



LAPORAN PROYEK AKHIR

KLASIFIKASI PENYAKIT RETINOPATI DIABETIK MENGGUNAKAN CNN DENGAN MULTIMODAL DATA FUSION PADA DATASET CITRA RETINA

AHMAD ALI MUNAWAR
NIM. 2055301003

Pembimbing
Kartina Diah Kesuma Wardhani, S.T., M.T.

PROGRAM STUDI TEKNIK INFORMATIKA
POLITEKNIK CALTEX RIAU
2024

LAPORAN PROYEK AKHIR

**KLASIFIKASI PENYAKIT RETINOPATI
DIABETIK MENGGUNAKAN CNN DENGAN
MULTIMODAL DATA FUSION PADA
DATASET CITRA RETINA**

AHMAD ALI MUNAWAR
NIM. 2055301003

Pembimbing
Kartina Diah Kesuma Wardhani, S.T., M.T.

**PROGRAM STUDI TEKNIK INFORMATIKA
POLITEKNIK CALTEX RIAU
2024**

HALAMAN PENGESAHAN

**KLASIFIKASI PENYAKIT RETINOPATI DIABETIK
MENGGUNAKAN CNN DENGAN MULTIMODAL DATA
FUSION PADA DATASET CITRA RETINA**

AHMAD ALI MUNAWAR

NIM. 2055301003

Proyek Akhir ini diajukan sebagai salah satu persyaratan untuk
memperoleh gelar Sarjana Terapan Komputer (S.Tr.Kom)
di Politeknik Caltex Riau

Pekanbaru, 15 Agustus 2024

Disetujui oleh:

Pembimbing,

Kartina Diah Kesuma Wardhani,
S.T., M.T.
NIP. 078310

Pengaji,

1. Memen Akbar, S.Si, M.T.
NIP. 078313

2. Dr. Juni Nurma Sari, S.Kom.,
M.MT.
NIP. 017218

Mengetahui,

Ketua Program Studi Teknik Informatika

Silvana Rasib Henim, S.ST, M.T.
NIP. 068407

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam proyek akhir yang berjudul:

“KLASIFIKASI PENYAKIT RETINOPATI DIABETIK MENGGUNAKAN CNN DENGAN MULTIMODAL DATA FUSION PADA DATASET CITRA RETINA”

Adalah benar hasil karya saya, dan tidak mengandung karya ilmiah atau tulisan yang pernah diajukan di suatu Perguruan Tinggi.

Setiap kata yang dituliskan tidak mengandung plagiat, pernah ditulis maupun diterbitkan orang lain kecuali yang secara tertulis diacu dalam laporan proyek akhir ini dan disebutkan pada daftar pustaka. Saya siap menanggung seluruh akibat apabila terbukti melakukan plagiat.

Pekanbaru, 24 Juli 2024

Ahmad Ali Munawar

ABSTRAK

Diabetes merupakan masalah kesehatan global yang serius dan dapat menyebabkan berbagai komplikasi, salah satunya adalah Retinopati Diabetik (RD), sebuah kondisi yang terjadi ketika tingkat gula darah yang tinggi merusak pembuluh darah di retina mata. Penderita diabetes perlu melakukan pemeriksaan rutin pada retina untuk mendeteksi RD secara dini. Teknik klasifikasi seperti *Convolutional Neural Networks* (CNN) telah menjanjikan dalam mendiagnosis RD secara otomatis dan akurat. Namun, tantangan muncul karena hanya menggunakan satu dataset dalam klasifikasi citra retina mungkin tidak menghasilkan prediksi yang tepat. Oleh karena itu, penelitian ini bertujuan untuk meningkatkan akurasi dan efektivitas diagnosis otomatis RD dengan menggabungkan dua jenis citra retina dari Fundus Photography dan Optical Coherence Tomography (OCT) dengan teknik early fusion menggunakan concatenate dari library numpy dan alpha blending, yang dilatih menggunakan model CNN. Dataset fusion dari citra fundus dan OCT yang dilatih menggunakan model CNN memiliki akurasi tertinggi mencapai 92%, specificity 96%, sensitivity 98%, AUC 99%, dan F1-score 96%, dengan learning rate 0.0001 dan epoch 100. Berdasarkan penelitian ini, fusi citra dari fundus dan OCT dengan early fusion dapat menghasilkan klasifikasi penyakit Retinopati Diabetik berdasarkan kelasnya, yaitu RD dan Normal, dengan lebih akurat.

