkan.

b. *Two-Way* ANOVA: Digunakan ketika ada dua faktor atau variabel bebas yang mempengaruhi satu variabel dependen.
c. *Repeated Measures* ANOVA: Digunakan ketika mengukur data pada kelompok yang sama pada waktu yang berbeda.

2.21.2 Uji Tukey HSD

Uji Tukey HSD (*Honestly Significant Difference*) adalah metode statistik yang digunakan untuk membandingkan rata-rata antara dua kelompok atau lebih dalam analisis varians (ANOVA). Uji ini dirancang untuk mengidentifikasi perbedaan signifikan di antara kelompok-kelompok tersebut setelah dilakukan uji ANOVA. Hasil dari Uji Tukey HSD menunjukkan bahwa terdapat perbedaan secara keseluruhan di antara kelompok-kelompok tersebut. Berikut adalah persamaan uji Tukey HSD.

$$Q = \frac{\bar{X}_i - \bar{X}_j}{\sqrt{\frac{MS_a}{n}}} \tag{2-16}$$

Q = Hasil uji Tukey HSD

$\bar{X}_i$ = Rata-rata kelompok i

$\bar{X}_j$ = Rata-rata kelompok j

MS_a = *mean square* (rata-rata Kuadrat) kelompok

n = Jumlah sampel setiap kelompok

Setelah mendapatkan nilai Q, bandingkan nilai Q dengan nilai kritis menggunakan tabel distribusi q dengan alpha dan derajat kebebasan yang sesuai. Hasil perbandingan tersebut digunakan untuk pengambilan kesimpulan dari hipotesis yang telah ditetapkan.

2.21.3 Uji Kruskal-Wallis

Uji Kruskal-Wallis adalah metode statistik non-parametrik yang digunakan untuk menguji apakah terdapat perbedaan signifikan antara tiga atau lebih kelompok. Dalam uji ini, hipotesis nol menyatakan bahwa tidak ada perbedaan signifikan antara median kelompok-kelompok tersebut. Uji ini sering digunakan sebagai alternatif ketika asumsi dari uji ANOVA tidak terpenuhi. Salah satunya adalah ketika data yang digunakan tidak berdistribusi normal. Metode ini dinamai sesuai dengan nama dua statistikawan, Maurice Kruskal dan William Wallis, yang

mengembangkannya pada tahun 1952. Berikut adalah persamaan uji Kruskal-Wallis.

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{{R_i}^2}{n_i} - 3(N+1) \quad (2\text{-}17)$$

H = Hasil uji Kruskal-Wallis

N = Jumlah sampel semua kelompok

k = Jumlah kelompok

R_i = Jumlah peringkat dalam kelompok i

n_i = Jumlah sampel kelompok i

Setelah mendapatkan nilai H, bandingkan nilai H dengan nilai kritis menggunakan tabel distribusi *chi-square* dengan alpha dan derajat kebebasan yang sesuai. Hasil perbandingan tersebut digunakan untuk pengambilan kesimpulan dari hipotesis yang telah ditetapkan.

2.21.4 Uji Mann-Whitney U

Uji Mann-Whitney U adalah uji statistik non-parametrik yang digunakan untuk membandingkan median dari dua sampel independen. Metode ini menjadi pilihan alternatif untuk uji T jika data tidak terdistribusi normal. Uji ini didasarkan pada peringkat observasi dalam dua sampel, bukan pada nilai aktualnya. Hipotesis nol dari uji ini menyatakan bahwa kedua sampel berasal dari populasi yang sama, sementara hipotesis alternatifnya menyatakan bahwa keduanya berasal dari populasi yang berbeda. Uji dapat dilakukan secara manual atau menggunakan perangkat lunak statistik, seperti SPSS. Berikut adalah persamaan uji Mann-Whitney U.

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1 \quad (2\text{-}18)$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2 \quad (2\text{-}19)$$

H = Hasil uji Mann-Whitney U

n_1 = Jumlah sampel pertama

n_2 = Jumlah sampel kedua

R_1 = Total peringkat dari sampel pertama

R_2 = Total peringkat dari sampel kedua

Nilai U yang lebih kecil akan digunakan sebagai nilai statistik uji. Jika nilai statistik uji bernilai lebih kecil dari alpha yang ditentukan, maka dianggap terdapat perbedaan signifikan diantara dua kelompok tersebut.

2.22 Formulasi Hipotesis

Sebelum menggunakan uji normalitas dan uji statistik, diperlukan penentuan hipotesis yang akan digunakan untuk pengambilan kesimpulan nantinya. Terdapat dua macam hipotesis yang akan digunakan dalam uji statistik, yaitu hipotesis nol (H0) dan hipotesis alternatif (H1). Hipotesis nol merupakan asumsi bahwa tidak terdapat perbedaan signifikan antara dua populasi atau variabel. Hipotesis satu merupakan asumsi bahwa terdapat perbedaan signifikan antara dua populasi atau variabel.

Proses pengambilan kesimpulan ditentukan oleh nilai *p-value* dan nilai alpha (α). *P-value* adalah ukuran probabilitas apakah hipotesis nol dapat diterima atau ditolak. Nilai alpha atau yang biasa dilambangkan dengan α adalah tingkat kesalahan maksimal yang akan ditentukan sendiri oleh peneliti. Pada umumnya nilai α yang digunakan adalah 1% (0.01), 5% (0.05), atau 10% (0.1). Kriteria penerimaan atau penolakan H0 adalah H0 diterima jika nilai *P-value* lebih besar dari nilai α, H0 ditolak jika nilai *P-value* lebih kecil dari nilai α.

Berikut merupakan penyusunan hipotesis untuk uji normalitas:

H0 : data berasal dari populasi yang berdistribusi normal

H1 : data berasal dari populasi yang tidak berdistribusi normal

Berikut merupakan penyusunan hipotesis untuk uji ANOVA:

H0 : tidak ada korelasi, pengaruh, dan perbedaan antar dua populasi atau lebih.

H1 : ada korelasi, pengaruh, dan perbedaan antar dua populasi atau lebih.

Berikut merupakan penyusunan hipotesis untuk uji Tukey HSD:

H0 : tidak ada korelasi, pengaruh, dan perbedaan antar dua populasi.

H1 : ada korelasi, pengaruh, dan perbedaan antar dua populasi.

Berikut merupakan penyusunan hipotesis untuk uji Kruskal-Wallis:

H0 : tidak ada korelasi, pengaruh, dan perbedaan antar dua populasi atau lebih.

H1 : ada korelasi, pengaruh, dan perbedaan antar dua populasi atau lebih.

Berikut merupakan penyusunan hipotesis untuk uji Mann-Whitney U:

H0 : tidak ada korelasi, pengaruh, dan perbedaan antar dua populasi.

H1 : ada korelasi, pengaruh, dan perbedaan antar dua populasi.

2.23 Penelitian Terdahulu

Beberapa penelitian terdahulu terkait deteksi penyakit retinopati diabetik digunakan untuk menjadi referensi dalam melaksanakan penelitian ini.

Shankar dkk., (2020) melakukan penelitian mengembangkan *Hyper Parameter Tuning Inception-v4* (HPTI-v4) untuk mengklasifikasi penyakit retinopati diabetik. Pada penelitian tersebut, digunakan *preprocessing* citra berupa *contrast limited adaptive histogram equalization* (CLAHE). Dataset yang digunakan berisi 1200 citra yang terdiri dari 542 citra mata normal, 154 citra mata RD ringan, 248 citra mata RD sedang dan berat, dan 255 citra mata PRD. Arsitektur HPTI-v4 berhasil mencapai tingkat akurasi 99,49%.

Penelitian yang dilakukan oleh Sudha dan Ganeshbabu (2021) berfokus pada *lesion detection and grading* pada penyakit retinopati diabetik. Penelitian ini menggunakan arsitektur VGG-19. Penelitian ini juga menggunakan *saliency map* dan *gradient descent method* untuk proses *preprocessing* citra. Dataset yang digunakan berisi 35.126 citra, dengan 25.810 citra mata normal, 2.443 citra RD ringan, 5.292 citra RD sedang, 873 citra RD berat, dan 708 citra PRD. Akurasi arsitektur VGG-19 pada penelitian ini mencapai 96%.

Lin dan Wu (2023) melakukan penelitian mengembangkan ResNet-50 untuk deteksi retinopati diabetik. Penelitian ini membandingkan performa dari

ResNet-50 dengan *revised* ResNet-50 yang menggunakan *adaptive learning rate*. Dataset yang digunakan berisi 35.126 citra, dengan 25.810 citra mata normal, 2.443 citra RD ringan, 5.292 citra RD sedang, 873 citra RD berat, dan 708 citra PRD. Akurasi dari model ResNet-50 mencapai 89%, sedangkan model *revised* ResNet-50 mencapai 83,95%.

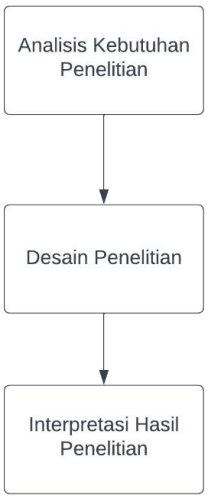
Mishra dkk., (2020) melakukan penelitian menerapkan *preprocessing* citra *cropping* untuk mengoptimasi model *deep learning* dalam melakukan pengawasan bencana banjir. Pada penelitian ini, *cropping* digunakan untuk hanya menunjukkan RDainase pada citra tersebut dan diklasifikasi menjadi 3 kelompok, yaitu *fully blocked*, *partially blocked*, dan *no blockage*. *Cropping* berhasil meningkatkan akurasi model *deep learning* dari 55% menjadi 76%.

Penelitian yang dilakukan Yang dkk., (2022) menerapkan berbagai macam augmentasi data pada berbagai macam model dan dataset. Jenis-jenis augmentasi yang dilakukan berupa *image manipulation* seperti *rotation*, *flipping*, dan lain-lain, *image erasing*, *image mix*, dan lain-lain. Pada penelitian tersebut disimpulkan bahwa augmentasi data merupakan solusi efektif apabila mengalami kekurangan data.

BAB III
RANCANGAN PENELITIAN

3.1 Tahapan Penelitian

Penelitian ini bertujuan untuk mengetahui apakah *preprocessing* citra dan augmentasi data yang dilakukan pada sistem deteksi penyakit retinopati diabetik dapat memengaruhi tingkat akurasi model. Tahapan penelitian dari proyek tugas akhir ini adalah sebagai berikut.



Gambar 3.1 Tahapan Penelitian

3.2 Analisis Kebutuhan Penelitian

Tahapan pertama dari penelitian tugas akhir ini adalah analisis kebutuhan penelitian. Analisis yang dilakukan untuk mengetahui pengaruh *preprocessing* citra dan augmentasi data pada sistem deteksi retinopati diabetik mencakup kebutuhan perangkat keras dan perangkat lunak.

A. Perangkat Keras

1. Laptop Ideapad 3 Slim 3 14inch
   i. Processor : AMD Ryzen 5 5500U
   ii. GPU : AMD Radeon™ Graphics
   iii. Memory : RAM 8 GB, 512 GB SSD

iv. OS : Windows 11 Home Single Language 64-bit

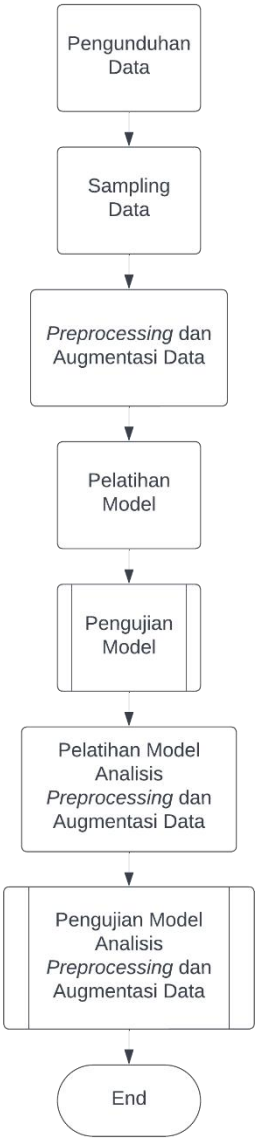
B. Perangkat Lunak

1. Python 3
2. Google Colab
3. Microsoft Studio Code

C. Dataset

1. Citra Fundus Retinopati Diabetik Seluruh Kelas

3.3 Desain Penelitian



Gambar 3.2 Desain Penelitian

Penelitian ini dimulai dari melakukan pengunduhan data. Data retinopati diabetik diunduh dari *website* Kaggle dengan judul "*Diabetic Retinopathy Detection*". Setelah diunduh, dataset akan di-*sampling* untuk membuat dataset asli. Lalu dataset asli diterapkan *preprocessing* citra dan augmentasi data untuk membuat dataset setelah *preprocessing* dan augmentasi citra. *Preprocessing* citra yang digunakan merupakan *cropping* dan *histogram equalization*. Augmentasi data yang dilakukan merupakan transformasi *flipping*. Masing-masing dataset akan dibagi menjadi data *train*, *test*, dan *validation* dengan rasio pembagian 70%, 20%, dan 10%.

Tahap pelatihan model akan dilakukan dengan menggunakan dataset setelah *preprocessing* dan augmentasi citra untuk menentukan arsitektur terbaik untuk mengklasifikasi penyakit RD. Arsitektur *deep learning* yang digunakan adalah Inception-v4, ResNet-50, VGG-19, dan YOLO v5Nano. *Output* dari proses pelatihan ini adalah empat buah model, *confusion matrix*, dan nilai *accuracy* yang akan dievaluasi. *Output* tersebut akan dievaluasi menggunakan uji statistik untuk mengetahui perbandingan performa dari empat model tersebut.

Selanjutnya adalah tahap pelatihan model analisis *preprocessing* dan augmentasi data. Pada tahap ini akan dilakukan pelatihan arsitektur terbaik terhadap dataset asli, dataset *cropping*, dataset histeq, dataset *flipping*, dataset *cropping* dan histeq, dataset *cropping* dan *flipping*, dataset histeq dan *flipping*, dan dataset setelah *preprocessing* dan augmentasi citra. Hal ini dilakukan agar dapat mengetahui pengaruh dari masing-masing *preprocessing* citra atau augmentasi citra terhadap performa dari model. *Output* dari proses pelatihan ini adalah delapan buah model, *confusion matrix*, dan nilai *accuracy* yang akan dievaluasi. *Output* tersebut akan dievaluasi menggunakan uji statistik untuk mengetahui perbandingan performa dari delapan model tersebut.

3.3.1 Pengunduhan Data

Dataset retinopati diabetik diunduh dari *website* Kaggle dengan judul "*Diabetic Retinopathy Detection*". Dataset ini berisi lima belas *files* dengan ukuran *files* tersebut mencapai 88,29 GB. Dari lima belas *files* yang ada, hanya *files train*.zip, dan *trainLabels*.csv.zip yang akan digunakan.

Berikut adalah rincian dari *files* tersebut:

1. Lima set data latih (*train*.zip)

   Total data berjumlah 35.126 citra, dengan 25.810 citra di kelas mata normal, 2.443 citra di kelas RD ringan, 5.292 citra di kelas RD sedang, 873 citra di kelas RD berat, dan 708 citra di kelas RD proliferatif.
2. Label dari data latih dalam format .csv (*trainLabels*.csv.zip)

   Berisi label citra dari data latih.

Masing-masing citra retina dalam dataset tersebut memiliki resolusi dan ukuran yang berbeda-beda. Setiap citra tersebut diberi label dengan format id, subjek, dan keterangan mata dari subjek tersebut. Contoh pelabelan citra-citra tersebut adalah "*1_left*.jpeg" yang berarti citra tersebut diambil dari mata kiri subjek dengan id satu. Citra retina diklasifikasikan ke dalam lima tingkatan retinopati diabetik dengan label 0 hingga 4. Label 0 merupakan citra fundus mata normal, label 1 merupakan citra fundus RD ringan, label 2 merupakan citra fundus RD sedang, label 3 merupakan citra fundus RD berat, dan label 4 merupakan citra fundus RD proliferatif. Berikut adalah contoh sampel gambar dari dataset tersebut:

(A) (B) (C) (D) (E)

Gambar 3.3 Contoh Citra Fundus (A) Normal, (B) RD Ringan, (C) RD Sedang, (D) RD Berat, (E) RD Proliferatif

Pada penelitian ini, terdapat delapan dataset yang akan digunakan, yaitu dataset asli, *cropping*, histeq, *flipping*, *cropping* dan histeq, *cropping* dan *flipping*, histeq dan *flipping*, dan setelah *preprocessing* dan augmentasi citra. Dataset tersebut kemudian akan dibagi untuk proses *training*, *testing*, dan *validation*. Berikut adalah pembagian citra pada masing-masing dataset.

Jumlah data citra pada tabel 4.1 dihitung berdasarkan pembagian dataset yaitu 70% untuk data *train*, 20% untuk data *test*, dan 10% untuk data *validation*. Dataset sebelum *preprocessing* akan langsung digunakan untuk proses *training* dan

diterapkan *preprocessing* dan augmentasi citra untuk membuat dataset setelah *preprocessing*.

3.3.2 *Sampling* Data

Sebelum diterapkan *preprocessing* dan augmentasi citra, dataset akan di-*sampling* agar jumlah citra untuk setiap kelas menjadi sama. Hal ini dilakukan untuk mengurangi waktu dan sumber daya yang diperlukan untuk melakukan pelatihan model. *Sampling* data juga dilakukan untuk mencegah terjadinya *overfitting* saat pengujian model. Jumlah citra untuk tiap kelas yang akan digunakan merupakan 500 citra per kelas. Sehingga jumlah citra untuk dataset asli adalah 2500 citra.

3.3.3 *Preprocessing* dan Augmentasi Data

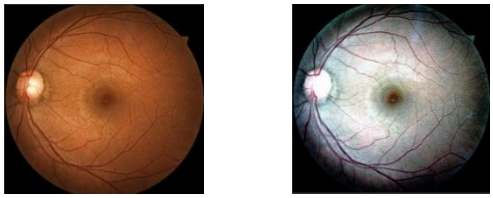
Dataset asli akan diterapkan *preprocessing* citra dan augmentasi data untuk menghasilkan dataset yang baru. *Preprocessing* citra yang dilakukan terhadap dataset asli adalah *cropping* dan *histogram equalization*. Masing-masing citra fundus akan diterapkan *preprocessing* citra yang telah disebutkan sebelumnya. Pada langkah pertama, citra fundus akan diterapkan *cropping* untuk menghilangkan bagian dari citra fundus yang tidak diinginkan. Berikut adalah contoh citra fundus yang telah diterapkan proses *cropping*:

(A) (B)

Gambar 3.4 Contoh Citra Fundus (A) Sebelum *Cropping*, (B) Setelah *Cropping*

Setelah *cropping*, langkah yang diterapkan berikutnya adalah *histogram equalization*. Hal ini dilakukan dengan menerapkan *histogram equalization* pada

citra fundus. Berikut adalah contoh citra fundus yang telah diterapkan proses *histogram equalization*:



(A) (B)

Gambar 3.5 Contoh Citra Fundus (A) Sebelum *Histogram Equalization*, (B) Setelah *Histogram Equalization*

Setelah dilakukan *preprocessing* citra akan dilakukan augmentasi citra untuk menambah jumlah citra dalam dataset, meningkatkan akurasi, dan mencegah *overfitting*. Augmentasi citra yang dilakukan adalah *flipping*, yaitu *horizontal flipping* dan *vertical flipping*. Masing-masing citra fundus akan diterapkan *horizontal flipping* menjadi citra yang baru, *vertical flipping* menjadi citra yang baru, dan *horizontal* dan *vertical flipping* menjadi citra yang baru. Jumlah citra dalam masing-masing kelas akan diterapkan augmentasi citra agar jumlah citra tiap kelas bertambah dari 500 citra menjadi 2000 citra. Sehingga jumlah citra dalam dataset setelah *preprocessing* citra dan augmentasi adalah 10000 citra.

Berikut adalah contoh citra fundus yang telah diterapkan augmentasi data berupa *flipping*:

(A) (B) (C) (D)

Gambar 3.6 Contoh Citra (A) Sebelum *Flipping*, (B) Setelah *Horizontal Flipping*, (C) Setelah *Vertical Flipping*, (D) Setelah *Horizontal* dan *Vertical Flipping*

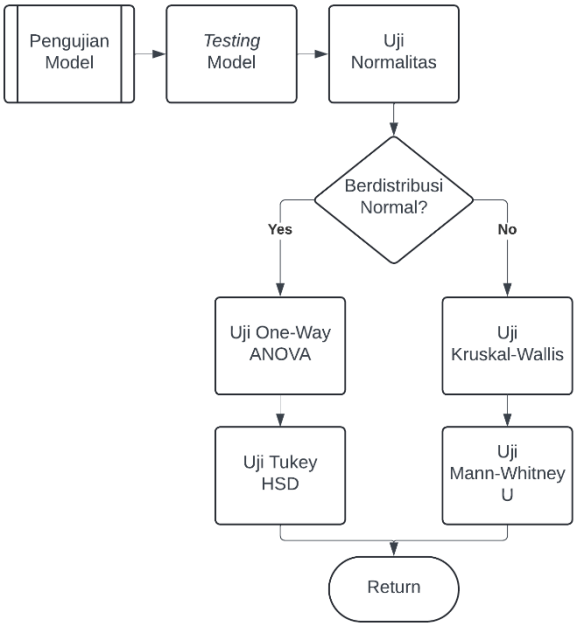
Pada gambar 3.6 (A) merupakan contoh citra fundus sebelum diterapkan augmentasi citra. Gambar 3.6 (B) merupakan hasil *horizontal flipping* dari gambar

3.6 (A). Gambar 3.6 (C) merupakan hasil *vertical flipping* dari gambar 3.6 (A). Gambar 3.6 (D) merupakan hasil *horizontal* dan *vertical flipping* dari gambar 3.6 (A).

3.3.4 Pelatihan Model

Setelah tercipta dataset hasil *preprocessing* dan augmentasi citra, hal yang dilakukan selanjutnya adalah melatih arsitektur untuk klasifikasi penyakit diabetik retinopati. Arsitektur *deep learning* yang digunakan dalam penelitian tugas akhir ini adalah Inception-v4, ResNet-50, VGG-19, dan YOLO v5 Nano. Ukuran citra *input* yang digunakan adalah 224x224. Seluruh arsitektur dilatih dengan menggunakan dataset setelah *preprocessing* dan augmentasi citra. Dataset tersebut akan dibagi menjadi data *train*, *test*, dan *validation* dengan rasio 70%, 20%, dan 10%. Hasil dari proses pelatihan ini adalah empat buah model, *confusion matrix*, dan nilai *accuracy* yang akan dievaluasi untuk menentukan arsitektur terbaik.

3.3.5 Pengujian Model



Gambar 3.7 *Flowchart* Pengujian Model

Setelah proses pelatihan model selesai, akan dilakukan pengujian model yang sudah terbentuk. Pada penelitian ini, digunakan *confusion matrix* untuk melakukan evaluasi model. *Confusion matrix* yang digunakan berukuran 5x5 dengan label 0 (mata normal), 1 (RD ringan), 2 (RD sedang), 3 (RD berat), dan 4 (RD proliferatif). Jumlah model yang diuji berjumlah empat, yaitu hasil empat arsitektur untuk dataset setelah *preprocessing* dan augmentasi citra. Masing-masing model akan dilakukan *testing* dengan data *testing* masing-masing sebanyak lima kali. Berikut contoh *confusion matrix* yang akan dihasilkan dari model:

Tabel 3.1 Contoh *Confusion Matrix Multiclass*

		Kelas Prediksi				
		0	1	2	3	4
Kelas Asli	0	A	B	C	D	E
	1	F	G	H	I	J
	2	K	L	M	N	O
	3	P	Q	R	S	T
	4	U	V	W	X	Y

Selain *confusion matrix*, model juga akan dievaluasi dengan menggunakan nilai tes *accuracy*. Nilai tes akurasi akan diuji normalitas untuk menentukan apakah data berdistribusi normal atau tidak. Hal ini perlu dilakukan untuk mengetahui uji statistik yang akan digunakan. Apabila data berdistribusi normal, maka akan menggunakan uji ANOVA untuk mengetahui apakah terdapat perbedaan nilai akurasi tes yang signifikan antara masing-masing model. Jika terdapat perbedaan signifikan, dilakukan uji Tukey HSD untuk mencari pasangan model yang terdapat perbedaan signifikan nilai akurasi tes.

Apabila data tidak berdistribusi normal, maka akan menggunakan uji Kruskal-Wallis untuk mengetahui apakah terdapat perbedaan nilai akurasi tes yang signifikan antara masing-masing model. Jika terdapat perbedaan signifikan, dilakukan uji Mann-Whitney U untuk mencari pasangan model yang terdapat perbedaan signifikan nilai akurasi tes. Pengujian model ini dilakukan untuk mencari arsitektur terbaik untuk mengklasifikasi penyakit RD.

3.3.5.1 *Testing* Model

Model yang merupakan hasil dari pelatihan model akan disimpan untuk dilakukan *testing* model. Data testing dibuat dengan cara menggunakan *train test split* pada dataset yang digunakan. Setelah itu model akan dites dengan data *test* yang telah dibuat. *Testing* model ini dilakukan sebanyak lima kali. *Random state* yang digunakan pada *train test split* akan berbeda-beda pada setiap tes. Hal ini dilakukan untuk menciptakan data tes yang berbeda-beda.

3.3.5.2 Uji Normalitas

Uji normalitas yang digunakan untuk penelitian ini adalah uji Shapiro-Wilk. Untuk melakukan uji Shapiro-Wilk, diperlukan hipotesis yang akan digunakan sebagai panduan untuk pengambilan kesimpulan. Berikut merupakan contoh hipotesis nol dan hipotesis alternatif yang akan digunakan dalam uji Shapiro-Wilk:

H0: data berasal dari populasi yang berdistribusi normal.
H1: data berasal dari populasi yang tidak berdistribusi normal.

3.3.5.3 Uji ANOVA

Apabila data berdistribusi normal, maka uji statistik yang digunakan adalah uji ANOVA. Untuk melakukan uji ANOVA, diperlukan hipotesis yang akan digunakan sebagai panduan untuk pengambilan kesimpulan. Berikut merupakan contoh hipotesis nol dan hipotesis alternatif yang akan digunakan dalam uji ANOVA:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model-model.
H1: terdapat perbedaan signifikan antara akurasi tes model-model.

3.3.5.4 Uji Tukey HSD

Jika terdapat perbedaan signifikan pada tahap uji ANOVA, maka akan dilanjutkan dengan melakukan uji Tukey HSD untuk mencari pasangan model yang

terdapat perbedaan signifikan nilai akurasi tes. Berikut merupakan contoh hipotesis nol dan hipotesis alternatif yang akan digunakan dalam uji Tukey HSD:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model A dengan model B.
H1: terdapat perbedaan signifikan antara akurasi tes model A dengan model B.

3.3.5.5 Uji Kruskal-Wallis

Apabila data tidak berdistribusi normal, maka uji statistik yang digunakan adalah uji Kruskal-Wallis. Untuk melakukan uji Kruskal-Wallis, diperlukan hipotesis yang akan digunakan sebagai panduan untuk pengambilan kesimpulan. Berikut merupakan contoh hipotesis nol dan hipotesis alternatif yang akan digunakan dalam uji Kruskal-Wallis:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model-model.
H1: terdapat perbedaan signifikan antara akurasi tes model-model.

3.3.5.6 Uji Mann-Whitney U

Jika terdapat perbedaan signifikan pada tahap uji Kruskal-Wallis, maka akan dilanjutkan dengan melakukan uji Mann-Whitney U untuk mencari pasangan model yang terdapat perbedaan signifikan nilai akurasi tes. Berikut merupakan contoh hipotesis nol dan hipotesis alternatif yang akan digunakan dalam uji Mann-Whitney U:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model A dengan model B.
H1: terdapat perbedaan signifikan antara akurasi tes model A dengan model B.

3.3.6 Pelatihan Model Analisis *Preprocessing* dan Augmentasi Data

Setelah menentukan arsitektur terbaik, arsitektur tersebut akan dilatih menggunakan dataset asli, dataset *cropping*, dataset histeq, dataset *flipping*, dataset

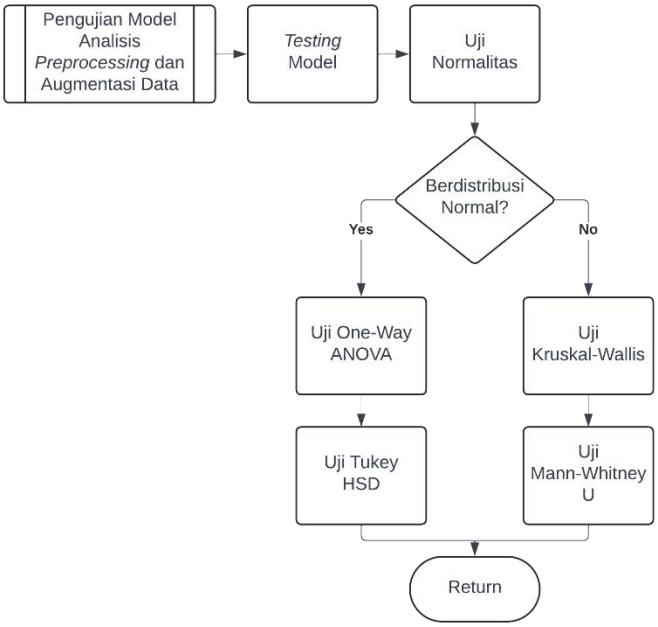
cropping dan histeq, dataset *cropping* dan *flipping*, dataset histeq dan *flipping*, dan dataset setelah *preprocessing* dan augmentasi data. Berikut adalah rincian dataset tersebut.

Tabel 3.2 Rincian Dataset RD Untuk Analisis Pengaruh *Preprocessing* dan Augmentasi Data

Nama Dataset	Train	Test	Validation	Total
Asli	1750	500	250	2.500
Cropping	1750	500	250	2.500
Histeq	1750	500	250	2.500
Flipping	7000	2000	1000	10.000
Cropping dan Histeq	1750	500	250	2.500
Cropping dan *Flipping*	7000	2000	1000	10.000
Histeq dan *Flipping*	7000	2000	1000	10.000
Setelah *Preprocessing* dan Augmentasi	7000	2000	1000	10.000

Seluruh dataset pada tabel 3.2 akan dibagi menjadi data *train*, *test*, dan *validation* dengan rasio 70%, 20%, dan 10%. Hasil dari proses pelatihan ini adalah delapan buah model, *confusion matrix*, dan nilai *accuracy* yang akan dievaluasi untuk menentukan *preprocessing* atau augmentasi data yang memiliki pengaruh paling signifikan terhadap performa arsitektur.

3.3.7 Pengujian Model Analisis *Preprocessing* dan Augmentasi Data



Gambar 3.8 *Flowchart* Pengujian Model Analisis *Preprocessing* dan Augmentasi Data

Setelah proses pelatihan model analisis *preprocessing* dan augmentasi data selesai, akan dilakukan pengujian terhadap model yang sudah terbentuk. Pada tahap ini, *confusion matrix* yang digunakan untuk melakukan evaluasi model sesuai dengan Tabel 3.1. Jumlah model yang diuji berjumlah delapan, yaitu model untuk dataset asli, dataset *cropping*, dataset histeq, dataset *flipping*, dataset *cropping* dan histeq, dataset *cropping* dan *flipping*, dataset histeq dan *flipping*, dan dataset setelah *preprocessing* dan augmentasi citra. Masing-masing model akan dilakukan *testing* dengan data *testing* masing-masing sebanyak lima kali.

3.3.7.1 *Testing* Model

Model yang merupakan hasil dari pelatihan model akan disimpan untuk dilakukan *testing* model. Data testing dibuat dengan cara menggunakan *train test split* pada dataset yang digunakan. Setelah itu model akan dites dengan data *test* yang telah dibuat. *Testing* model ini dilakukan sebanyak lima kali. *Random state*

yang digunakan pada *train test split* akan berbeda-beda pada setiap tes. Hal ini dilakukan untuk menciptakan data tes yang berbeda-beda.

3.3.7.2 Uji Normalitas

Uji normalitas yang digunakan untuk penelitian ini adalah uji Shapiro-Wilk. Untuk melakukan uji Shapiro-Wilk, diperlukan hipotesis yang akan digunakan sebagai panduan untuk pengambilan kesimpulan. Berikut merupakan contoh hipotesis nol dan hipotesis alternatif yang akan digunakan dalam uji Shapiro-Wilk:

H0: data berasal dari populasi yang berdistribusi normal.
H1: data berasal dari populasi yang tidak berdistribusi normal.

3.3.7.3 Uji ANOVA

Apabila data berdistribusi normal, maka uji statistik yang digunakan adalah uji ANOVA. Untuk melakukan uji ANOVA, diperlukan hipotesis yang akan digunakan sebagai panduan untuk pengambilan kesimpulan. Berikut merupakan contoh hipotesis nol dan hipotesis alternatif yang akan digunakan dalam uji ANOVA:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model-model.
H1: terdapat perbedaan signifikan antara akurasi tes model-model.

3.3.7.4 Uji Tukey HSD

Jika terdapat perbedaan signifikan pada tahap uji ANOVA, maka akan dilanjutkan dengan melakukan uji Tukey HSD untuk mencari pasangan model yang terdapat perbedaan signifikan nilai akurasi tes. Berikut merupakan contoh hipotesis nol dan hipotesis alternatif yang akan digunakan dalam uji Tukey HSD:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model A dengan model B.
H1: terdapat perbedaan signifikan antara akurasi tes model A dengan model B.

3.3.7.5 Uji Kruskal-Wallis

Apabila data tidak berdistribusi normal, maka uji statistik yang digunakan adalah uji Kruskal-Wallis. Untuk melakukan uji Kruskal-Wallis, diperlukan hipotesis yang akan digunakan sebagai panduan untuk pengambilan kesimpulan. Berikut merupakan contoh hipotesis nol dan hipotesis alternatif yang akan digunakan dalam uji Kruskal-Wallis:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model-model.
H1: terdapat perbedaan signifikan antara akurasi tes model-model.

3.3.7.6 Uji Mann-Whitney U

Jika terdapat perbedaan signifikan pada tahap uji Kruskal-Wallis, maka akan dilanjutkan dengan melakukan uji Mann-Whitney U untuk mencari pasangan model yang terdapat perbedaan signifikan nilai akurasi tes. Berikut merupakan contoh hipotesis nol dan hipotesis alternatif yang akan digunakan dalam uji Mann-Whitney U:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model A dengan model B.
H1: terdapat perbedaan signifikan antara akurasi tes model A dengan model B.

3.4 Interpretasi Hasil Penelitian

Tahapan terakhir setelah analisis *preprocessing* dan augmentasi data adalah pengambilan kesimpulan atau interpretasi hasil penelitian. Tahap interpretasi hasil penelitian akan menggunakan uji hipotesis yang telah dilakukan pada tahap evaluasi model. Pengambilan kesimpulan akan berdasar pada rumusan masalah yang telah ditetapkan, dalam hal ini pencarian arsitektur CNN yang optimal untuk mengklasifikasi penyakit retinopati diabetik dan pencarian *preprocessing* citra dan augmentasi data terbaik yang mampu meningkatkan akurasi model dalam mengklasifikasi penyakit retinopati diabetik.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Pelatihan Model

Seperti yang dijelaskan pada bab sebelumnya, penelitian ini diawali dengan mencari arsitektur terbaik untuk mengklasifikasi penyakit RD. Arsitektur ResNet-50, Inception-v4, VGG-19, dan YOLO v5Nano akan dilatih menggunakan dataset setelah *preprocessing* dan augmentasi. Masing-masing model hasil *training* akan disimpan dan diuji sebanyak lima kali. Pengujian model akan dilakukan terhadap model yang telah disimpan dan menggunakan dataset yang sama. Untuk masing-masing pengujian, dataset tersebut akan diterapkan *train test split* untuk menciptakan dataset *train* dan dataset *test*. Masing-masing *train test split* akan menggunakan *random state* yang berbeda untuk menciptakan dataset *test* yang berbeda-beda.

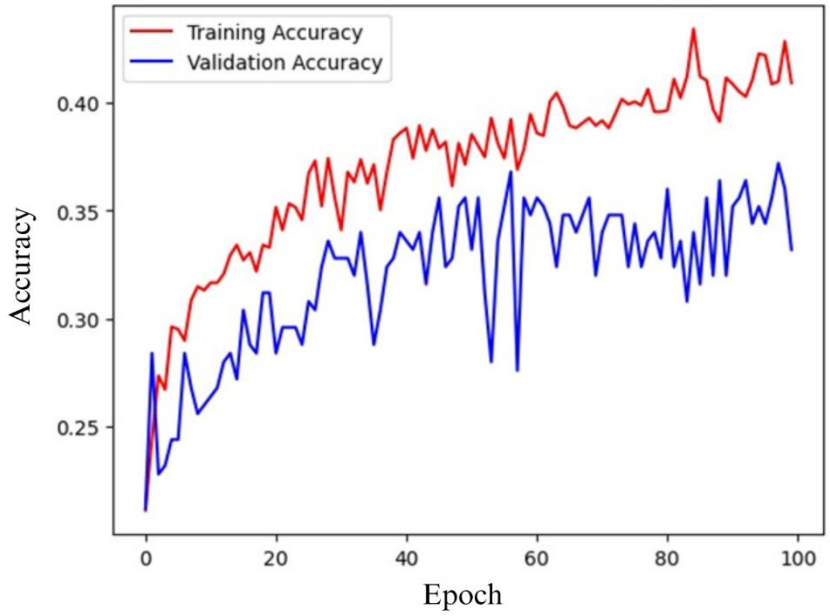
Tabel 4.1 Hasil *Training* dan *Testing* Model

Arsitektur	Akurasi Training	Akurasi Tes				
		Tes 1	Tes 2	Tes 3	Tes 4	Tes 5
ResNet-50	0.41	0.41	0.41	0.41	0.42	0.41
Inception-v4	0.21	0.45	0.44	0.44	0.45	0.44
VGG-19	0.82	0.72	0.22	0.21	0.19	0.19
YOLO v5Nano	0.93	0.74	0.71	0.73	0.73	0.73

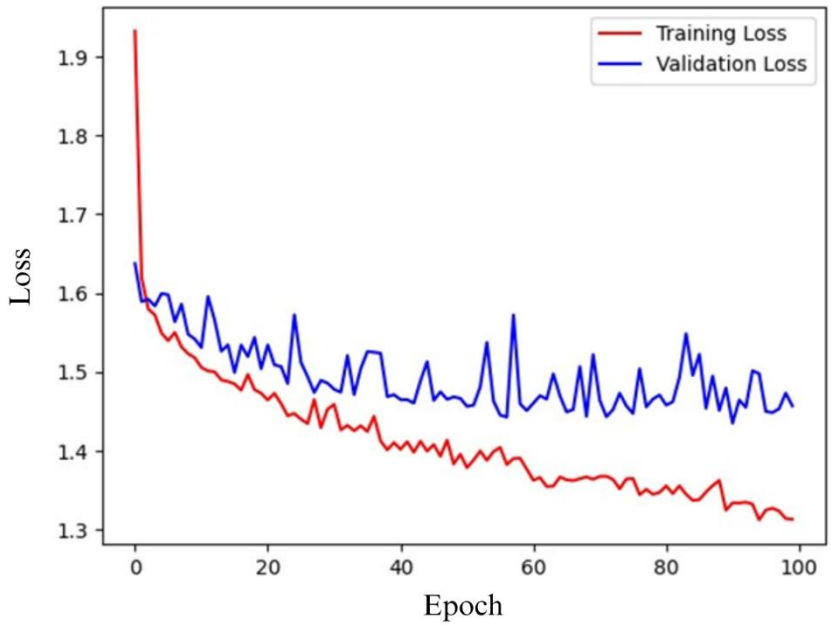
Tabel 4.1 merupakan hasil akurasi *training* dan *testing* masing-masing model. Akurasi hasil tes masing-masing model akan diuji dengan uji ANOVA dan dilanjutkan dengan uji Tukey HSD untuk menentukan arsitektur terbaik dalam mengklasifikasi penyakit RD.

4.1.1 Hasil Pelatihan ResNet-50

Model ResNet-50 tidak menunjukkan adanya *overfitting* maupun *underfitting* pada hasil *testing*. Nilai akurasi *training* dan *testing* dari model ResNet-50 berkisar pada angka 0.41. Berikut adalah grafik *loss* dan akurasi hasil *training* dari model ResNet-50.