Kata kunci: *Retinopati Diabetik, Convolutional Neural Networks (CNN), multimodal data fusion, Fundus Photography, Optical Coherence Tomography (OCT), alpha blending, concatenate, image classification, retinal image classification.*

ABSTRACT

Diabetes is a serious global health issue that can lead to various complications, one of which is Diabetic Retinopathy (DR), a condition where high blood sugar levels damage the blood vessels in the retina. Diabetic patients need to undergo regular retinal examinations to detect DR early. Classification techniques such as Convolutional Neural Networks (CNN) have shown promise in automatically and accurately diagnosing DR. However, challenges arise as using a single dataset in retinal image classification may not yield accurate predictions. Therefore, this study aims to enhance the accuracy and effectiveness of automatic DR diagnosis by combining two types of retinal images from Fundus Photography and Optical Coherence Tomography (OCT) using an early fusion technique with concatenation from the numpy library and alpha blending, trained with a CNN model. The fusion dataset from fundus and OCT images trained using the CNN model achieved a highest accuracy of 92%, specificity of 96%, sensitivity of 98%, AUC of 99%, and an F1-score of 96%, with a learning rate of 0.0001 and 100 epochs. Based on this research, fusing fundus and OCT images with early fusion can result in a more accurate classification of Diabetic Retinopathy into its classes, namely DR and Normal.

Keywords: *Retinopati Diabetik, Convolutional Neural Networks (CNN), multimodal data fusion, Fundus Photography, Optical Coherence Tomography (OCT), alpha blending, concatenate, image classification, retinal image classification.*

KATA PENGANTAR

Segala puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan barokah-Nya sehingga penulis dapat menyelesaikan proyek akhir yang berjudul “KLASIFIKASI PENYAKIT RETINOPATI DIABETIK MENGGUNAKAN CNN DENGAN MULTIMODAL DATA FUSION PADA DATASET CITRA RETINA”. Proyek akhir ini disusun sebagai salah satu syarat untuk menyelesaikan jenjang pendidikan Diploma IV pada Program Studi Teknik Informatika Politeknik Caltex Riau.

Pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada pihak yang telah banyak memberikan bantuan dan dukungan yang tiada terhingga baik secara langsung maupun tidak langsung. Ucapan terima kasih tersebut penulis tujuhan kepada:

1. Allah SWT atas rahmat dan karunia-Nya, sehingga penulis bisa menyelesaikan tugas akhir ini tepat waktu.
2. Orang tua dan abang-abang penulis atas dukungan dan kasih sayang tak terhingga, sehingga penulis bisa menyelesaikan tugas akhir tepat waktu.
3. Dr. Dadang Syarif Sihabudin Sahid, S.Si,M.Sc.. selaku Direktur Politeknik Caltex Riau yang telah memberikan dukungan moral dalam menyelesaikan proyek akhir ini.
4. Ibu Silvana Rasio Henim, S.ST, M.T. selaku Ketua Program Studi Teknik Informatika yang telah memberikan izin untuk menyelesaikan proyek akhir.
5. Ibu Kartina Diah Kesuma Wardhani, S.T., M.T. selaku dosen pembimbing yang telah membimbing dan memberikan arahan serta bantuan untuk menyelesaikan proyek akhir dengan penuh kesabaran.
6. Bapak Muhammad Arif Fadhl Ridha, S.Kom., M.T. selaku koor PA yang selalu memberikan bimbingan, motivasi, dan membantu ketika penulis mengalami kesulitan.
7. Bapak Memen Akbar, S.Si, M.T. selaku dosen penguji 1 dan Ibu Dr. Juni Nurma Sari, S.Kom., M.MT. selaku dosen penguji 2 yang

telah membimbing dan memberikan ilmu serta saran-saran yang berguna.

8. Seluruh Dosen Program Studi Teknik Informatika yang telah memberikan bekal ilmu kepada penulis dalam menyelesaikan proyek akhir.
9. Teman-teman TIB G20 yang telah mensupport dan membantu mengerjakan Proyek Akhir ini. Dan juga teman-teman G20 senasib seperjuangan yang saling memberikan dukungan dan motivasi sehingga penulis dapat menyelesaikan Proyek Akhir dengan tepat waktu.

Penulis sangat menyadari sepenuhnya bahwa laporan proyek akhir ini masih jauh dari sempurna, oleh karena itu segala jenis kritik, saran dan masukan yang membangun sangat penulis harapkan agar dapat memberikan wawasan bagi pembaca dan yang paling utama penulis sendiri.