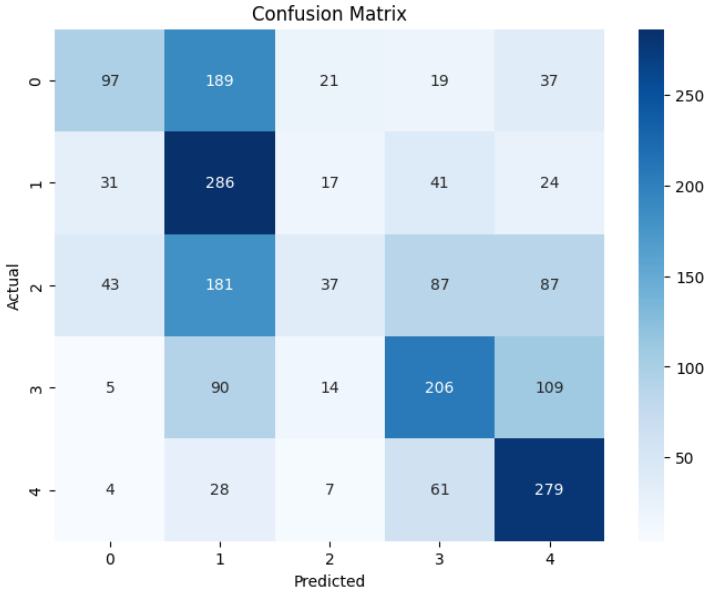


Gambar 4.1 Grafik Akurasi *Training* dan Validasi ResNet-50



Gambar 4.2 Grafik *Loss Training* dan Validasi ResNet-50

Pada gambar 4.1 dapat dilihat terjadi peningkatan akurasi dari model ResNet-50, walaupun terdapat fluktuasi didalamnya. Nilai akurasi *training* model ResNet-50 adalah 0.41, sedangkan nilai akurasi validasi adalah 0.33. Pada gambar 4.2 juga dapat dilihat bahwa terdapat fluktuasi di penurunan *loss* dari model ResNet-50. Nilai *loss training* model ResNet-50 adalah 1.31, sedangkan *loss* validasi adalah 1.4566. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari *testing* model ResNet-50.

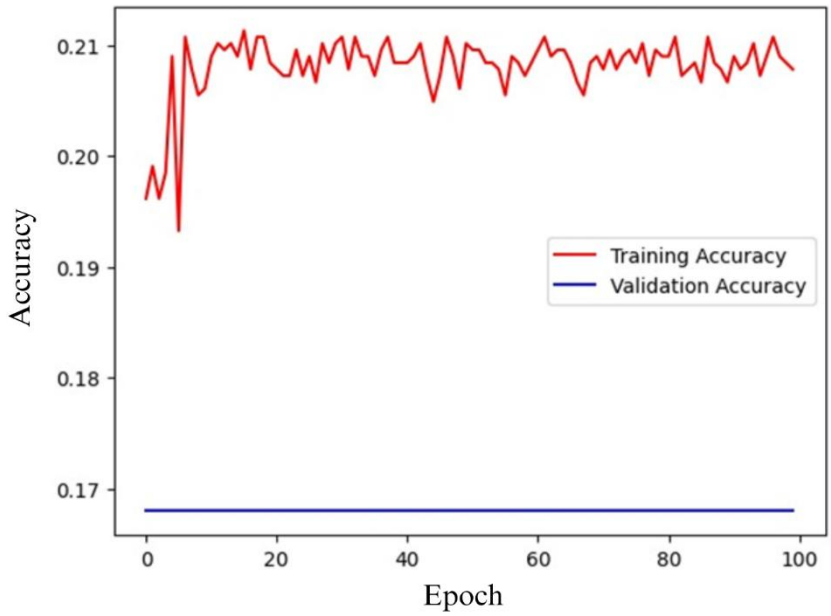


Gambar 4.3 *Confusion Matrix Testing* ResNet-50

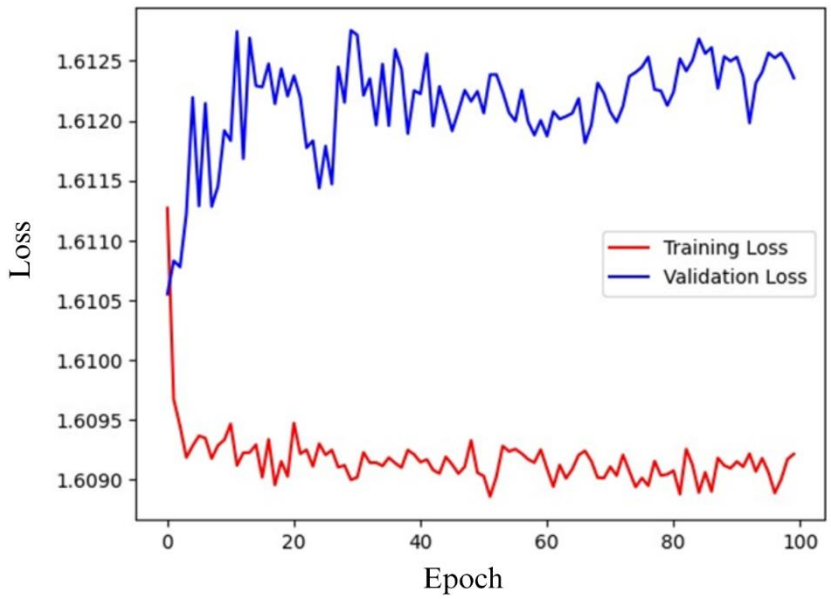
Berdasarkan gambar 4.3, model ResNet-50 berhasil mengklasifikasikan RD dengan akurasi 0.41. Model ini masih belum dapat mengklasifikasi kelas 0 dan kelas 2 dengan baik, dimana kelas-kelas ini diprediksi menjadi kelas 1. Untuk kelas 3 mampu diklasifikasi oleh model dengan lebih baik. Hal ini dapat dilihat dari jumlah TP dari kelas 3 mencapai setengah dari total kelas. Sedangkan untuk kelas 1 dan 4 dapat diklasifikasi dengan baik oleh model ResNet-50. Hal ini ditunjukkan dengan angka TP paling tinggi terdapat pada kelas 1 dan 4.

4.1.2 Hasil Pelatihan Inception-v4

Model Inception-v4 menunjukkan adanya *underfitting*. Nilai akurasi *training* jauh lebih kecil dibandingkan dengan nilai akurasi *testing*. Nilai akurasi *training* model Inception-v4 hanya mencapai 0.21, sedangkan nilai akurasi *testing* model Inception-v4 rata-rata mencapai 0.44. Berikut adalah grafik *loss* dan akurasi hasil *training* dari model Inception-v4.

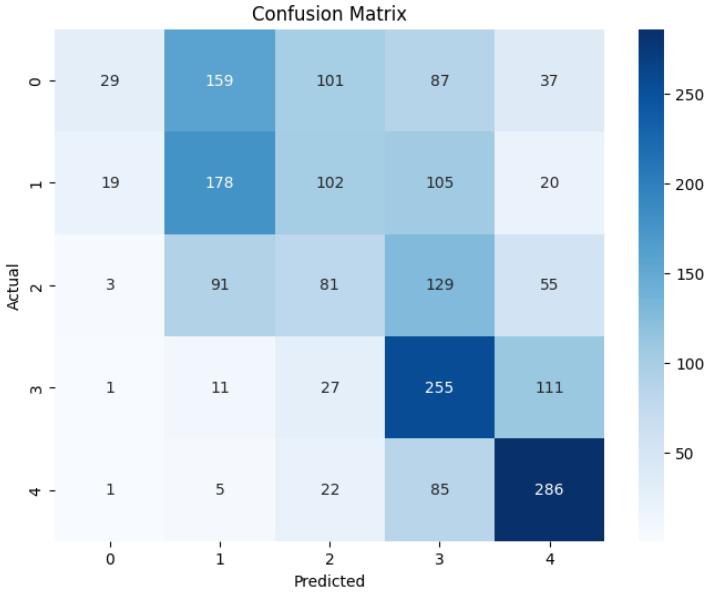


Gambar 4.4 Grafik Akurasi *Training* dan Validasi Inception-v4



Gambar 4.5 Grafik *Loss Training* dan Validasi Inception-v4

Pada gambar 4.4 terlihat terjadi peningkatan nilai akurasi *training* dari model Inception-v4, namun peningkatan nilai tersebut hanya berubah sedikit saja. Sedangkan nilai akurasi validasi tidak mengalami perubahan. Nilai akurasi *training* model Inception-v4 adalah 0.21, sedangkan nilai akurasi validasi adalah 0.17. Pada gambar 4.5 juga terlihat bahwa terdapat penurunan *loss training* dan terjadi peningkatan *loss validation* dari model Inception-v4, namun nilai tersebut hanya berubah sedikit saja. Nilai *loss training* model Inception-v4 adalah 1.61, sedangkan nilai *loss* validasi adalah 1.61. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari *testing* model Inception-v4.

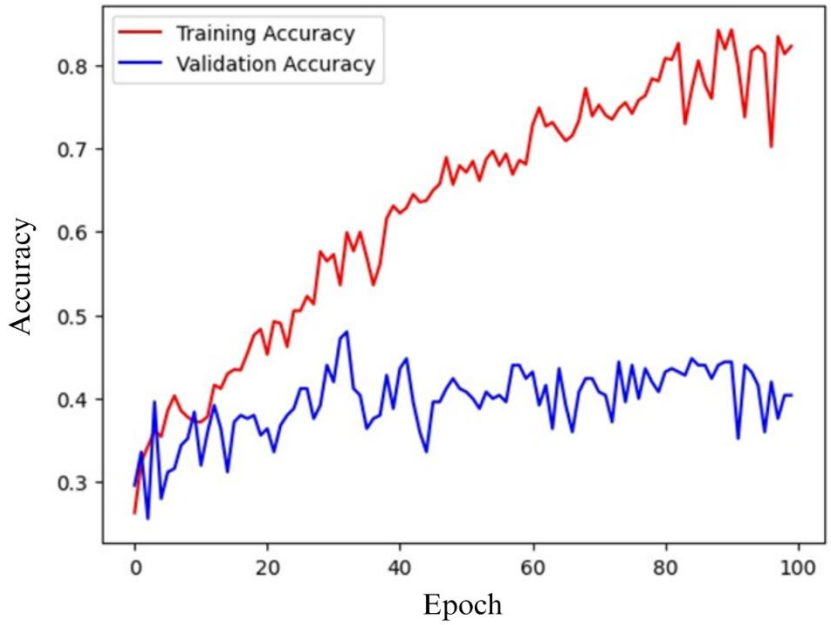


Gambar 4.6 *Confusion Matrix Testing* Inception-v4

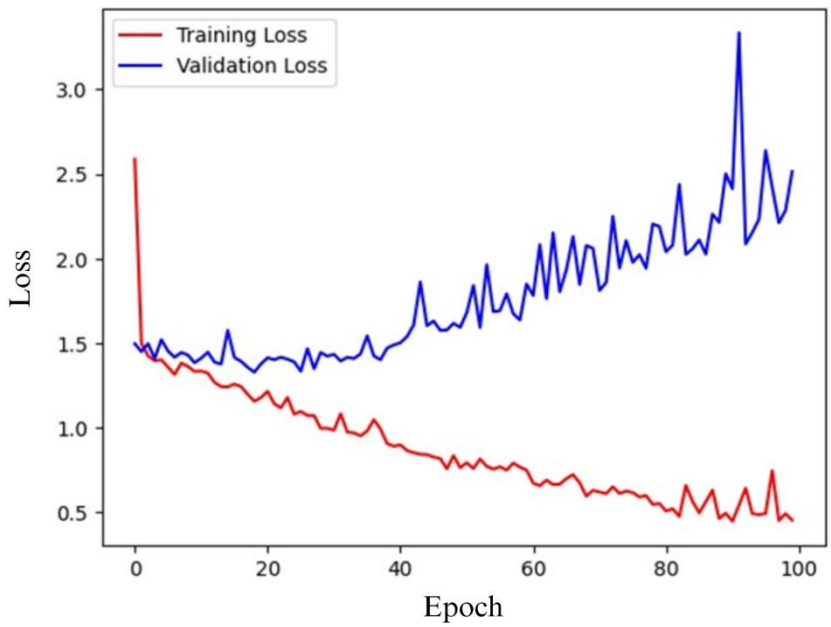
Berdasarkan gambar 4.6, model Inception-v4 berhasil mengklasifikasikan RD dengan akurasi 0.45. Model ini masih belum dapat mengklasifikasi kelas 0 dan 2 dengan baik. Hal ini dapat dilihat dengan nilai TP dari kelas 0 dan 2 kecil. Untuk kelas 1 dapat diklasifikasi oleh model dengan lebih baik. Hal ini dapat dilihat dari nilai TP dari kelas 1 cukup tinggi. Sedangkan untuk kelas 3 dan 4 dapat diklasifikasi dengan baik oleh model Inception-v4. Hal ini ditunjukkan dengan angka TP paling tinggi terdapat pada kelas 3 dan 4.

4.1.3 Hasil Pelatihan VGG-19

Model VGG-19 menunjukkan adanya *overfitting*. Nilai akurasi *training* jauh lebih tinggi dibandingkan dengan nilai akurasi *testing*. Nilai akurasi *training* model VGG-19 mencapai 0.82, sedangkan nilai akurasi *testing* model VGG-19 banyak yang hanya mencapai 0.20. Berikut adalah grafik *loss* dan *accuracy* hasil *training* dari model VGG-19.

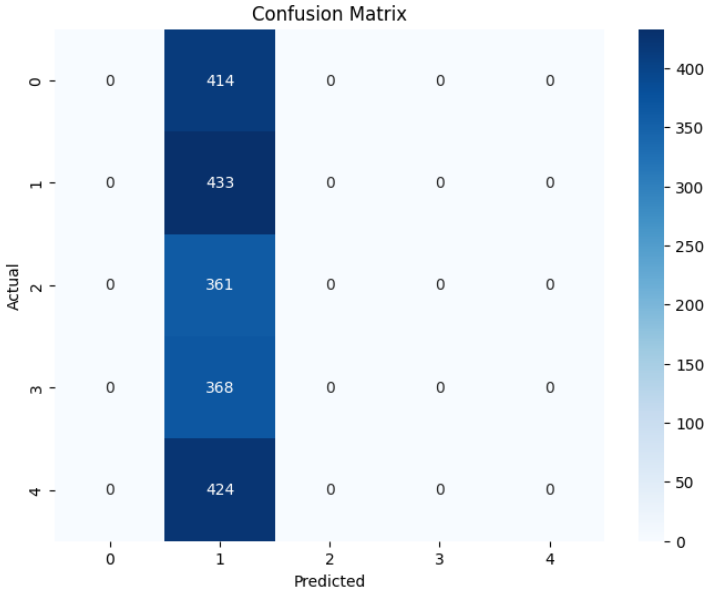


Gambar 4.7 Grafik Akurasi *Training* dan Validasi VGG-19



Gambar 4.8 Grafik *Loss Training* dan Validasi VGG-19

Pada gambar 4.7 dapat dilihat terjadi peningkatan akurasi *training* yang signifikan dari model VGG-19, namun hal tersebut tidak terjadi pada akurasi validasi. Nilai akurasi *training* model VGG-19 adalah 0.82, sedangkan nilai akurasi validasi hanya mencapai 0.40. Pada gambar 4.8 juga dapat dilihat bahwa terdapat penurunan *loss training* yang signifikan dan terjadi peningkatan *loss* validasi yang signifikan dari model VGG-19. Nilai *loss training* model VGG-19 adalah 0.45, sedangkan *loss* validasi adalah 2.51. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari *testing* model VGG-19.

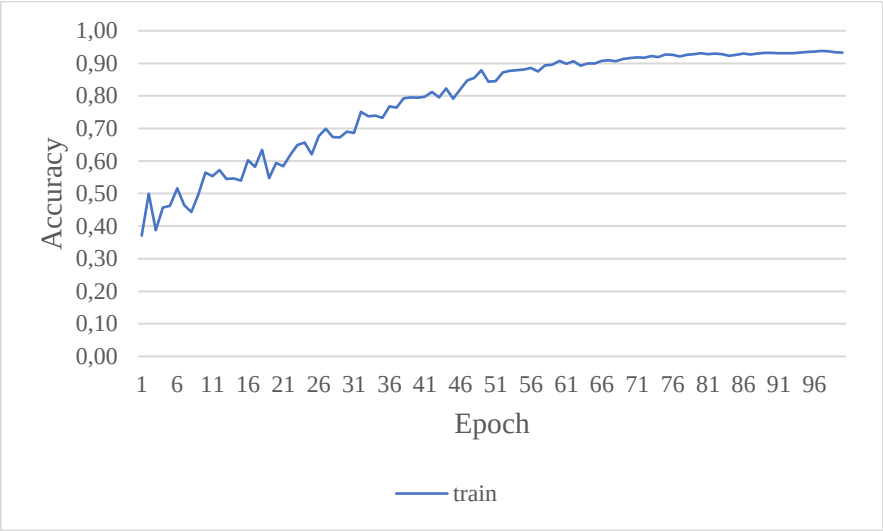


Gambar 4.9 *Confusion Matrix Testing* VGG-19

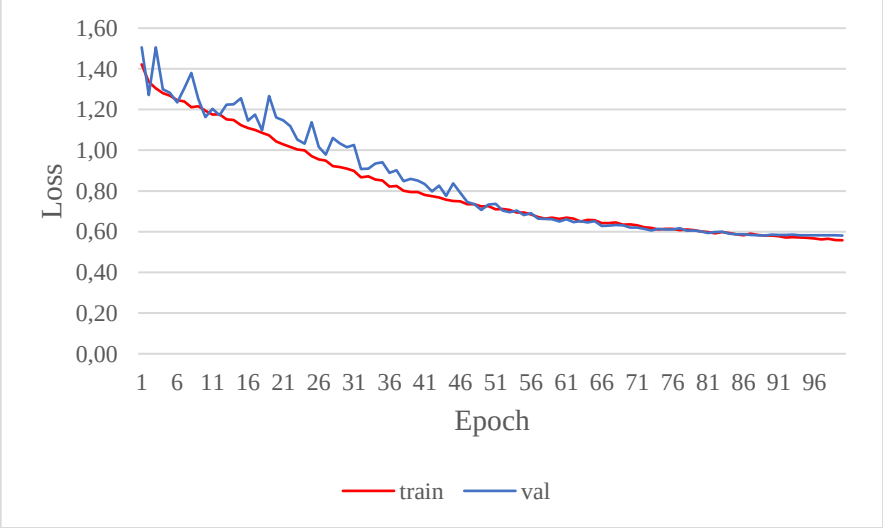
Berdasarkan gambar 4.9, model VGG-19 belum dapat mengklasifikasikan penyakit RD dengan benar dan terjadi *overfitting*. Hal ini dapat dilihat dari hasil *confusion matrix* dimana *predicted label* seluruhnya mengarah ke kelas 1, sedangkan kelas-kelas lainnya masih tidak dapat diklasifikasi. *Overfitting* dapat terjadi karena model terlalu cocok dengan data pelatihan atau kompleksitas model VGG-19 tidak sesuai.

4.1.4 Hasil Pelatihan YOLO v5Nano

Model YOLO v5Nano menunjukkan adanya *overfitting*. Nilai akurasi *training* lebih tinggi dibandingkan dengan akurasi *testing*. Nilai akurasi *training* model YOLO v5Nano mencapai 0.93, sedangkan nilai akurasi *testing* model YOLO v5Nano rata-rata hanya mencapai 0.73. Berikut adalah grafik *loss* dan *accuracy* dari *training* model YOLO v5Nano.



Gambar 4.10 Grafik Akurasi *Training* dan Validasi YOLO v5Nano



Gambar 4.11 Grafik *Loss Training* dan Validasi YOLO v5Nano

Pada gambar 4.10 dapat dilihat model YOLO v5Nano mengalami peningkatan akurasi *training* yang signifikan. Nilai akurasi *training* model ini

mencapai 0.93. Pada gambar 4.11 juga dapat dilihat model YOLO v5Nano mengalami penurunan loss yang signifikan. Nilai *loss training* model YOLO v5Nano adalah 0.56, sedangkan nilai *loss* validasi adalah 0.58. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari model YOLO v5Nano.

		Predicted				
		0	1	2	3	4
Actual	0	368	20	10	2	2
	1	23	375	8	2	2
	2	13	12	345	4	11
	3	3	7	9	370	8
	4	0	2	3	12	389

Gambar 4.12 *Confusion Matrix* YOLO v5Nano

Berdasarkan gambar 4.12, model YOLO v5Nano berhasil mengklasifikasikan penyakit RD dengan akurasi 0.924. Model YOLO v5Nano dapat mengklasifikasikan seluruh kelas RD dengan baik. Hal ini dapat dilihat dari hasil *confusion matrix* dimana *predicted label* masing-masing kelas hampir seluruhnya telah sesuai dengan *actual label*.

4.2 Hasil Pengujian Model

Empat model hasil pelatihan model dengan dataset setelah *preprocessing* dan augmentasi data akan diuji untuk menentukan arsitektur terbaik untuk mengklasifikasi penyakit RD. Dalam tahap pengujian model, digunakan uji normalitas dan uji statistik. Uji normalitas yang digunakan pada tahap ini adalah uji Shapiro-Wilk. Uji statistik yang digunakan pada tahap ini adalah uji ANOVA dan uji Tukey HSD atau uji Kruskal-Wallis dan uji Mann-Whitney U. Berikut adalah hasil dari uji normalitas dan uji statistik dari empat model tersebut.

4.2.1 Hasil Uji Shapiro-Wilk Pengujian Model

Pada tahap ini menggunakan uji Shapiro-Wilk untuk menentukan apakah sampel akurasi tes berdistribusi normal atau tidak. Nilai alpha yang ditentukan untuk uji Sapiro-Wilk adalah 0.05. Berikut adalah hipotesis nol dan hipotesis alternatif yang digunakan dalam uji Shapiro-Wilk:

H0: data berdistribusi normal.
H1: data tidak berdistribusi normal.

Contoh perhitungan uji Shapiro-Wilk pada VGG-19:

Data VGG-19:

$$X_{(1)} = 0.1875, X_{(2)} = 0.1915, X_{(3)} = 0.2135, X_{(4)} = 0.2165, X_{(5)} = 0.72$$

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.1875 + 0.1915 + 0.2135 + 0.2165 + 0.72}{5}$$
$$= 0.3058$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$
$$= 0.2151148$$

Nilai koefisien Shapiro-Wilk untuk $n = 5$:

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(n+1-i)} - X_i))^2}{\sum_{i=1}^{n} (X_i - \bar{X})^2}$$
$$= \frac{(0.3538995 + 0.0060325)^2}{0.2151148}$$
$$= 0.602$$

Karena nilai W lebih kecil dari nilai W untuk *p-value* 0.01 pada tabel nilai kritis uji Shapiro-Wilk, maka nilai *p-value* dianggap lebih rendah dari 0.01. Karena

p-value lebih kecil dibandingkan alpha, maka H0 ditolak, maka dapat disimpulkan sampel tidak terdistribusi normal.

Contoh perhitungan uji Shapiro-Wilk pada ResNet-50:

Data ResNet-50:

$$X_{(1)} = 0.407, X_{(2)} = 0.4085, X_{(3)} = 0.409, X_{(4)} = 0.41, X_{(5)} = 0.418$$

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.407 + 0.4085 + 0.409 + 0.41 + 0.418}{5}$$
$$= 0.4105$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2 = 0.000075$$

Nilai koefisien Shapiro-Wilk untuk $n = 5$:

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(n+1-i)} - X_i))^2}{\sum_{i=1}^{n} (X_i - \bar{X})^2}$$
$$= \frac{(0.0073106 + 0.00036195)^2}{0.000075}$$
$$= 0.784$$

Karena nilai W lebih besar dari nilai W untuk alpha 0.05, maka nilai *p-value* dianggap lebih tinggi dari 0.05. Karena nilai *p-value* lebih tinggi dibandingkan alpha, maka H0 diterima, maka dapat disimpulkan bahwa sampel terdistribusi normal. Berikut adalah tabel hasil uji Shapiro-Wilk dari empat model yang digunakan.

Tabel 4.2 Hasil Uji Shapiro-Wilk Pengujian Model

Arsitektur	*p-value*	Hipotesis
ResNet-50	0.06	H0 diterima
Inception-v4	0.98	H0 diterima
VGG-19	0.00	H0 ditolak
YOLO v5Nano	0.37	H0 diterima

Dari tabel 4.2, dapat dilihat bahwa terdapat data yang tidak terdistribusi normal, yaitu hasil tes VGG-19. Oleh karena itu, uji statistik yang digunakan pada tahap selanjutnya adalah uji Kruskal-Wallis.

4.2.2 Hasil Uji Kruskal-Wallis Pengujian Model

Pada tahap ini menggunakan uji Kruskal-Wallis untuk menentukan apakah terdapat perbedaan signifikan antara akurasi tes dari model-model tersebut. Nilai alpha yang ditentukan untuk uji Kruskal-Wallis adalah 0.05. Berikut adalah hipotesis nol dan hipotesis alternatif yang digunakan dalam uji Kruskal-Wallis:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model-model.

H1: terdapat perbedaan signifikan antara akurasi tes model-model.

Perhitungan uji Kruskal-Wallis Pengujian Model:

Data model:

ResNet-50: 0.407, 0.4085, 0.409, 0.41, 0.418

Inception-v4: 0.4355, 0.4395, 0.4435, 0.4465, 0.452

VGG-19: 0.1875, 0.1915, 0.2135, 0.2165, 0.72

YOLO v5Nano: 0.708, 0.727, 0.728, 0.729, 0.739

Peringkat Sampel Model:

ResNet-50: 8, 7, 5, 9, 6

Inception-v4: 14, 12, 11, 13, 10

VGG-19: 16, 4, 3, 2, 1

YOLO v5Nano: 20, 15, 19, 18, 17

Total Peringkat masing-masing kelompok:

ResNet-50: 35

Inception-v4: 60

VGG-19: 26

YOLO v5Nano: 89

$$H = \frac{12}{20(20+1)} \sum_{i=1}^{k} \frac{{R_i}^2}{n_i} - 3(20+1)$$

$$H = \frac{12}{420} \times \left(\frac{35^2}{5} + \frac{60^2}{5} + \frac{26^2}{5} + \frac{89^2}{5} \right) - 63$$

$$H = 13.69$$

$$df = N - 1 = 4 - 1 = 3$$

Nilai *p-value* jika $df = 3$ adalah 0.0035. Karena nilai *p-value* lebih kecil dibandingkan nilai alpha, maka hipotesis nol ditolak. Oleh karena itu, dapat disimpulkan bahwa terdapat perbedaan signifikan antara akurasi tes model-model. Untuk mengetahui model-model yang terdapat perbedaan signifikan, dilakukan uji Mann-Whitney U.

4.2.3 Hasil Uji Mann-Whitney U Pengujian Model

Sebagai kelanjutan dari uji Kruskal-Wallis, digunakan uji Mann-Whitney U untuk mengetahui model yang terdapat perbedaan signifikan. Nilai alpha yang ditentukan untuk uji Mann-Whitney U adalah 0.05. Berikut adalah hipotesis nol dan hipotesis alternatif yang digunakan dalam uji Mann-Whitney U:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model A dengan model B.

H1: terdapat perbedaan signifikan antara akurasi tes model A dengan model B.

Perhitungan uji Mann-Whitney U Pengujian Model antara ResNet-50 dan VGG-19:

Data model:

ResNet-50: 0.407, 0.4085, 0.409, 0.41, 0.418

VGG-19: 0.1875, 0.1915, 0.2135, 0.2165, 0.72

Peringkat Sampel Model:

ResNet-50: 5, 6, 7, 8, 9

VGG-19: 1, 2, 3, 4, 10

Total Peringkat masing-masing kelompok:

ResNet-50: 35

VGG-19: 20

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 35 = 5$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 20 = 20$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 20 = 5$$

Nilai U jika menggunakan alpha 0.05 adalah 2. Karena nilai *p-value* lebih besar dibandingkan nilai alpha, maka hipotesis nol diterima. Oleh karena itu, dapat disimpulkan bahwa tidak terdapat perbedaan signifikan antara akurasi tes model ResNet-50 dengan model VGG-19.

Perhitungan uji Mann-Whitney U Pengujian Model antara ResNet-50 dan YOLO v5Nano:

Data model:

ResNet-50: 0.407, 0.4085, 0.409, 0.41, 0.418

YOLO v5Nano: 0.708, 0.727, 0.728, 0.729, 0.739

Peringkat Sampel Model:

ResNet-50: 1, 2, 3, 4, 5

YOLO v5Nano: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

ResNet-50: 15

YOLO v5Nano: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Nilai U jika menggunakan alpha 0.05 adalah 2. Karena nilai *p-value* lebih kecil dibandingkan nilai alpha, maka hipotesis nol ditolak. Oleh karena itu, dapat disimpulkan bahwa terdapat perbedaan signifikan antara akurasi tes model ResNet-50 dengan model YOLO v5Nano.

Tabel 4.3 Hasil Uji Mann-Whitney U Pengujian Model

Arsitektur 1	Arsitektur 2	*p-value*	Hipotesis
ResNet-50	Inception-v4	0.97	H0 ditolak
ResNet-50	VGG-19	0.50	H0 diterima
ResNet-50	YOLO v5Nano	0.00	H0 ditolak
Inception-v4	VGG-19	0.28	H0 diterima
Inception-v4	YOLO v5Nano	0.01	H0 Ditolak
VGG-19	YOLO v5Nano	0.00	H0 Ditolak

Dari tabel 4.2 dapat dilihat bahwa pada model ResNet-50, Inception-v4, dan VGG-19 terdapat perbedaan signifikan dari nilai akurasi tes dengan model YOLO v5Nano. Karena arsitektur YOLO v5Nano merupakan arsitektur terbaik untuk mengklasifikasi penyakit RD, maka pada tahap analisis *preprocessing* dan augmentasi akan menggunakan arsitektur YOLO v5Nano.

4.3 Hasil Pelatihan Model Analisis *Preprocessing* dan Augmentasi Data

Pada tahap analisis *preprocessing* dan augmentasi data, arsitektur terbaik yaitu YOLO v5Nano dilatih dengan delapan dataset untuk menentukan *preprocessing* atau augmentasi yang terbaik untuk meningkatkan performa model. Berikut adalah hasil training arsitektur YOLO v5Nano terhadap delapan dataset tersebut.

Tabel 4.4 Akurasi Tes Analisis *Preprocessing* dan Augmentasi Data

Model	Akurasi *Training*	Akurasi Tes				
		Tes 1	Tes 2	Tes 3	Tes 4	Tes 5
Asli	0.46	0.69	0.67	0.67	0.59	0.58
Cropping	0.46	0.57	0.54	0.54	0.54	0.55
Histeq	0.43	0.45	0.46	0.44	0.46	0.47
Flipping	0.91	0.91	0.94	0.94	0.94	0.93
Cropping dan Histeq	0.40	0.46	0.54	0.54	0.56	0.57

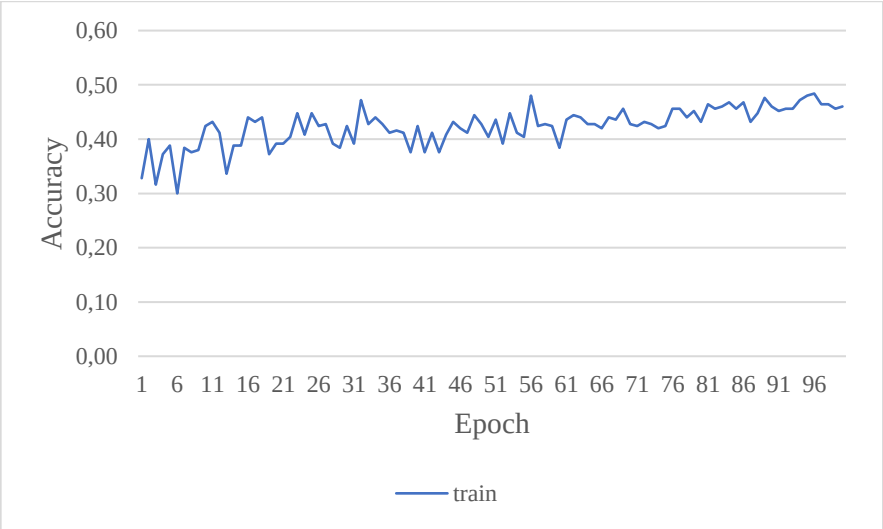
Tabel 4.4 Lanjutan

Cropping dan *Flipping*	0.93	0.82	0.81	0.81	0.81	0.82
Histeq dan *Flipping*	0.92	0.69	0.70	0.68	0.70	0.69
Setelah *Preprocessing* dan Augmentasi	0.93	0.74	0.71	0.73	0.73	0.73

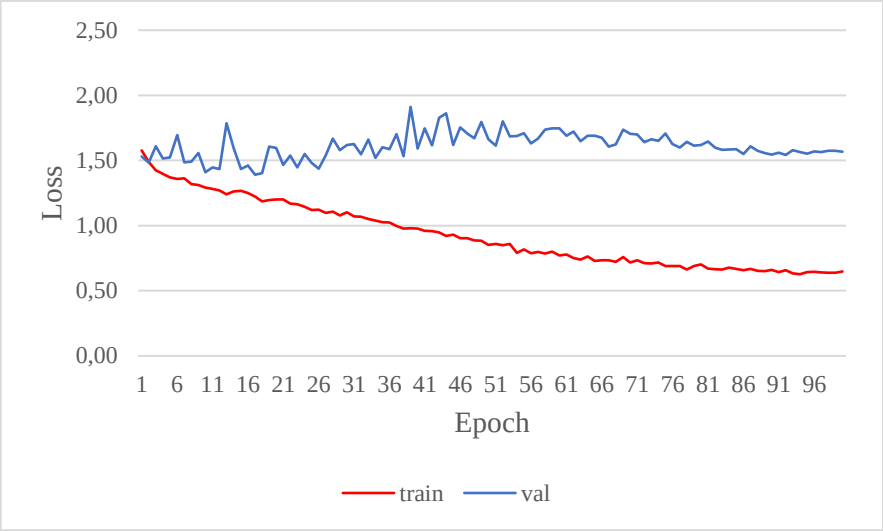
Tabel 4.3 merupakan hasil akurasi model dengan menggunakan dataset asli, *cropping*, histeq, *flipping*, *cropping* dan histeq, *cropping* dan *flipping*, histeq dan *flipping*, dan setelah *preprocessing* dan augmentasi data untuk menentukan *preprocessing* atau augmentasi data yang terbaik untuk meningkatkan performa model. Model hasil pelatihan akan disimpan dengan format h5 untuk dilakukan *test* model. Masing-masing model dites sebanyak lima kali.

4.3.1 Hasil Pelatihan Dataset Asli

Model dataset asli menunjukkan adanya *underfitting*. Nilai akurasi training jauh lebih rendah dibandingkan nilai akurasi *testing*. Nilai akurasi *training* model dataset asli hanya mencapai 0.46, sedangkan rata-rata nilai akurasi *testing* model dataset asli adalah 0.64. Berikut adalah grafik *loss* dan akurasi dari *training* model dataset asli.



Gambar 4.13 Grafik Akurasi *Training* Dataset Asli



Gambar 4.14 Grafik *Loss Training* dan Validasi Dataset Asli

Pada gambar 4.13 dapat dilihat grafik model dataset asli mengalami sedikit peningkatan akurasi *training* dari 0.33 menjadi 0.46. Pada gambar 4.14 juga dapat dilihat grafik model dataset asli mengalami penurunan *loss training* yang signifikan dari 1.58 menjadi 0.65, sedangkan nilai *loss* validasi mengalami sedikit peningkatan dari 1.53 menjadi 1.57. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari model dataset asli.

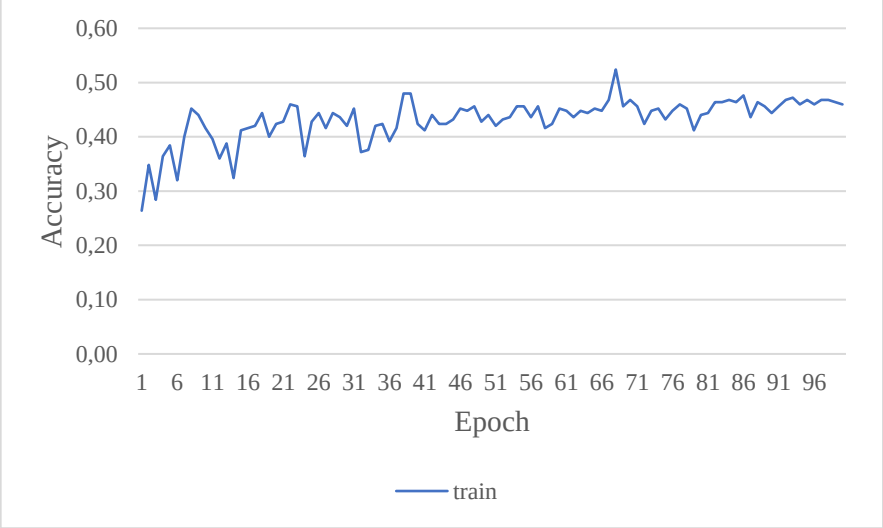
		Predicted				
		0	1	2	3	4
Actual	0	41	30	17	6	7
	1	23	46	26	2	3
	2	31	20	30	20	11
	3	1	10	16	59	12
	4	3	1	7	24	54

Gambar 4.15 *Confusion Matrix* Dataset Asli

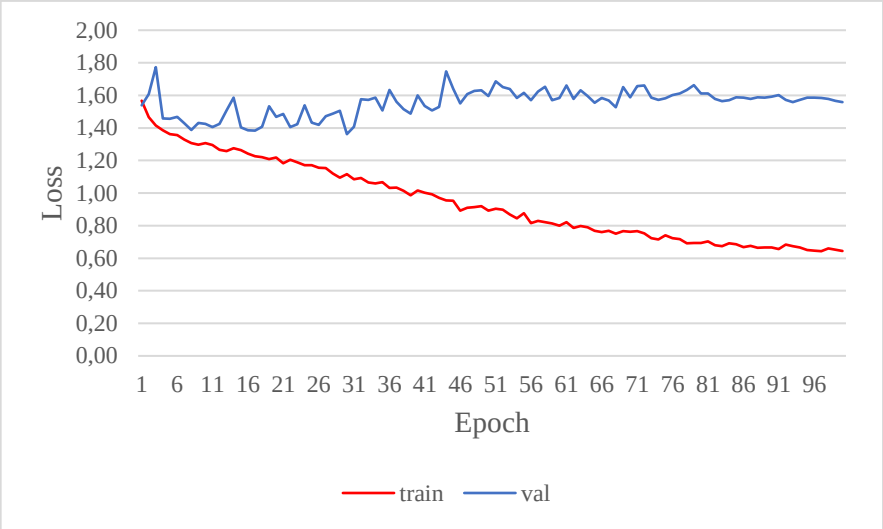
Berdasarkan gambar 4.15, model dataset asli berhasil mengklasifikasikan RD dengan akurasi 0.46. Model ini masih belum dapat mengklasifikasi kelas 0, 1, dan kelas 2 dengan baik, dimana prediksi kelas-kelas ini masih banyak yang tertukar satu sama lain. Untuk kelas 3 dan 4 mampu diklasifikasi oleh model dengan lebih baik. Hal ini dapat dilihat dari jumlah TP dari kelas 3 dan 4 lebih banyak dibandingkan kelas yang lain.

4.3.2 Hasil Pelatihan Dataset *Cropping*

Model dataset *cropping* menunjukkan adanya *underfitting*. Nilai akurasi training jauh lebih rendah dibandingkan nilai akurasi *testing*. Nilai akurasi *training* model dataset *cropping* hanya mencapai 0.46, sedangkan rata-rata nilai akurasi *testing* model dataset *cropping* adalah 0.55. Berikut adalah grafik *loss* dan akurasi dari *training* model dataset *cropping*.



Gambar 4.16 Grafik Akurasi *Training* Dataset *Cropping*



Gambar 4.17 Grafik *Loss Training* dan Validasi Dataset *Cropping*

Pada gambar 4.16 dapat dilihat grafik model dataset *cropping* mengalami sedikit peningkatan akurasi *training* dari 0.26 menjadi 0.46. Pada gambar 4.17 juga dapat dilihat grafik model dataset *cropping* mengalami penurunan *loss training* dari 1.57 menjadi 0.65, sedangkan nilai *loss* validasi hanya mengalami sedikit perubahan dari 1.54 menjadi 1.56. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari model dataset *cropping*.