Pekanbaru, 24 Juli 2024

Ahmad Ali Munawar

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN	Error! Bookmark not defined.
PERNYATAAN	iii
ABSTRAK	iv
ABSTRACT	v
KATA PENGANTAR	vi
DAFTAR ISI	viii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	3
1.6 Metodologi Penelitian	4
1.7 Sistematika Penulisan	4
BAB II TINJAUAN PUSTAKA	6
2.1 Tinjauan Pustaka	6
2.2 Landasan Teori	11
2.2.1 Retinopati Diabetik (RD)	11
2.2.2 Pencitraan Retina <i>Fundus Photography</i>	11
2.2.3 <i>Optical Coherence Tomograph (OCT)</i>	12

2.2.4	<i>Deep Learning</i>	13
2.2.5	<i>Convolutional Neural Network (CNN)</i>	16
2.2.6	Multimodal Data vs Unimodal Data	17
2.2.7	Teknik Fusion	19
2.2.8	Augmentasi	20
2.2.9	Resize	21
2.2.10	Grayscale	22
2.2.11	<i>Alpha Blending</i>	22
2.2.12	<i>Concatenate</i>	23
2.2.13	Python	23
2.2.14	Jupyter	24
BAB III	PERANCANGAN	25
3.1	Tahapan Alur Klasifikasi Retinopati Diabetik	25
3.2	Langkah-langkah Klasifikasi Retinopati Diabetik	27
3.2.1	Dataset	27
3.2.2	Preposessing	27
3.2.3	Early Fusion	29
3.2.4	Modelling	30
3.2.5	Akurasi Model	32
3.2.6	Testing	33
BAB IV	PENGUJIAN DAN ANALISIS	35

4.1	Dataset	35
4.2	. Prossesing	36
	4.2.1 Augmentasi Dataset	36
	4.2.2 <i>Resize</i> Citra.	39
	4.2.3 <i>Grayscale</i> .	39
4.3	Early Fusion	40
	4.3.1 <i>Concatenate + Alpha Blending</i> .	40
	4.3.2 Pembagian Dataset.	44
4.4	Implementasi Model.	46
	4.4.1 Normalisasi	46
	4.4.2 Model CNN	47
	4.4.3 <i>Trainning</i> Model	50
4.5	Implementasi Pengujian	55
	4.5.1 Inputan Citra.	56
	4.5.2 Preprosessing.	56
	4.5.3 Inputan Citra.	58
4.6	Analisis	60
BAB V	KESIMPULAN DAN SARAN	61
5.1	Kesimpulan	61
5.2	Saran	61
DAFTAR PUSTAKA		62
LAMPIRAN		A

DAFTAR GAMBAR

Gambar 2. 1 Retinopati Diabetik.....	11
Gambar 2. 2 Hasil Pencitraan Retina Fundus Photography.	12
Gambar 2. 3 Hasil Pencitraan Retina optical coherence tomograph (OCT).	13
Gambar 2. 4 Ruang Lingkup Deep Learning	14
Gambar 2. 5 Perbandingan Multimodal Data vs Unimodal Data....	18
Gambar 2. 6 Ilustrasi Teknik Early Fusion.....	19
Gambar 2. 7 Augmentasi.....	20
Gambar 2. 8 Contoh Resize.....	21
Gambar 2. 9 Contoh Grayslce	22
Gambar 2. 10 Contoh <i>Alpha Blending</i>	23
Gambar 2. 11 Tampilan Jupyter	24
Gambar 3. 1 Diagram Alur Penelitian.....	26
Gambar 3. 2 Teknik Alpha Blending + Concatenate.....	29
Gambar 3. 3 Rancangan Arsitektur CNN.....	32
Gambar 3. 4 <i>Flowchart</i> proses <i>Testing Model</i>	34
Gambar 4. 1 Augmentasi Dataset.....	37
Gambar 4. 2 Proses Resize	39
Gambar 4. 3 Proses Grayscale.....	39
Gambar 4. 4 Preposesing Citra Fundus dan OCT.	40
Gambar 4. 5 Proses Early Fusion	41
Gambar 4. 6 Code Program Early Fusion.....	42
Gambar 4. 7 Pembagian Dataset.....	45
Gambar 4. 8 Proses Normalisasi.....	47
Gambar 4. 9 Arsitektur CNN.....	48
Gambar 4. 10 Kode Program CNN	49
Gambar 4. 11 80:20 Grafik Model Loss dan Accuracy dengan LR 0.001	51