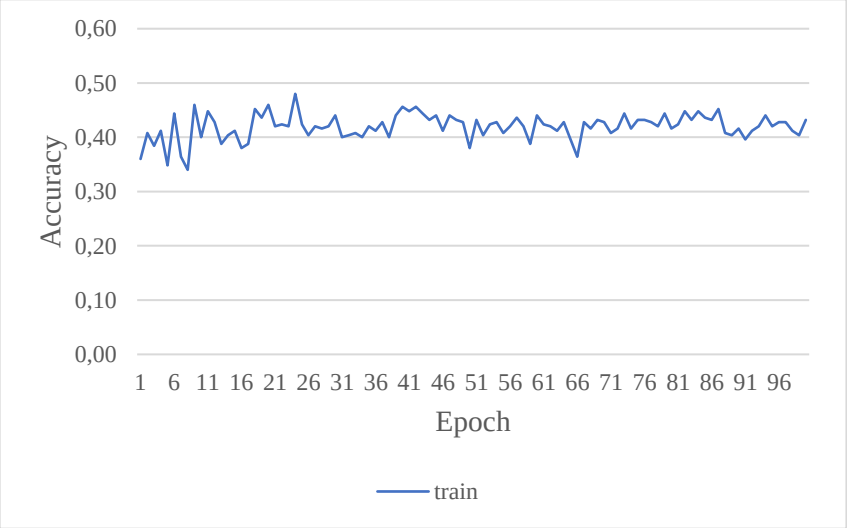
		Predicted				
		0	1	2	3	4
Actual	0	53	35	11	4	6
	1	26	41	21	3	5
	2	18	18	37	18	11
	3	5	5	17	48	9
	4	3	3	9	25	69

Gambar 4.18 *Confusion Matrix* Dataset *Cropping*

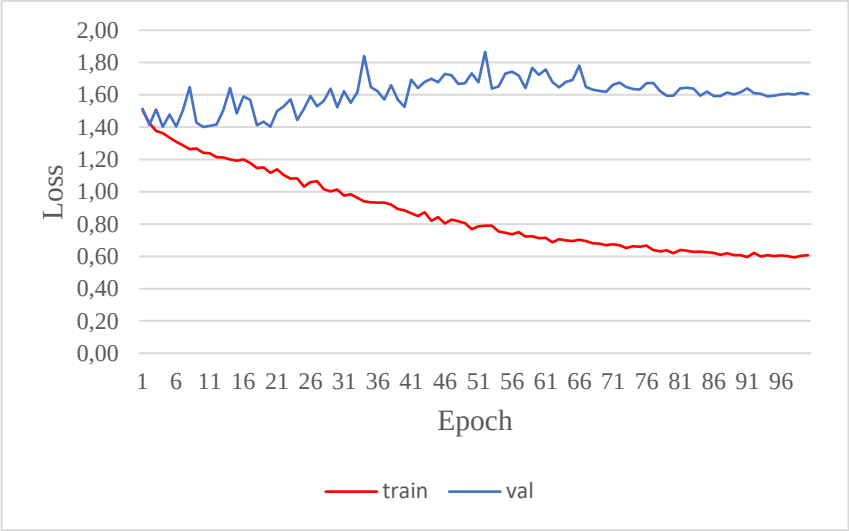
Berdasarkan gambar 4.18, model dataset *cropping* berhasil mengklasifikasikan RD dengan akurasi 0.49. Model ini masih belum dapat mengklasifikasi kelas 1 dan 2 dengan baik, dimana TP kelas ini tidak mencapai setengah dari total citra tes kelas tersebut. Untuk kelas 0 dan kelas 3 mampu diklasifikasi oleh model dengan lebih baik. Hal ini dapat dilihat dari jumlah TP dari kelas 0 dan kelas 3 mencapai setengah dari total kelas. Sedangkan untuk kelas 4 dapat diklasifikasi dengan baik oleh model dataset *cropping*. Hal ini ditunjukkan dengan angka TP paling tinggi terdapat pada kelas 4.

4.3.3 Hasil Pelatihan Dataset Histeq

Model dataset histeq tidak menunjukkan adanya *underfitting* atau *overfitting*. Hal ini dapat dilihat dari nilai akurasi *training* dan nilai akurasi *testing* yang tidak berbeda jauh. Nilai akurasi *training* model dataset histeq mencapai 0.43, sedangkan rata-rata nilai akurasi *testing* model dataset histeq mencapai 0.46. Berikut adalah grafik *loss* dan akurasi dari *training* model dataset histeq.



Gambar 4.19 Grafik Akurasi *Training* Dataset Histeq



Gambar 4.20 Grafik *Loss Training* dan Validasi Dataset Histeq

Pada gambar 4.19 dapat dilihat grafik model dataset histeq mengalami sedikit peningkatan akurasi *training* dari 0.36 menjadi 0.43. Pada gambar 4.20 juga dapat dilihat grafik model dataset histeq mengalami penurunan *loss training* yang signifikan dari 1.50 menjadi 0.61, sedangkan nilai *loss* validasi mengalami sedikit peningkatan dari 1.51 menjadi 1.60. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari model dataset histeq.

		Predicted				
		0	1	2	3	4
Actual	0	30	51	19	5	4
	1	22	53	19	6	1
	2	11	29	25	19	8
	3	4	4	18	59	12
	4	5	4	17	19	56

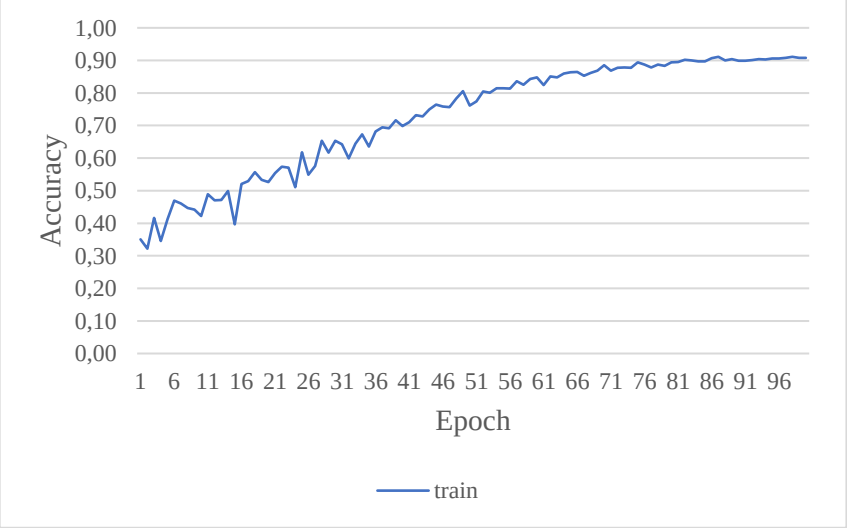
Gambar 4.21 *Confusion Matrix* Dataset Histeq

Berdasarkan gambar 4.18, model dataset histeq berhasil mengklasifikasikan RD dengan akurasi 0.45. Model ini masih belum dapat mengklasifikasi kelas 0 dan

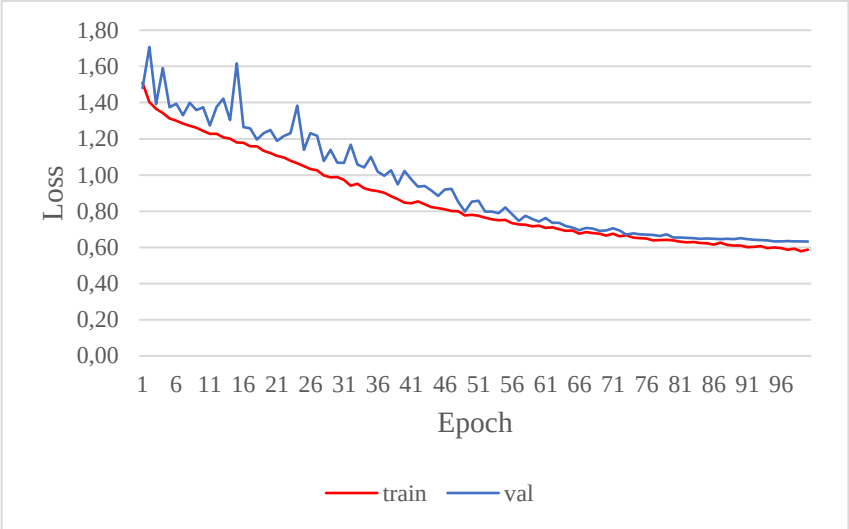
2 dengan baik, dimana TP kelas ini tidak mencapai setengah dari total citra tes kelas tersebut. Untuk kelas 1, 3, dan 4 mampu diklasifikasi oleh model dengan lebih baik. Hal ini dapat dilihat dari jumlah TP dari kelas 1, 3, dan 4 mencapai setengah dari total kelas.

4.3.4 Hasil Pelatihan Dataset *Flipping*

Model dataset *flipping* tidak menunjukkan adanya *underfitting* atau *overfitting*. Hal ini dapat dilihat dari nilai akurasi *training* dan nilai akurasi *testing* yang tidak berbeda jauh. Nilai akurasi *training* model dataset *flipping* mencapai 0.91, sedangkan rata-rata nilai akurasi *testing* model dataset *flipping* mencapai 0.93. Berikut adalah grafik *loss* dan akurasi dari *training* model dataset *flipping*.



Gambar 4.22 Grafik Akurasi *Training* Dataset *Flipping*



Gambar 4.23 Grafik *Loss Training* dan Validasi Dataset *Flipping*

Pada gambar 4.22 dapat dilihat grafik model dataset *flipping* mengalami peningkatan akurasi *training* yang signifikan dari 0.35 menjadi 0.91. Pada gambar 4.23 juga dapat dilihat grafik model dataset *flipping* mengalami penurunan *loss training* dari 1.51 menjadi 0.59, sedangkan nilai *loss* validasi mengalami penurunan dari 1.48 menjadi 0.63. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari model dataset *flipping*.

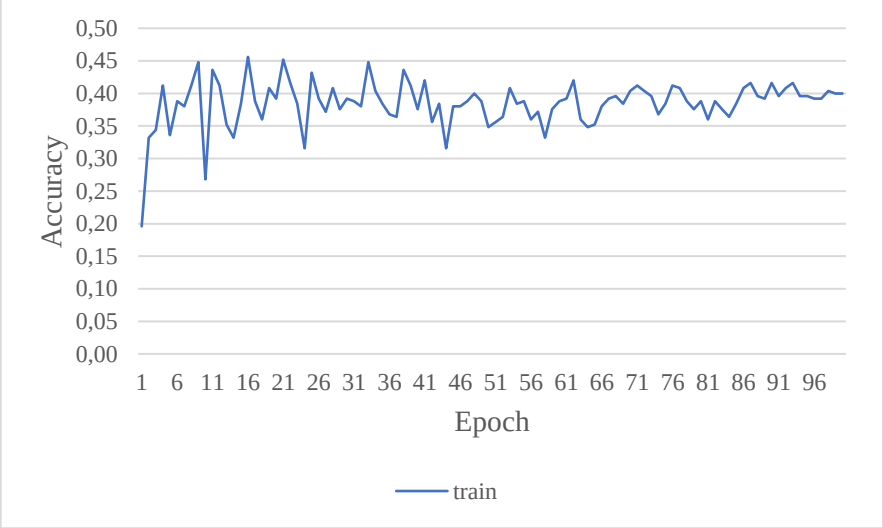
		Predicted				
		0	1	2	3	4
	0	402	10	7	3	0
	1	19	368	6	2	2
Actual	2	13	20	341	2	3
	3	13	3	24	360	4
	4	1	4	13	9	371

Gambar 4.24 *Confusion Matrix* Dataset *Flipping*

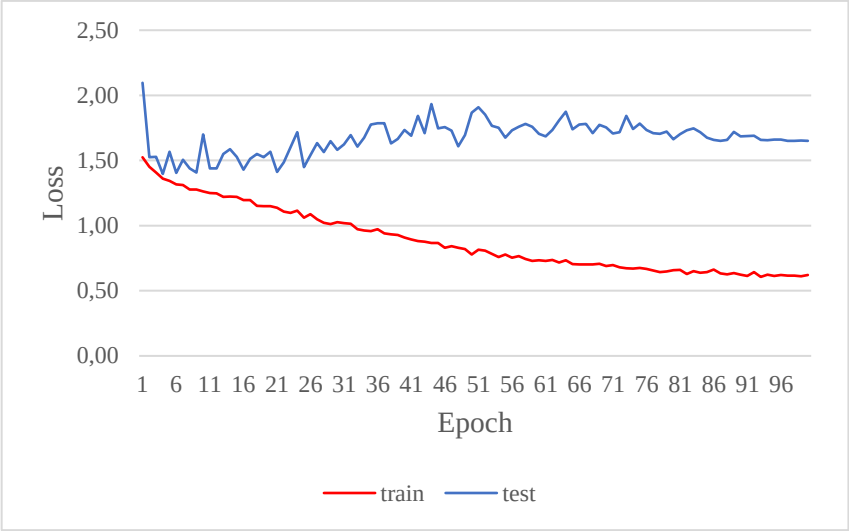
Berdasarkan gambar 4.12, model dataset *flipping* berhasil mengklasifikasikan penyakit RD dengan akurasi 0.92. Model dataset *flipping* dapat mengklasifikasikan seluruh kelas RD dengan baik. Hal ini dapat dilihat dari hasil *confusion matrix* dimana *predicted label* masing-masing kelas hampir seluruhnya telah sesuai dengan *actual label*.

4.3.5 Hasil Pelatihan Dataset *Cropping* dan Histeq

Model dataset *cropping* dan histeq menunjukkan adanya *underfitting*. Hal ini dapat dilihat dari nilai akurasi training lebih rendah dibandingkan nilai akurasi *testing*. Nilai akurasi *training* model dataset *cropping* dan histeq hanya mencapai 0.40, sedangkan rata-rata nilai akurasi *testing* model dataset *cropping* dan histeq adalah 0.53. Berikut adalah grafik *loss* dan akurasi dari *training* model dataset *cropping* dan histeq.



Gambar 4.25 Grafik Akurasi *Training* Dataset *Cropping* dan Histeq



Gambar 4.26 Grafik *Loss Training* dan Validasi Dataset *Cropping* dan Histeq

Pada gambar 4.25 dapat dilihat grafik model dataset *cropping* dan histeq mengalami peningkatan akurasi *training* dari 0.19 menjadi 0.40. Pada gambar 4.26 juga dapat dilihat grafik model dataset *cropping* dan histeq mengalami penurunan *loss training* dari 1.50 menjadi 0.62, sedangkan nilai *loss* validasi mengalami penurunan dari 2.10 menjadi 1.65. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari model dataset *cropping* dan histeq.

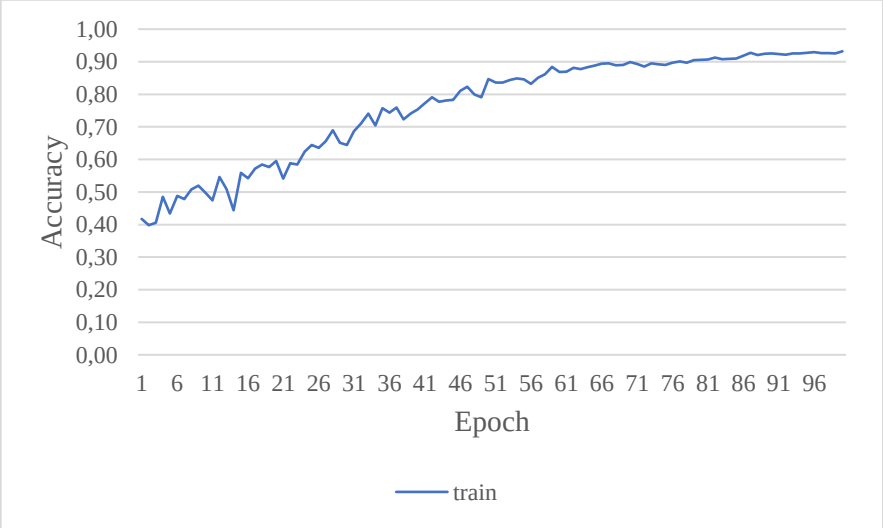
		Predicted				
		0	1	2	3	4
Actual	0	40	39	24	3	6
	1	9	51	24	2	5
	2	9	13	50	14	8
	3	2	4	16	70	11
	4	3	1	5	17	74

Gambar 4.27 *Confusion Matrix* Dataset *Cropping* dan Histeq

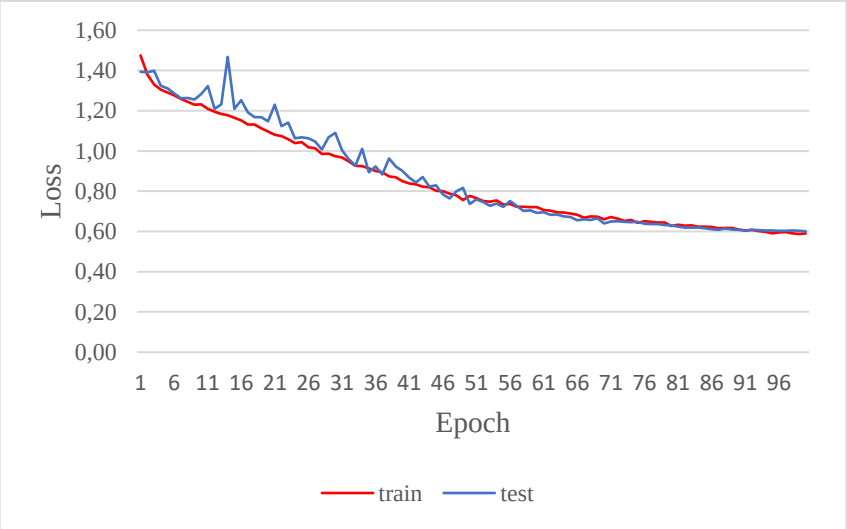
Berdasarkan gambar 4.27, model dataset *cropping* dan histeq berhasil mengklasifikasikan RD dengan akurasi 0.57. Model ini masih belum dapat mengklasifikasi kelas 0 dengan baik, dimana TP kelas ini tidak mencapai setengah dari total citra tes kelas tersebut. Untuk kelas 1 dan kelas 2 mampu diklasifikasi oleh model dengan lebih baik. Hal ini dapat dilihat dari jumlah TP dari kelas 1 dan kelas 2 mencapai setengah dari total kelas. Sedangkan untuk kelas 3 dan kelas 4 mampu diklasifikasi dengan baik, hal ini dapat dilihat dari jumlah TP kelas 3 dan kelas 4 hampir mencapai total citra tes kelas tersebut.

4.3.6 Hasil Pelatihan Dataset *Cropping* dan *Flipping*

Model dataset *cropping* dan *flipping* menunjukkan adanya *overfitting*. Hal ini dapat dilihat dari nilai akurasi *training* lebih tinggi dari nilai akurasi *testing*. Nilai akurasi *training* model dataset *cropping* dan *flipping* mencapai 0.93, sedangkan rata-rata nilai akurasi *testing* model dataset *cropping* dan *flipping* mencapai 0.82. Berikut adalah grafik *loss* dan akurasi dari *training* model dataset *cropping* dan *flipping*.



Gambar 4.28 Grafik Akurasi *Training* Dataset *Cropping* dan *Flipping*



Gambar 4.29 Grafik *Loss Training* dan Validasi Dataset *Cropping* dan *Flipping*

Pada gambar 4.28 dapat dilihat grafik model dataset *cropping* dan *flipping* mengalami peningkatan akurasi *training* dari 0.41 menjadi 0.93. Pada gambar 4.29 juga dapat dilihat grafik model dataset *cropping* dan *flipping* mengalami penurunan *loss training* dari 1.50 menjadi 0.59, sedangkan nilai *loss* validasi mengalami penurunan dari 1.40 menjadi 0.60. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari model dataset *cropping* dan *flipping*.

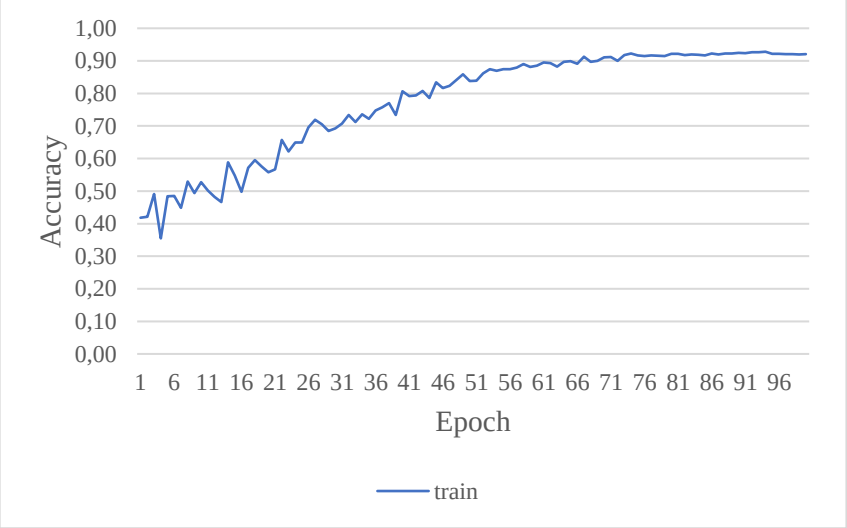
			Predicted			
		0	1	2	3	4
	0	421	11	9	2	4
	1	22	316	10	2	1
Actual	2	20	12	369	14	9
	3	9	0	12	381	8
	4	1	0	4	14	345

Gambar 4.30 *Confusion Matrix* Dataset *Cropping* dan *Flipping*

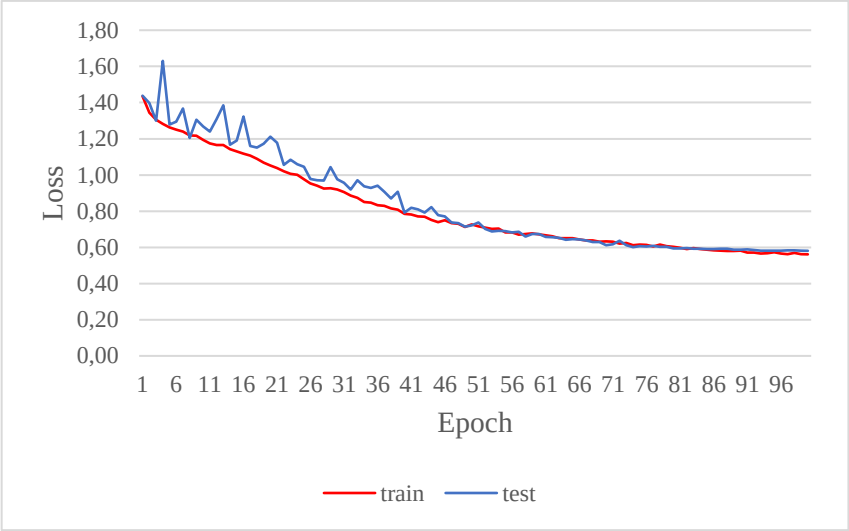
Berdasarkan gambar 4.30, model dataset *cropping* dan *flipping* berhasil mengklasifikasikan penyakit RD dengan akurasi 0.92. Model dataset *cropping* dan *flipping* dapat mengklasifikasikan seluruh kelas RD dengan baik. Hal ini dapat dilihat dari hasil *confusion matrix* dimana *predicted label* masing-masing kelas hampir seluruhnya telah sesuai dengan *actual label*.

4.3.7 Hasil Pelatihan Dataset Histeq dan *Flipping*

Model dataset histeq dan *flipping* menunjukkan adanya *overfitting*. Hal ini dapat dilihat dari nilai akurasi *training* jauh lebih tinggi dari nilai akurasi *testing*. Nilai akurasi *training* model dataset histeq dan *flipping* mencapai 0.92, sedangkan rata-rata nilai akurasi *testing* model dataset histeq dan *flipping* mencapai 0.69. Berikut adalah grafik *loss* dan akurasi dari *training* model dataset histeq dan *flipping*.



Gambar 4.31 Grafik Akurasi *Training* Dataset Histeq dan *Flipping*



Gambar 4.32 Grafik *Loss Training* dan Validasi Dataset Histeq dan *Flipping*

Pada gambar 4.31 dapat dilihat grafik model dataset histeq dan *flipping* mengalami peningkatan akurasi *training* dari 0.41 menjadi 0.92. Pada gambar 4.32 juga dapat dilihat grafik model dataset histeq dan *flipping* mengalami penurunan *loss training* dari 1.40 menjadi 0.56, sedangkan nilai *loss* validasi mengalami penurunan dari 1.40 menjadi 0.58. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari model dataset histeq dan *flipping*.

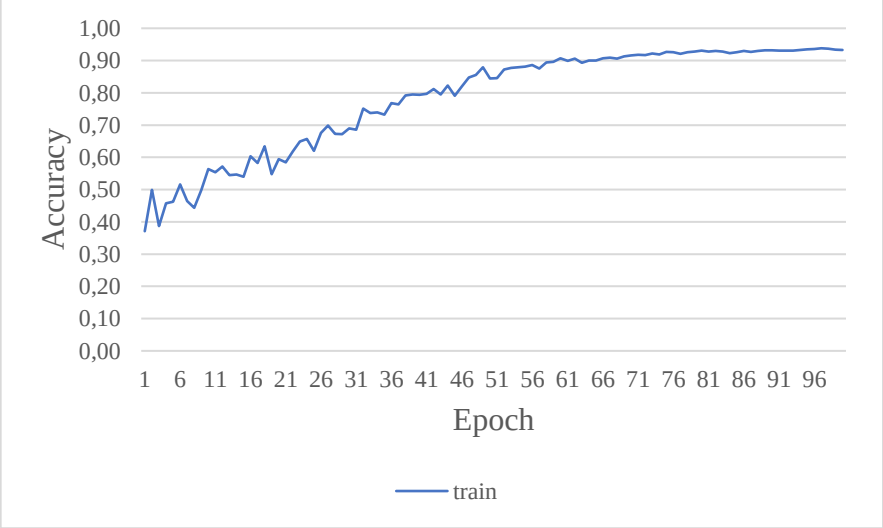
		Predicted				
		0	1	2	3	4
Actual	0	355	23	10	0	3
	1	31	372	22	1	1
	2	9	18	345	8	2
	3	9	1	7	380	10
	4	1	3	4	11	374

Gambar 4.33 *Confusion* Matrix Dataset Histeq dan *Flipping*

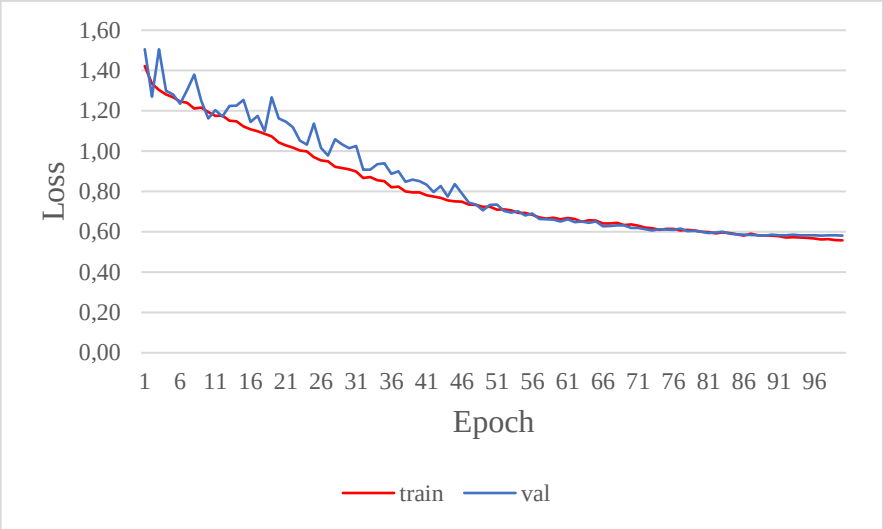
Berdasarkan gambar 4.33, model dataset histeq dan *flipping* berhasil mengklasifikasikan penyakit RD dengan akurasi 0.91. Model dataset histeq dan *flipping* dapat mengklasifikasikan seluruh kelas RD dengan baik. Hal ini dapat dilihat dari hasil *confusion matrix* dimana *predicted label* masing-masing kelas hampir seluruhnya telah sesuai dengan *actual label*.

4.3.8 Hasil Pelatihan Dataset Setelah *Preprocessing* dan Augmentasi

Model setelah *preprocessing* dan augmentasi menunjukkan adanya *overfitting*. Nilai akurasi *training* lebih tinggi dibandingkan dengan akurasi *testing*. Nilai akurasi *training* model setelah *preprocessing* dan augmentasi mencapai 0.93, sedangkan nilai akurasi *testing* model setelah *preprocessing* dan augmentasi rata-rata hanya mencapai 0.73. Berikut adalah grafik *loss* dan akurasi dari *training* model setelah *preprocessing* dan augmentasi.



Gambar 4.34 Grafik Akurasi *Training* Dataset Setelah *Preprocessing* dan Augmentasi



Gambar 4.35 Grafik *Loss Training* dan Validasi Dataset Setelah *Preprocessing* dan Augmentasi

Pada gambar 4.34 dapat dilihat grafik model dataset setelah *preprocessing* dan augmentasi mengalami peningkatan akurasi *training* dari 0.38 menjadi 0.93. Pada gambar 4.35 juga dapat dilihat grafik model dataset setelah *preprocessing* dan augmentasi mengalami penurunan *loss training* dari 1.50 menjadi 0.56, sedangkan nilai *loss* validasi mengalami penurunan dari 1.40 menjadi 0.58. Setelah itu dilakukan *testing* model menggunakan model yang telah disimpan sebelumnya. Berikut adalah salah satu *confusion matrix* dari model dataset setelah *preprocessing* dan augmentasi.

		Predicted				
		0	1	2	3	4
Actual	0	368	20	10	2	2
	1	23	375	8	2	2
	2	13	12	345	4	11
	3	3	7	9	370	8
	4	0	2	3	12	389

Gambar 4.36 *Confusion Matrix* Dataset Setelah *Preprocessing* dan Augmentasi

Berdasarkan gambar 4.36, model dataset setelah *preprocessing* dan augmentasi berhasil mengklasifikasikan penyakit RD dengan akurasi 0.92. Model dataset setelah *preprocessing* dan augmentasi dapat mengklasifikasikan seluruh kelas RD dengan baik. Hal ini dapat dilihat dari hasil *confusion matrix* dimana *predicted label* masing-masing kelas hampir seluruhnya telah sesuai dengan *actual label*.

4.4 Hasil Pengujian Model Analisis *Preprocessing* dan Augmentasi Data

Delapan model di atas akan diuji untuk menentukan *preprocessing* dan augmentasi data yang memiliki pengaruh terbaik terhadap akurasi model. Dalam tahap ini, digunakan uji normalitas dan uji statistik. Uji normalitas yang digunakan pada tahap ini adalah uji Shapiro-Wilk. Uji statistik yang digunakan pada tahap ini adalah uji ANOVA dan uji Tukey HSD atau uji Kruskal-Wallis dan uji Mann-Whitney U. Berikut adalah hasil dari uji normalitas dan uji statistik dari delapan model tersebut.

4.4.1 Hasil Uji Shapiro-Wilk Pengujian Model Analisis *Preprocessing* dan Augmentasi Data

Pada tahap ini menggunakan uji Shapiro-Wilk untuk menentukan apakah sampel akurasi tes berdistribusi normal atau tidak. Nilai alpha yang ditentukan untuk uji Sapiro-Wilk adalah 0.05. Berikut adalah hipotesis nol dan hipotesis alternatif yang digunakan dalam uji Shapiro-Wilk:

H0: data berdistribusi normal.
H1: data tidak berdistribusi normal.

Contoh perhitungan uji Shapiro-Wilk pada model *flipping*:

Data model *flipping*:

$$X_{(1)} = 0.913, X_{(2)} = 0.934, X_{(3)} = 0.935, X_{(4)} = 0.935, X_{(5)} = 0.935$$

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.913 + 0.934 + 0.935 + 0.935 + 0.935}{5}$$

$$= 0.9304$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$

$$= 0.0003792$$

Nilai koefisien Shapiro-Wilk untuk $n = 5$:

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(n+1-i)} - X_i))^2}{\sum_{i=1}^{5} (X_i - \bar{X})^2}$$

$$= \frac{(0.0146212 + 0.0002413)^2}{0.0003792}$$

$$= 0.582$$

Karena nilai W lebih kecil dari nilai W untuk *p-value* 0.01 pada tabel nilai kritis uji Shapiro-Wilk, maka dianggap nilai *p-value* lebih kecil dari 0.01. Karena *p-value* lebih kecil dibandingkan alpha, maka H0 ditolak, maka dapat disimpulkan data tidak terdistribusi normal.

Contoh perhitungan uji Shapiro-Wilk pada model asli:

Data model asli:

$$X_{(1)} = 0.578, X_{(2)} = 0.589, X_{(3)} = 0.666, X_{(4)} = 0.674, X_{(5)} = 0.694$$

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.578 + 0.589 + 0.666 + 0.674 + 0.694}{5}$$

$$= 0.6402$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$

$$= 0.0111928$$

Nilai koefisien Shapiro-Wilk untuk $n = 5$:

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(n+1-i)} - X_i))^2}{\sum_{i=1}^{5} (X_i - \bar{X})^2}$$

$$= \frac{0.00952656033681}{0.0111928}$$

$$= 0.851132$$

Karena nilai W lebih besar dari nilai W untuk *p-value* 0.10 pada tabel nilai kritis uji Shapiro-Wilk, maka dianggap *p-value* lebih tinggi dari 0.10. Karena *p-value* lebih besar dibandingkan alpha, maka H0 diterima, maka dapat disimpulkan bahwa data terdistribusi normal.

Tabel 4.5 Hasil Uji Shapiro-Wilk Pengujian Model Analisis *Preprocessing* dan Augmentasi Data

Model	*p-value*	Hipotesis
Asli	0.19	H0 diterima
Cropping	0.05	H0 ditolak
Histeq	0.50	H0 diterima
Flipping	0.00	H0 ditolak
Cropping dan Histeq	0.08	H0 diterima
Cropping dan *Flipping*	0.54	H0 diterima
Histeq dan *Flipping*	0.73	H0 diterima
Setelah *Preprocessing* dan Augmentasi Data	0.37	H0 diterima

Dari tabel 4.5 dapat dilihat bahwa terdapat data yang tidak terdistribusi normal, yaitu model *cropping* dan model *flipping*. Oleh karena itu, pada tahap selanjutnya akan digunakan uji Kruskal-Wallis.

4.4.2 Hasil Uji Kruskall-Wallis Pengujian Model Analisis *Preprocessing* dan Augmentasi Data

Pada tahap ini digunakan uji Kruskal-Wallis untuk menentukan apakah terdapat perbedaan signifikan antara akurasi tes dari model-model tersebut. Nilai alpha yang ditentukan untuk uji Kruskal-Wallis adalah 0.05. Berikut adalah hipotesis nol dan hipotesis alternatif yang digunakan dalam uji ANOVA:

H0: tidak terdapat perbedaan signifikan antara akurasi tes masing-masing model.

H1: terdapat perbedaan signifikan antara akurasi tes masing-masing model.

Perhitungan uji Kruskal-Wallis Pengujian Model:

Data model:

Asli = 0.694, 0.666, 0.674, 0.589, 0.578

Cropping = 0.57, 0.536, 0.536, 0.54, 0.546

Histeq = 0.45, 0.462, 0.442, 0.462, 0.468

Flipping = 0.913, 0.935, 0.935, 0.935, 0.934

Cropping dan histeq = 0.456, 0.54, 0.538, 0.56, 0.568

Cropping dan *flipping* = 0.823, 0.813, 0.813, 0.81, 0.818

Histeq dan *flipping* = 0.687, 0.697, 0.682, 0.695, 0.692

Setelah *preprocessing dan augmentasi* = 0.739, 0.708, 0.731, 0.728, 0.727

Peringkat Sampel Model:

Asli = 23, 18, 19, 17, 16

Cropping = 15, 7, 8, 10.5, 12

Histeq = 2, 4, 1, 5, 6

Flipping = 35, 38, 39, 40, 37

Cropping dan histeq = 3, 10.5, 9, 13, 14

Cropping dan *flipping* = 35, 33, 32, 31, 34

Histeq dan *flipping* = 21, 25, 20, 24, 22

Setelah *preprocessing dan augmentasi* = 30, 26, 29, 27, 28

Total Peringkat masing-masing kelompok:

Asli = 93

Cropping =52.5

Histeq = 18

Flipping = 190

Cropping dan histeq = 49.5

Cropping dan *flipping* = 165

Histeq dan *flipping* = 112

Setelah *preprocessing dan augmentasi* = 140

$$H = \frac{12}{40(40+1)} \sum_{i=1}^{k} \frac{{R_i}^2}{n_i} - 3(40+1)$$

$$H = \frac{12}{1640} \times \left(\frac{93^2}{5} + \frac{52.5^2}{5} + \frac{18^2}{5} + \frac{190^2}{5} + \frac{49.5^2}{5} + \frac{165^2}{5} + \frac{112^2}{5} + \frac{140^2}{5} \right) - 123$$

$$H = 37.49$$

$$df = N - 1 = 8 - 1 = 7$$

Nilai *p-value* jika $df = 7$ adalah 0.00. Karena nilai *p-value* lebih kecil dibandingkan nilai alpha, maka hipotesis nol ditolak. Oleh karena itu, dapat disimpulkan bahwa terdapat perbedaan signifikan antara akurasi tes model-model. Untuk mengetahui model-model yang terdapat perbedaan signifikan, dilakukan uji Mann-Whitney U.

4.4.3 Hasil Uji Mann-Whitney U Pengujian Model Analisis *Preprocessing* dan Augmentasi Data

Sebagai kelanjutan dari uji Kruskal-Wallis, digunakan uji Mann-Whitney U untuk mengetahui model yang terdapat perbedaan signifikan. Nilai alpha yang ditentukan untuk uji Mann-Whitney U adalah 0.05. Berikut adalah hipotesis nol dan hipotesis alternatif yang digunakan dalam uji Mann-Whitney U:

H0: tidak terdapat perbedaan signifikan antara akurasi tes model A dengan model B.

H1: terdapat perbedaan signifikan antara akurasi tes model A dengan model B.

Perhitungan uji Mann-Whitney U Pengujian Model Analisis Preprocessing dan Augmentasi Data antara model asli dan model *cropping*:

Data model:

Asli: 0.578, 0.589, 0.666, 0.674, 0.694

Cropping: 0.536, 0.536, 0.54, 0.546, 0.57

Peringkat Sampel Model:

Asli: 6, 7, 8, 9, 10

Cropping: 1, 2, 3, 4, 5

Total Peringkat masing-masing kelompok:
Asli: 40
Cropping: 15

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Nilai U jika menggunakan alpha 0.05 adalah 2. Karena nilai *p-value* lebih kecil dibandingkan nilai alpha, maka hipotesis nol ditolak. Oleh karena itu, dapat disimpulkan bahwa terdapat perbedaan signifikan antara akurasi tes model asli dengan model *cropping*.

Perhitungan uji Mann-Whitney U Pengujian Model Analisis Preprocessing dan Augmentasi Data antara model asli dan model histeq *flipping*:

Data model:
asli: 0.578, 0.589, 0.666, 0.674, 0.694
histeq *flipping*: 0.682, 0.687, 0.692, 0.695, 0.697

Peringkat Sampel Model:

asli: 1, 2, 3, 4, 8

histeq *flipping*: 5, 6, 7, 9, 10

Total Peringkat masing-masing kelompok:

asli: 18

histeq *flipping*: 37

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 18 = 22$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 37 = 3$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 22 = 3$$

Nilai U jika menggunakan alpha 0.05 adalah 3. Karena nilai *p-value* lebih besar dibandingkan nilai alpha, maka hipotesis nol diterim. Oleh karena itu, dapat disimpulkan bahwa tidak terdapat perbedaan signifikan antara akurasi tes model asli dengan model histeq *flipping*.

Tabel 4.6 Hasil Uji Mann-Whitney U Pengujian Model Analisis *Preprocessing* dan Augmentasi Data

Model A	Model B	*p-value*	Hipotesis
Asli	*Cropping*	0.01	H0 Ditolak
Asli	Histeq	0.01	H0 Ditolak
Asli	*Flipping*	0.01	H0 Ditolak

Tabel 4.6 Lanjutan

Asli	*Cropping* dan Histeq	0.01	H0 Ditolak
Asli	*Cropping* dan *Flipping*	0.01	H0 Ditolak
Asli	Histeq dan *Flipping*	0.06	H0 Diterima
Asli	Setelah *Preprocessing* dan Augmentasi Data	0.01	H0 Ditolak
Cropping	Histeq	0.01	H0 Ditolak
Cropping	*Flipping*	0.01	H0 Ditolak
Cropping	*Cropping* dan Histeq	1.00	H0 Diterima
Cropping	*Cropping* dan *Flipping*	0.01	H0 Ditolak
Cropping	Histeq dan *Flipping*	0.01	H0 Ditolak
Cropping	Setelah *Preprocessing* dan Augmentasi Data	0.01	H0 Ditolak
Histeq	*Flipping*	0.01	H0 Ditolak
Histeq	*Cropping* dan Histeq	0.06	H0 Diterima
Histeq	*Cropping* dan *Flipping*	0.01	H0 Ditolak
Histeq	Histeq dan *Flipping*	0.01	H0 Ditolak
Histeq	Setelah *Preprocessing* dan Augmentasi Data	0.00	H0 Ditolak
Flipping	*Cropping* dan Histeq	0.01	H0 Ditolak
Flipping	*Cropping* dan *Flipping*	0.01	H0 Ditolak
Flipping	Histeq dan *Flipping*	0.01	H0 Ditolak
Flipping	Setelah *Preprocessing* dan Augmentasi Data	0.01	H0 Ditolak
Cropping dan Histeq	*Cropping* dan *Flipping*	0.01	H0 Ditolak
Cropping dan Histeq	Histeq dan *Flipping*	0.00	H0 Ditolak
Cropping dan Histeq	Setelah *Preprocessing* dan Augmentasi Data	0.00	H0 Ditolak
Cropping dan *Flipping*	Histeq dan *Flipping*	0.01	H0 Ditolak
Cropping dan *Flipping*	Setelah *Preprocessing* dan Augmentasi Data	0.01	H0 Ditolak

Tabel 4.6 Lanjutan

Histeq dan *Flipping*	Setelah *Preprocessing* dan Augmentasi Data	0.00	H0 Ditolak

Dari tabel 4.6 dapat dilihat bahwa terdapat perbedaan signifikan di hampir seluruh dataset. Dimana hanya dataset asli dengan histeq dan *flipping*, *cropping* dengan *cropping* dan histeq, dan histeq dengan *cropping* dan histeq yang tidak berbeda signifikan. Merujuk pada tabel 4.3, model dataset *flipping* memiliki akurasi *training* dan *testing* yang paling baik, maka *preprocessing* atau augmentasi yang terbaik untuk meningkatkan performa model adalah augmentasi *flipping* tanpa *preprocessing*.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berikut kesimpulan dari penelitian ini.