Gambar 4. 12 80:20 Grafik Model Loss dan Accuracy dengan LR 0.0001	51
Gambar 4. 13 70:30 Grafik Model Loss dan Accuracy dengan LR 0.001	51
Gambar 4. 14 70:30 Grafik Model Loss dan Accuracy dengan LR 0.0001	52
Gambar 4. 15 60:40 Grafik Model Loss dan Accuracy dengan LR 0.001	52
Gambar 4. 16 60:40 Grafik Model Loss dan Accuracy dengan LR 0.0001	52
Gambar 4. 17 Proses Klasifikasi.....	57
Gambar 4. 18 Tampilan antarmuka klasifikasi Retinopati Diabetik.	58
Gambar 4. 19 Hasil Klasifikasi Retinopati Diabetik	59

DAFTAR TABEL

Table 2. 1 Penelitian terdahulu	7
Table 3. 1 Dataset Fundus.....	27
Table 3. 2 Dataset OCT	27
Tabel 4. 1 Dataset Fundus dan OCT.....	35
Tabel 4. 2 Hail Augmentasi OCT	38
Tabel 4. 3 Hasil Early Fusion	44
Tabel 4. 4 Distribusi data train dan test.....	46
Tabel 4. 5 Training Dataset	50
Tabel 4. 6 Validasi Akurasi	54

BAB I

PENDAHULUAN

1.1 Latar Belakang

Diabetes diperkirakan menempati urutan ketujuh yang termasuk penyakit fatal (Shankar et al., 2020). Kondisi ini telah menjadi epidemi global yang memerlukan perhatian serius. Diperkirakan sekitar sepertiga dari penderita diabetes akan mengalami komplikasi mata, dengan Retinopati Diabetik (RD) sebagai masalah serius yang muncul. Salah suatu komplikasi yang dapat terjadi pada mata akibat diabetes. Kondisi ini timbul ketika tingkat gula darah tinggi dalam jangka waktu yang lama menyebabkan kerusakan pada pembuluh darah di retina, lapisan di dalam mata yang peka terhadap cahaya (Izzati, 2023). Penderita RD pada tahapan awal memerlukan pengecekan pada retina direkomendasikan untuk pasien diabetes (Jena et al., 2023). RD dapat berkembang secara perlahan tanpa menimbulkan gejala pada tahap awal. Namun, seiring berjalannya waktu, kondisi ini dapat menyebabkan pembuluh darah retina bocor atau membengkak, yang dapat mengakibatkan kehilangan penglihatan (Duh, 2017), (Wu, 2013), (Qummar, 2019).

Untuk mendeteksi RD dibutuhkan teknik klasifikasi, seperti penggunaan algoritma Convolutional Neural Networks (CNN), CNN telah menjadi pendekatan yang menjanjikan dalam mendiagnosis RD secara otomatis dan akurat. CNN merupakan jenis arsitektur jaringan saraf tiruan yang secara khusus dirancang untuk memproses data citra secara efektif (Rupsa Ghosh, 2017). CNN adalah jenis algoritma komputer yang khusus dirancang untuk memproses gambar atau foto dengan cara yang mirip dengan cara otak manusia memproses informasi visual. CNN bisa belajar dari contoh-contoh gambar dan mengenali pola-pola yang kompleks dalam citra retina. Dengan teknologi ini, sistem klasifikasi dapat lebih mudah dan lebih akurat mendeteksi adanya dukungan dataset yang memumpuni sebagai sumber informasinya.

Dateset yang digunakan seperti Fundus Photography, dan Optical Coherence Tomography (OCT), dll. Fundus adalah kumpulan data yang terstruktur berisi gambar bagian dalam mata yang dapat dilihat melalui oftalmoskop (Dr. Anita Dhanorkar, 2024), sedangkan

OCT adalah sebuah teknik pencitraan medis yang menggunakan cahaya untuk menangkap gambar tiga dimensi beresolusi mikrometer dari dalam media hamburan optik seperti jaringan biologis. Teknik ini didasarkan pada interferometri koherensi rendah, dengan menggunakan cahaya inframerah dekat (M.D., 2023). Dataset tersebut dibutuhkan untuk mencapai prediksi yang lebih akurat juga mendapatkan gambaran detil tentang struktur mata dengan multimodal data untuk memfusion citra retina. Namun, tantangan muncul dalam klasifikasi data yang pada umumnya hanya menggunakan satu dataset.