1. Pada penelitian ini telah berhasil dikembangkan model CNN untuk mengklasifikasi penyakit retinopati diabetik (RR). Model CNN terbaik untuk mengklasifikasi penyakit RD adalah model YOLO v5Nano, dengan akurasi *train* 0.93 dan rata-rata akurasi tes mencapai 0.73.
2. Pada penelitian ini berhasil diterapkan *preprocessing* dan augmentasi data untuk meningkatkan performa dari model CNN yang digunakan. *Preprocessing* dan augmentasi data terbaik untuk meningkatkan performa model CNN adalah augmentasi *flipping* tanpa *preprocessing*. Augmentasi *flipping* mampu meningkatkan akurasi *train* model YOLO v5Nano menjadi 0.91 dan akurasi tes mencapai 0.93.

5.2 Saran

Adapun saran dan masukan yang dapat dilakukan sebagai referensi untuk penelitian selanjutnya yang berkaitan dengan topik penelitian ini adalah sebagai berikut:

1. Menggunakan dataset yang menunjukkan perbedaan antar kelas dengan lebih jelas.
2. Menggunakan arsitektur yang berbeda.
3. Menggunakan augmentasi data lebih beragam.

DAFTAR PUSTAKA

Al Husaini, M. A. S., Habaebi, M. H., Gunawan, T. S., Islam, M. R., Elsheikh, E. A. A., & Suliman, F. M. (2022). Thermal-based early breast cancer detection using inception V3, inception V4 and modified inception MV4. *Neural Computing and Applications*, *34*(1), 333–348. https://doi.org/10.1007/s00521-021-06372-1

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1). https://doi.org/10.1186/s40537-021-00444-8

Bi, Q., Goodman, K. E., Kaminsky, J., & Lessler, J. (2019). What is machine learning? A primer for the epidemiologist. *American Journal of Epidemiology*, *188*(12), 2222–2239. https://doi.org/10.1093/aje/kwz189

Dugas, E., Jared, Jorge, & Cukierski, W. (2015). *Diabetic Retinopathy Detection*. Kaggle. https://www.kaggle.com/competitions/diabetic-retinopathy-detection

Fajariyanti, Y. (2017). *Perbedaan Quality of Life Pada Penderita Proliferative Diabetic Retinopathy Dengan dan Tanpa Laser Panretinal Photocoagulation*.

Fan, J., Ma, C., & Zhong, Y. (2021). A Selective Overview of Deep Learning. *Statistical Science*, *36*(2), 264–290. https://doi.org/10.1214/20-STS783

International Diabetes Federation, & The Fred Hollows Foundation. (2015). *Diabetes eye health: A guide for health professionals*. www.idf.org/eyecare

Kamal, K., Ez-Zahraouy, H., & Halloum, K. (2023). *A comparison between the VGG16, VGG19 and ResNet50 architecture frameworks for classiication of normal and CLAHE processed medical images A comparison between the VGG16, VGG19 and ResNet50 architecture frameworks for classification of normal and CLAHE processed medical images*. https://doi.org/10.21203/rs.3.rs-2863523/v1

Kementerian Kesehatan Republik Indonesia. (2018). *Pedoman Nasional Pelayanan Kedokteran Retinopati Diabetika*.

Lin, C. L., & Wu, K. C. (2023). Development of revised ResNet-50 for diabetic retinopathy detection. *BMC Bioinformatics*, *24*(1). https://doi.org/10.1186/s12859-023-05293-1

Mishra, B. K., Thakker, D., Mazumdar, S., Neagu, D., Gheorghe, M., & Simpson, S. (2020). A novel application of deep learning with image cropping: a smart city use case for flood monitoring. *Journal of Reliable Intelligent Environments*, *6*(1), 51–61. https://doi.org/10.1007/s40860-020-00099-x

Mumuni, A., & Mumuni, F. (2022). Data augmentation: A comprehensive survey of modern approaches. In *Array* (Vol. 16). Elsevier B.V. https://doi.org/10.1016/j.array.2022.100258

Pramunendar, R. A., Andono, P. N., Soeleman, Moch. A., Prabowo, D. P., & Pergiwati, D. (2020). *Buku Pengenalan Citra 2 Dimensi Menggunakan MATLAB*.

Qummar, S., Khan, F. G., Shah, S., Khan, A., Shamshirband, S., Rehman, Z. U., Khan, I. A., & Jadoon, W. (2019). A Deep Learning Ensemble Approach for Diabetic Retinopathy Detection. *IEEE Access*, *7*, 150530–150539. https://doi.org/10.1109/ACCESS.2019.2947484

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. http://arxiv.org/abs/1506.02640

Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence A Modern Approach Third Edition*. www.PlentyofeBooks.net

Shankar, K., Zhang, Y., Liu, Y., Wu, L., & Chen, C. H. (2020). Hyperparameter Tuning Deep Learning for Diabetic Retinopathy Fundus Image Classification. *IEEE Access*, *8*, 118164–118173. https://doi.org/10.1109/ACCESS.2020.3005152

Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. http://arxiv.org/abs/1409.1556

Sudha, V., & Ganeshbabu, T. R. (2021). A convolutional neural network classifier VGG-19 architecture for lesion detection and grading in diabetic retinopathy based on deep learning. *Computers, Materials and Continua*, *66*(1), 827–842. https://doi.org/10.32604/cmc.2020.012008

Sulistiyanti, S. R., Setyawan, F. A., & Komarudin, M. (2016). *PENGOLAHAN CITRA; DASAR DAN CONTOH PENERAPANNYA*.

Valverde, C., Garcia, M., Hornero, R., & Lopez-Galvez, M. (2016). Automated detection of diabetic retinopathy in retinal images. In *Indian Journal of Ophthalmology* (Vol. 64, Issue 1, pp. 26–32). Medknow Publications. https://doi.org/10.4103/0301-4738.178140

Wang, W., & Lo, A. C. Y. (2018). Diabetic retinopathy: Pathophysiology and treatments. In *International Journal of Molecular Sciences* (Vol. 19, Issue 6). MDPI AG. https://doi.org/10.3390/ijms19061816

Yang, S., Xiao, W., Zhang, M., Guo, S., Zhao, J., & Shen, F. (2022). *Image Data Augmentation for Deep Learning: A Survey*. http://arxiv.org/abs/2204.08610

LAMPIRAN

Lampiran nomor 1:

Kode Arsitektur ResNet-50

```
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

# Load pre-trained ResNet-50 model
base_model = ResNet50(input_shape= (224,224,3),
                      include_top = False,
                      weights = 'imagenet')

# Freeze the layers in the base model (optional)
for layer in base_model.layers:
    layer.trainable = False

last_layer = base_model.get_layer('conv5_block3_out')
print('last layer output shape: ', last_layer.output_shape)
last_output = last_layer.output

from tensorflow.keras.optimizers import Adam
# Flatten the output layer to 1 dimension
x = layers.Flatten()(last_output)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dropout(0.4)(x)
x = layers.Dense(5, activation='softmax')(x)

# Append the dense network to the base model
model = Model(base_model.input, x)

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Lampiran nomor 2:

Kode Arsitektur Inception-v4

```
def inception_v4_block(x, num1x1, num3x3red, num3x3, num3x3dblred,
num3x3dbl, poolproj, name=None):
    if name is not None:
        conv1_1 = layers.Conv2D(num1x1, (1, 1), padding='same',
activation='relu', name=name + '_1x1')(x)

        conv3_3_reduce = layers.Conv2D(num3x3red, (1, 1),
padding='same', activation='relu', name=name + '_3x3_reduce')(x)
        conv3_3 = layers.Conv2D(num3x3, (3, 3), padding='same',
activation='relu', name=name + '_3x3')(conv3_3_reduce)

        conv3_3dbl_reduce = layers.Conv2D(num3x3dblred, (1, 1),
padding='same', activation='relu', name=name +
'_3x3dbl_reduce')(x)
        conv3_3dbl = layers.Conv2D(num3x3dbl, (3, 3),
padding='same', activation='relu', name=name +
'_3x3dbl')(conv3_3dbl_reduce)
        conv3_3dbl = layers.Conv2D(num3x3dbl, (3, 3),
padding='same', activation='relu', name=name +
'_3x3dbl_2')(conv3_3dbl)

        pool_proj = layers.AveragePooling2D((3, 3), strides=(1,
1), padding='same', name=name + '_pool_proj')(x)
        pool_proj = layers.Conv2D(poolproj, (1, 1),
padding='same', activation='relu', name=name +
'_pool_proj_1x1')(pool_proj)

        output = layers.concatenate([conv1_1, conv3_3, conv3_3dbl,
pool_proj], axis=-1, name=name + '_output')
    else:
        raise ValueError('Blok Inception harus memiliki nama.')

    return output

def inception_v4_model(input_shape=(224, 224, 3), num_classes=5):
    input_layer = layers.Input(shape=input_shape,
name='input_layer')
```

```
    x = layers.Conv2D(32, (3, 3), strides=(2, 2), padding='valid',
activation='relu', name='conv1_3x3/2')(input_layer)
    x = layers.Conv2D(32, (3, 3), padding='valid',
activation='relu', name='conv2_3x3/1')(x)
    x = layers.Conv2D(64, (3, 3), padding='same',
activation='relu', name='conv3_3x3/1')(x)
    x = layers.MaxPooling2D((3, 3), strides=(2, 2),
name='maxpool1_3x3/2')(x)

    x = layers.Conv2D(80, (1, 1), padding='same',
activation='relu', name='conv4_1x1/1')(x)
    x = layers.Conv2D(192, (3, 3), padding='valid',
activation='relu', name='conv5_3x3/1')(x)
    x = layers.MaxPooling2D((3, 3), strides=(2, 2),
name='maxpool2_3x3/2')(x)

    x = inception_v4_block(x, 64, 96, 128, 16, 32, 32,
name='inception_a')
    x = inception_v4_block(x, 64, 96, 128, 16, 32, 32,
name='inception_b')
    x = inception_v4_block(x, 64, 96, 128, 16, 32, 32,
name='inception_c')

    x = inception_v4_block(x, 384, 192, 448, 512, 256, 256,
name='inception_d')

    x = layers.GlobalAveragePooling2D(name='avg_pool')(x)
    x = layers.Dense(num_classes, activation='softmax',
name='output')(x)

    model = Model(inputs=input_layer, outputs=x,
name='inception_v4')
    return model

# Buat model Inception-v4 dengan ukuran input 224x224
model = inception_v4_model(input_shape=(224, 224, 3))

# Compile the model
```

```
model.compile(optimizer=Adam(lr=0.0001),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Lampiran nomor 3:

Kode Arsitektur VGG-19

```
from tensorflow.keras.applications import VGG19
from tensorflow.keras.layers import Input, Flatten, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

# Create an Input layer for your data
input_layer = Input(shape=(224, 224, 3))

# Instantiate the VGG19 model with weights pre-trained on ImageNet
base_model = VGG19(weights='imagenet',
                   include_top=False,
                   input_tensor=input_layer)

# Freeze the layers in the pre-trained model
for layer in base_model.layers:
    layer.trainable = False

# Get the output of the desired layer from the InceptionV4 model
last_layer = base_model.get_layer('block5_pool')
print('last layer output shape: ', last_layer.output_shape)
last_output = last_layer.output

# Get the output of the last layer from the VGG19 model
last_layer = base_model.get_layer('block5_pool')

# Flatten the output layer to 1 dimension
x = Flatten()(last_layer.output)
x = Dense(512, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.4)(x)
output = Dense(5, activation='softmax')(x)

# Create the custom model by specifying the input and output
layers
model = Model(inputs=input_layer, outputs=output)

# Print the model summary
```

```
model.summary()

# Compile the model
model.compile(optimizer=Adam(lr=0.0001),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Lampiran nomor 4:

Kode Arsitektur YOLO v5Nano

```
def train(opt, device):
    """Trains a YOLOv5 model, managing datasets, model
optimization, logging, and saving checkpoints."""
    init_seeds(opt.seed + 1 + RANK, deterministic=True)
    save_dir, data, bs, epochs, nw, imgsz, pretrained = (
        opt.save_dir,
        Path(opt.data),
        opt.batch_size,
        opt.epochs,
        min(os.cpu_count() - 1, opt.workers),
        opt.imgsz,
        str(opt.pretrained).lower() == "true",
    )
    cuda = device.type != "cpu"

    # Directories
    wdir = save_dir / "weights"
    wdir.mkdir(parents=True, exist_ok=True)  # make dir
    last, best = wdir / "last.pt", wdir / "best.pt"

    # Save run settings
    yaml_save(save_dir / "opt.yaml", vars(opt))

    # Logger
    logger = GenericLogger(opt=opt, console_logger=LOGGER) if RANK
in {-1, 0} else None

    # Download Dataset
    with torch_distributed_zero_first(LOCAL_RANK),
WorkingDirectory(ROOT):
        data_dir = data if data.is_dir() else (DATASETS_DIR /
data)
        if not data_dir.is_dir():
            LOGGER.info(f"\nDataset not found ⚠️, missing path
{data_dir}, attempting download...")
            t = time.time()
            if str(data) == "imagenet":
```

```
                subprocess.run(["bash", str(ROOT /
"data/scripts/get_imagenet.sh")], shell=True, check=True)
            else:
                url =
f"https://github.com/ultralytics/yolov5/releases/download/v1.0/{da
ta}.zip"
                download(url, dir=data_dir.parent)
            s = f"Dataset download success ✅ ({time.time() -
t:.1f}s), saved to {colorstr('bold', data_dir)}\n"
            LOGGER.info(s)

    # Dataloaders
    nc = len([x for x in (data_dir / "train").glob("*") if
x.is_dir()])  # number of classes
    trainloader = create_classification_dataloader(
        path=data_dir / "train",
        imgsz=imgsz,
        batch_size=bs // WORLD_SIZE,
        augment=True,
        cache=opt.cache,
        rank=LOCAL_RANK,
        workers=nw,
    )

    test_dir = data_dir / "test" if (data_dir / "test").exists()
else data_dir / "val"  # data/test or data/val
    if RANK in {-1, 0}:
        testloader = create_classification_dataloader(
            path=test_dir,
            imgsz=imgsz,
            batch_size=bs // WORLD_SIZE * 2,
            augment=False,
            cache=opt.cache,
            rank=-1,
            workers=nw,
        )

    # Model
```

```
    with torch_distributed_zero_first(LOCAL_RANK),
WorkingDirectory(ROOT):
        if Path(opt.model).is_file() or opt.model.endswith(".pt"):
            model = attempt_load(opt.model, device="cpu",
fuse=False)
        elif opt.model in torchvision.models.__dict__:  #
TorchVision models i.e. resnet50, efficientnet_b0
            model =
torchvision.models.__dict__[opt.model](weights="IMAGENET1K_V1" if
pretrained else None)
        else:
            m = hub.list("ultralytics/yolov5")  # +
hub.list('pytorch/vision')  # models
            raise ModuleNotFoundError(f"--model {opt.model} not
found. Available models are: \n" + "\n".join(m))
        if isinstance(model, DetectionModel):
            LOGGER.warning("WARNING ⚠️ pass YOLOv5 classifier
model with '-cls' suffix, i.e. '--model yolov5s-cls.pt'")
            model = ClassificationModel(model=model, nc=nc,
cutoff=opt.cutoff or 10)  # convert to classification model
        reshape_classifier_output(model, nc)  # update class count
    for m in model.modules():
        if not pretrained and hasattr(m, "reset_parameters"):
            m.reset_parameters()
        if isinstance(m, torch.nn.Dropout) and opt.Dropout is not
None:
            m.p = opt.Dropout  # set Dropout
    for p in model.parameters():
        p.requires_grad = True  # for training
    model = model.to(device)

    # Info
    if RANK in {-1, 0}:
        model.names = trainloader.dataset.classes  # attach class
names
        model.transforms = testloader.dataset.torch_transforms  #
attach inference transforms
        model_info(model)
        if opt.verbose:
```

```
            LOGGER.info(model)
        images, labels = next(iter(trainloader))
        file = imshow_cls(images[:25], labels[:25],
names=model.names, f=save_dir / "train_images.jpg")
        logger.log_images(file, name="Train Examples")
        logger.log_graph(model, imgsz)  # log model

    # Optimizer
    optimizer = smart_optimizer(model, opt.optimizer, opt.lr0,
momentum=0.9, decay=opt.decay)

    # Scheduler
    lrf = 0.01  # final lr (fraction of lr0)
    # lf = lambda x: ((1 + math.cos(x * math.pi / epochs)) / 2) *
(1 - lrf) + lrf  # cosine
    lf = lambda x: (1 - x / epochs) * (1 - lrf) + lrf  # linear
    scheduler = lr_scheduler.LambdaLR(optimizer, lr_lambda=lf)
    # scheduler = lr_scheduler.OneCycleLR(optimizer, max_lr=lr0,
total_steps=epochs, pct_start=0.1,
    #                                     final_div_factor=1 / 25 /
lrf)

    # EMA
    ema = ModelEMA(model) if RANK in {-1, 0} else None

    # DDP mode
    if cuda and RANK != -1:
        model = smart_DDP(model)

    # Train
    t0 = time.time()
    criterion =
smartCrossEntropyLoss(label_smoothing=opt.label_smoothing)  # loss
function
    best_fitness = 0.0
    scaler = amp.GradScaler(enabled=cuda)
    val = test_dir.stem  # 'val' or 'test'
    LOGGER.info(
        f'Image sizes {imgsz} train, {imgsz} test\n'
```

```
        f'Using {nw * WORLD_SIZE} dataloader workers\n'
        f"Logging results to {colorstr('bold', save_dir)}\n"
        f'Starting {opt.model} training on {data} dataset with
{nc} classes for {epochs} epochs...\n\n'

f"{'Epoch':>10}{'GPU_mem':>10}{'train_loss':>12}{f'{val}_loss':>12
}{'top1_acc':>12}{'top5_acc':>12}"
    )
    for epoch in range(epochs):  # loop over the dataset multiple
times
        tloss, vloss, fitness = 0.0, 0.0, 0.0  # train loss, val
loss, fitness
        model.train()
        if RANK != -1:
            trainloader.sampler.set_epoch(epoch)
        pbar = enumerate(trainloader)
        if RANK in {-1, 0}:
            pbar = tqdm(enumerate(trainloader),
total=len(trainloader), bar_format=TQDM_BAR_FORMAT)
        for i, (images, labels) in pbar:  # progress bar
            images, labels = images.to(device, non_blocking=True),
labels.to(device)

            # Forward
            with amp.autocast(enabled=cuda):  # stability issues
when enabled
                loss = criterion(model(images), labels)

            # Backward
            scaler.scale(loss).backward()

            # Optimize
            scaler.unscale_(optimizer)  # unscale gradients
            torch.nn.utils.clip_grad_norm_(model.parameters(),
max_norm=10.0)  # clip gradients
            scaler.step(optimizer)
            scaler.update()
            optimizer.zero_grad()
            if ema:
```

```
                ema.update(model)

            if RANK in {-1, 0}:
                # Print
                tloss = (tloss * i + loss.item()) / (i + 1)  #
update mean losses
                mem = "%.3gG" % (torch.cuda.memory_reserved() /
1e9 if torch.cuda.is_available() else 0)  # (GB)
                pbar.desc = f"{f'{epoch +
1}/{epochs}':>10}{mem:>10}{tloss:>12.3g}" + " " * 36

                # Test
                if i == len(pbar) - 1:  # last batch
                    top1, top5, vloss = validate.run(
                        model=ema.ema, dataloader=testloader,
criterion=criterion, pbar=pbar
                    )  # test accuracy, loss
                    fitness = top1  # define fitness as top1
accuracy

        # Scheduler
        scheduler.step()

        # Log metrics
        if RANK in {-1, 0}:
            # Best fitness
            if fitness > best_fitness:
                best_fitness = fitness

            # Log
            metrics = {
                "train/loss": tloss,
                f"{val}/loss": vloss,
                "metrics/accuracy_top1": top1,
                "metrics/accuracy_top5": top5,
                "lr/0": optimizer.param_groups[0]["lr"],
            }  # learning rate
            logger.log_metrics(metrics, epoch)
```

```
            # Save model
            final_epoch = epoch + 1 == epochs
            if (not opt.nosave) or final_epoch:
                ckpt = {
                    "epoch": epoch,
                    "best_fitness": best_fitness,
                    "model": deepcopy(ema.ema).half(),  #
deepcopy(de_parallel(model)).half(),
                    "ema": None,  # deepcopy(ema.ema).half(),
                    "updates": ema.updates,
                    "optimizer": None,  # optimizer.state_dict(),
                    "opt": vars(opt),
                    "git": GIT_INFO,  # {remote, branch, commit}
if a git repo
                    "date": datetime.now().isoformat(),
                }

                # Save last, best and delete
                torch.save(ckpt, last)
                if best_fitness == fitness:
                    torch.save(ckpt, best)
                del ckpt

    # Train complete
    if RANK in {-1, 0} and final_epoch:
        LOGGER.info(
            f'\nTraining complete ({(time.time() - t0) / 3600:.3f}
hours)'
            f"\nResults saved to {colorstr('bold', save_dir)}"
            f'\nPredict:         python classify/predict.py --
weights {best} --source im.jpg'
            f'\nValidate:        python classify/val.py --weights
{best} --data {data_dir}'
            f'\nExport:          python export.py --weights {best}
--include onnx'
            f"\nPyTorch Hub:     model =
torch.hub.load('ultralytics/yolov5', 'custom', '{best}')"
            f'\nVisualize:       https://netron.app\n'
        )
```

```
        # Plot examples
        images, labels = (x[:25] for x in next(iter(testloader)))
# first 25 images and labels
        pred = torch.max(ema.ema(images.to(device)), 1)[1]
        file = imshow_cls(images, labels, pred,
de_parallel(model).names, verbose=False, f=save_dir /
"test_images.jpg")

        # Log results
        meta = {"epochs": epochs, "top1_acc": best_fitness,
"date": datetime.now().isoformat()}
        logger.log_images(file, name="Test Examples (true-
predicted)", epoch=epoch)
        logger.log_model(best, epochs, metadata=meta)
```

Lampiran nomor 5:

Perhitungan Uji Shapiro Wilk Pengujian Model Inception-v4

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.4525 + 0.4435 + 0.4395 + 0.4465 + 0.4355}{5}$$

$$= 0.44$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$

$$= 0.00017$$

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(n+1-i)} - X_i))^2}{\sum_{i=1}^{n} (X_i - \bar{X})^2}$$

$$= \frac{(0.0112982 + 0.0016891)^2}{0.00017}$$

$$= 0.992$$

Lampiran nomor 6:

Perhitungan Uji Shapiro Wilk Pengujian Model YOLO v5Nano

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.739, +0.708 + \ 0.731 + \ 0.728 + \ 0.727}{5}$$

$$= 0.7266$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$

$$= 0.0005212$$

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(n+1-i)} - X_i))^2}{\sum_{i=1}^{n} (X_i - \bar{X})^2}$$

$$= \frac{(0.0206026 + 0.0009652)^2}{0.0005212}$$

$$= 0.892498$$

Lampiran Nomor 7:

Perhitungan Uji Mann-Whitney U Pengujian Model ResNet-50 dan Inception-v4

Data model:

ResNet-50: 0.407, 0.4085, 0.409, 0.41, 0.418

Inception-v4: 0.4355, 0.4395, 0.4435, 0.4465, 0.4525

Peringkat Sampel Model:

ResNet-50: 1, 2, 3, 4, 5

Inception-v4: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

ResNet-50: 15

Inception-v4: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 8:

Perhitungan Uji Mann-Whitney U Pengujian Model Inception-v4 dan VGG-19

Data model:

Inception-v4: 0.4355, 0.4395, 0.4435, 0.4465, 0.4525

VGG-19: 0.1875, 0.1915, 0.2135, 0.2165, 0.72

Peringkat Sampel Model:

Inception: 5, 6, 7, 8, 9

VGG-19: 1, 2, 3, 4, 10

Total Peringkat masing-masing kelompok:

Inception-v4: 35

VGG-19: 20

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 35 = 5$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 20 = 20$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 20 = 5$$

Lampiran Nomor 9:

Perhitungan Uji Mann-Whitney U Pengujian Model Inception-v4 dan YOLO v5Nano

Data model:

Inception-v4: 0.4355, 0.4395, 0.4435, 0.4465, 0.4525

YOLO v5Nano: 0.708, 0.727, 0.728, 0.729, 0.739

Peringkat Sampel Model:

Inception-v4: 1, 2, 3, 4, 5

YOLO v5Nano: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

Inception-v4: 15

YOLO v5Nano: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 10:

Perhitungan Uji Mann-Whitney U Pengujian Model VGG-19 dan YOLO v5Nano

Data model:

VGG-19: 0.1875, 0.1915, 0.2135, 0.2165, 0.72

YOLO v5Nano: 0.708, 0.727, 0.728, 0.729, 0.739

Peringkat Sampel Model:

VGG-19: 1, 2, 3, 4, 6

YOLO v5Nano: 5, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

VGG-19: 16

YOLO v5Nano: 39

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 16 = 24$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 39 = 1$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 24 = 1$$

Lampiran Nomor 11:

Perhitungan Uji Shapiro-Wilk Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping*

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.57 + 0.536 + 0.536 + 0.54 + 0.546}{5}$$
$$= 0.5456$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$
$$= 0.000811$$

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(5+1-i)} - X_i))^2}{\sum_{i=1}^{5} (X_i - \bar{X})^2}$$
$$= \frac{(0.022596 + 0.002413)^2}{0.000811}$$
$$= 0.77$$

Lampiran Nomor 12:

Perhitungan Uji Shapiro-Wilk Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Histeq

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.442 + 0.45 + 0.462 + 0.462 + 0.468}{5}$$
$$= 0.4568$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$
$$= 0.000445$$

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(5+1-i)} - X_i))^2}{\sum_{i=1}^{5} (X_i - \bar{X})^2}$$
$$= \frac{(0.01728 + 0.002896)^2}{0.000455}$$
$$= 0.92$$

Lampiran Nomor 13:

Perhitungan Uji Shapiro-Wilk Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* dan Histeq

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.456 + 0.538 + 0.54 + 0.56 + 0.568}{5}$$
$$= 0.5324$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$
$$= 0.007955$$

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(5+1-i)} - X_i))^2}{\sum_{i=1}^{5} (X_i - \bar{X})^2}$$
$$= \frac{(0.074435 + 0.005309)^2}{0.007955}$$
$$= 0.799361$$

Lampiran Nomor 14:

Perhitungan Uji Shapiro-Wilk Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* dan *Flipping*

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.81 + 0.813 + 0.813 + 0.818 + 0.823}{5}$$
$$= 0.8154$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$
$$= 0.000105$$

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(5+1-i)} - X_i))^2}{\sum_{i=1}^{5} (X_i - \bar{X})^2}$$
$$= \frac{(0.00864 + 0.001207)^2}{0.000105}$$
$$= 0.921574$$

Lampiran Nomor 15:

Perhitungan Uji Shapiro-Wilk Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Histeq dan *Flipping*

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.682 + 0.687 + 0.692 + 0.695 + 0.697}{5}$$
$$= 0.6906$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$
$$= 0.000149$$

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(5+1-i)} - X_i))^2}{\sum_{i=1}^{5} (X_i - \bar{X})^2}$$
$$= \frac{(0.009969 + 0.00193)^2}{0.000105}$$
$$= 0.949033$$

Lampiran Nomor 16:

Perhitungan Uji Shapiro-Wilk Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Setelah *Preprocessing* dan Augmentasi

$$\bar{X} = \frac{1}{5}\sum_{i=1}^{5} X_i = \frac{0.708 + 0.727 + 0.728 + 0.731 + 0.739}{5}$$
$$= 0.7266$$

$$S^2 = \sum_{i=1}^{5} (X_i - \bar{X})^2$$
$$= 0.000521$$

$$a_1 = 0.6646, a_2 = 0.2413$$

$$W = \frac{(\sum_{i=1}^{2} a_i (X_{(5+1-i)} - X_i))^2}{\sum_{i=1}^{5} (X_i - \bar{X})^2}$$
$$= \frac{(0.020603 + 0.000965)^2}{0.000521}$$
$$= 0.892498$$

Lampiran Nomor 17:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Asli dan Model Histeq

Data model:

asli: 0.578, 0.589, 0.666, 0.674, 0.694

histeq: 0.536, 0.536, 0.54, 0.546, 0.57

Peringkat Sampel Model:

asli: 6, 7, 8, 9, 10

histeq: 1, 2, 3, 4, 5

Total Peringkat masing-masing kelompok:

asli: 40

histeq: 15

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 18:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Asli dan Model *Flipping*

Data model:

asli: 0.578, 0.589, 0.666, 0.674, 0.694

flipping: 0.913, 0.934, 0.935, 0.935, 0.935

Peringkat Sampel Model:

asli: 1, 2, 3, 4, 5

flipping: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

asli: 15

flipping: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 19:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Asli dan Model *Cropping* dan Histeq

Data model:

asli: 0.578, 0.589, 0.666, 0.674, 0.694

cropping dan histeq: 0.456, 0.538, 0.54, 0.56, 0.568

Peringkat Sampel Model:

asli: 6, 7, 8, 9, 10

flipping: 1, 2, 3, 4, 5

Total Peringkat masing-masing kelompok:

asli: 40

flipping: 15

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 20:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Asli dan Model *Cropping* dan *Flipping*

Data model:

asli: 0.578, 0.589, 0.666, 0.674, 0.694

cropping dan histeq: 0.81, 0.813, 0.813, 0.818, 0.823

Peringkat Sampel Model:

asli: 1, 2, 3, 4, 5

flipping: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

asli: 15

flipping: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 20:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Asli dan Model Setelah *Preprocessing* dan Augmentasi

Data model:

asli: 0.578, 0.589, 0.666, 0.674, 0.694

setelah *preprocessing* dan augmentasi: 0.708, 0.727, 0.728, 0.731, 0.739

Peringkat Sampel Model:

asli: 1, 2, 3, 4, 5

setelah *preprocessing* dan augmentasi: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

asli: 15

setelah *preprocessing* dan augmentasi: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 21:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* dan Model Histeq

Data model:

cropping: 0.536, 0.536, 0.54, 0.546, 0.57

histeq: 0.442, 0.45, 0.462, 0.462, 0.468

Peringkat Sampel Model:

cropping: 6, 7, 8, 9, 10

histeq: 1, 2, 3, 4, 5

Total Peringkat masing-masing kelompok:

cropping: 40

histeq: 15

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 22:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* dan Model *Flipping*

Data model:

cropping: 0.536, 0.536, 0.54, 0.546, 0.57

flipping: 0.913, 0.934, 0.935, 0.935, 0.935

Peringkat Sampel Model:

cropping: 1, 2, 3, 4, 5

flipping: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

cropping: 15

flipping: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 23:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* dan Model *Cropping* Histeq

Data model:

cropping: 0.536, 0.536, 0.54, 0.546, 0.57

cropping histeq: 0.456, 0.538, 0.54, 0.56, 0.568

Peringkat Sampel Model:

cropping: 2, 3, 5.5, 7, 10

cropping histeq: 1, 4, 5.5, 8, 9

Total Peringkat masing-masing kelompok:

cropping: 27.5

histeq: 27.5

$$U = n_1 \times n_2 + \frac{n_1(n_1 + 1)}{2} - R_1$$

$$U = 25 + \frac{5(5 + 1)}{2} - 27.5 = 12.5$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 12.5 = 12.5$$

Lampiran Nomor 24:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* dan Model *Cropping Flipping*

Data model:

cropping: 0.536, 0.536, 0.54, 0.546, 0.57

cropping flipping: 0.81, 0.813, 0.813, 0.818, 0.823

Peringkat Sampel Model:

cropping: 1, 2, 3, 4, 5

cropping flipping: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

cropping: 15

cropping flipping: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 25:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* dan Model Histeq *Flipping*

Data model:

cropping: 0.536, 0.536, 0.54, 0.546, 0.57

histeq *flipping*: 0.682, 0.687, 0.692, 0.695, 0.697

Peringkat Sampel Model:

cropping: 1, 2, 3, 4, 5

histeq *flipping*: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

cropping: 15

histeq *flipping*: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 26:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* dan Model Setelah *Preprocessing* dan Augmentasi

Data model:

cropping: 0.536, 0.536, 0.54, 0.546, 0.57

setelah *preprocessing* dan augmentasi: 0.708, 0.727, 0.728, 0.731, 0.739

Peringkat Sampel Model:

cropping: 1, 2, 3, 4, 5

setelah *preprocessing* dan augmentasi: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

cropping: 15

setelah *preprocessing* dan augmentasi: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 27:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Histeq dan Model *Flipping*

Data model:

histeq: 0.442, 0.45, 0.462, 0.462, 0.468

flipping: 0.913, 0.934, 0.935, 0.935, 0.935

Peringkat Sampel Model:

histeq: 1, 2, 3, 4, 5

flipping: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

histeq: 15

flipping: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 28:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Histeq dan Model *Cropping* Histeq

Data model:

histeq: 0.442, 0.45, 0.462, 0.462, 0.468

flipping: 0.456, 0.538, 0.54, 0.56, 0.568

Peringkat Sampel Model:

histeq: 1, 2, 4, 5, 6

flipping: 3, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

histeq: 20

flipping: 35

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 20 = 20$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 35 = 5$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 20 = 5$$

Lampiran Nomor 29:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Histeq dan Model *Cropping Flipping*

Data model:

histeq: 0.442, 0.45, 0.462, 0.462, 0.468

cropping flipping: 0.81, 0.813, 0.813, 0.818, 0.823

Peringkat Sampel Model:

histeq: 1, 2, 3, 4, 5

cropping flipping: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

histeq: 15

cropping flipping: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 30:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Histeq dan Model Histeq *Flipping*

Data model:

histeq: 0.442, 0.45, 0.462, 0.462, 0.468

histeq *flipping*: 0.682, 0.687, 0.692, 0.695, 0.697

Peringkat Sampel Model:

histeq: 1, 2, 3, 4, 5

histeq *flipping*: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

histeq: 15

histeq *flipping*: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 31:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Histeq dan Model Setelah *Preprocessing* dan Augmentasi

Data model:

histeq: 0.442, 0.45, 0.462, 0.462, 0.468

setelah *preprocessing* dan augmentasi: 0.708, 0.727, 0.728, 0.731, 0.739

Peringkat Sampel Model:

histeq: 1, 2, 3, 4, 5

setelah *preprocessing* dan augmentasi: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

histeq: 15

setelah *preprocessing* dan augmentasi: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 32:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Flipping* dan Model *Cropping* Histeq

Data model:

flipping: 0.913, 0.934, 0.935, 0.935, 0.935

cropping histeq: 0.456, 0.538, 0.54, 0.56, 0.568

Peringkat Sampel Model:

flipping: 6, 7, 8, 9, 10

cropping histeq: 1, 2, 3, 4, 5

Total Peringkat masing-masing kelompok:

flipping: 40

cropping histeq: 15

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 33:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Flipping* dan Model *Cropping Flipping*

Data model:

flipping: 0.913, 0.934, 0.935, 0.935, 0.935

cropping flipping: 0.81, 0.813, 0.813, 0.818, 0.823

Peringkat Sampel Model:

flipping: 6, 7, 8, 9, 10

cropping flipping: 1, 2, 3, 4, 5

Total Peringkat masing-masing kelompok:

flipping: 40

cropping flipping: 15

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 34:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Flipping* dan Model Histeq *Flipping*

Data model:

flipping: 0.913, 0.934, 0.935, 0.935, 0.935

histeq *flipping*: 0.682, 0.687, 0.692, 0.695, 0.697

Peringkat Sampel Model:

flipping: 6, 7, 8, 9, 10

histeq *flipping*: 1, 2, 3, 4, 5

Total Peringkat masing-masing kelompok:

flipping: 40

histeq *flipping*: 15

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 35:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Flipping* dan Model Setelah *Preprocessing* dan Augmentasi

Data model:

flipping: 0.913, 0.934, 0.935, 0.935, 0.935

setelah *preprocessing* dan augmentasi: 0.708, 0.727, 0.728, 0.731, 0.739

Peringkat Sampel Model:

flipping: 6, 7, 8, 9, 10

setelah *preprocessing* dan augmentasi: 1, 2, 3, 4, 5

Total Peringkat masing-masing kelompok:

flipping: 40

setelah *preprocessing* dan augmentasi: 15

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 36:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* Histeq dan Model *Cropping Flipping*

Data model:

cropping histeq: 0.456, 0.538, 0.54, 0.56, 0.568

cropping flipping: 0.81, 0.813, 0.813, 0.818, 0.823

Peringkat Sampel Model:

cropping histeq: 1, 2, 3, 4, 5

cropping flipping: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

cropping histeq: 15

cropping flipping: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 37:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* Histeq dan Model Histeq *Flipping*

Data model:

cropping histeq: 0.456, 0.538, 0.54, 0.56, 0.568

histeq *flipping*: 0.682, 0.687, 0.692, 0.695, 0.697

Peringkat Sampel Model:

cropping histeq: 1, 2, 3, 4, 5

histeq *flipping*: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

cropping histeq: 15

histeq *flipping*: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 38:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping* Histeq dan Model Setelah *Preprocessing* dan Augmentasi

Data model:

cropping histeq: 0.456, 0.538, 0.54, 0.56, 0.568

setelah *preprocessing* dan augmentasi: 0.708, 0.727, 0.728, 0.731, 0.739

Peringkat Sampel Model:

cropping histeq: 1, 2, 3, 4, 5

setelah *preprocessing* dan augmentasi: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

cropping histeq: 15

setelah *preprocessing* dan augmentasi: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 39:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping Flipping* dan Model Histeq *Flipping*

Data model:

cropping flipping: 0.81, 0.813, 0.813, 0.818, 0.823

histeq *flipping*: 0.682, 0.687, 0.692, 0.695, 0.697

Peringkat Sampel Model:

cropping flipping: 6, 7, 8, 9, 10

histeq *flipping*: 1, 2, 3, 4, 5

Total Peringkat masing-masing kelompok:

cropping flipping: 40

histeq *flipping*: 15

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 40:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model *Cropping Flipping* dan Model Setelah *Preprocessing* dan Augmentasi

Data model:

cropping flipping: 0.81, 0.813, 0.813, 0.818, 0.823

setelah *preprocessing* dan augmentasi: 0.708, 0.727, 0.728, 0.731, 0.739

Peringkat Sampel Model:

cropping flipping: 6, 7, 8, 9, 10

setelah *preprocessing* dan augmentasi: 1, 2, 3, 4, 5

Total Peringkat masing-masing kelompok:

cropping flipping: 40

setelah *preprocessing* dan augmentasi: 15

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$

Lampiran Nomor 40:

Perhitungan Uji Mann-Whitney Pengujian Model Analisis *Preprocessing* dan Augmentasi Data Model Histeq *Flipping* dan Model Setelah *Preprocessing* dan Augmentasi

Data model:

histeq *flipping*: 0.682, 0.687, 0.692, 0.695, 0.697

setelah *preprocessing* dan augmentasi: 0.708, 0.727, 0.728, 0.731, 0.739

Peringkat Sampel Model:

histeq *flipping*: 1, 2, 3, 4, 5

setelah *preprocessing* dan augmentasi: 6, 7, 8, 9, 10

Total Peringkat masing-masing kelompok:

histeq *flipping*: 15

setelah *preprocessing* dan augmentasi: 40

$$U = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

$$U = 25 + \frac{5(5+1)}{2} - 15 = 25$$

atau

$$U = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

$$U = 25 + \frac{5(5+1)}{2} - 40 = 0$$

$$p - value = n_1 \times n_2 - U_{terbesar} = 25 - 25 = 0$$