Pada penelitian yang dilakukan oleh (Sagala, 2022) dengan dataset yang digunakan adalah fundus, dilakukan sebuah uji penelitian menggunakan algoritma Convolutional Neural Networks (CNN), hasil yang diperoleh pada penelitian tersebut mendapatkan nilai akurasi adalah 0.34. Pada penelitian lainnya yang dilakukan oleh (Dyah Retno Mutia, 2021) dengan dataset OCT, dilakukan uji penelitian menggunakan algoritma CNN, hasil yang diperoleh pada penelitian tersebut mendapatkan nilai akurasi adalah 0.87. Pada penelitian yang dijelaskan, dapat disimpulkan bahwa hanya menggunakan satu set data saja tidak menghasilkan prediksi yang sangat tepat. Hal ini terjadi karena data yang digunakan mungkin tidak sepenuhnya mencerminkan apa yang sebenarnya ingin kita klasifikasikan. Selain itu, model yang dilatih hanya dengan satu set data mungkin tidak dapat dengan baik mengenali atau memahami data baru yang belum pernah dilihat sebelumnya.

Sehingga dalam menjawab tantangan ini, Penelitian "Klasifikasi Penyakit Retinopati Diabetik Menggunakan CNN Dengan Multimodal Data Fusion Pada Dataset Citra Retina " bertujuan untuk menggabungkan teknologi citra retina dari citra Fundus Photography dan OCT dengan teknik early fusion. Fokusnya pada pengembangan dataset citra dari multimodal data fusion dengan algoritma CNN yang dapat memberikan prediksi RD dengan tingkat akurasi yang baik.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang ada, rumusan masalah dalam penelitian ini adalah sebagai berikut:

- 1) Bagaimana klasifikasi RD menggunakan algoritma CNN dari model dataset citra multimodal data fusion.
- 2) Bagaimana cara dataset baru dari multimodal data fusion dapat meningkatkan akurasi untuk klasifikasi RD.

1.3 Batasan Masalah

Adapun batasan masalah dalam proyek akhir ini adalah:

- 1) Penelitian akan menggunakan multimodal data citra retina, dari dataset *Fundus Photography* dan *OCT* sebagai sumber data utama untuk deteksi dan prediksi RD.
- 2) Dataset yang digunakan dalam penelitian diperoleh dari Kaggle ini berupa dataset *Diagnosis of Diabetic Retinopathy* yang didapat dari Kaggle (<https://www.kaggle.com/datasets/pkdarabi/diagnosis-of-diabetic-retinopathy>) yang berisi gambar *Fundus Photogr* dan *Retinal OCT Images (optical coherence tomography)* yang didapat dari Google Drive (https://drive.google.com/drive/folders/14OVRuOiFl2jICh54m17niU1RhtWX9SdU?usp=drive_link) berisi gambar *OCT*.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk menciptakan dataset baru dengan metode early fusion yang mengintegrasikan citra Fundus Photography dan OCT. Dataset early fusion tersebut akan digunakan untuk melatih model Convolutional Neural Network (CNN) dalam klasifikasi Retinopati Diabetik (RD).

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah:

- 1) Mengembangkan model dataset baru dari multimodal data fusion untuk mendukung deteksi dini dalam pencegahan Retinopati Diabetik (RD).
- 2) Menghasilkan solusi teknologi yang lebih relevan dengan menggabungkan teknologi citra retina dan deep learning, untuk meningkatkan akurasi dan efisiensi dalam proses klasifikasi.

1.6 Metodologi Penelitian

Metode penelitian yang dipakai dalam pembuatan proyek akhir ini adalah:

1) Studi Literatur

Dilakukan studi untuk mengumpulkan bahan-bahan referensi dan mempelajarinya baik dari buku, jurnal, paper, makalah, maupun situs internet.

2) Pengumpulan Data

Data yang digunakan dari data citra retina dari teknologi Fundus Photography. Sedangkan dataset yang digunakan adalah Diabetic Diagnosis of Diabetic Retinopathy dan Retinal OCT Images (optical coherence tomography) yang diperoleh dari Kaggle dan Google Drive.

3) Perancangan

Metode ini dilakukan untuk merancang klasifikasi RD. yang diamana pada pemodelan dataset yang digunakan berupa citra retina dari teknologi Fundus Photography dan OCT dengan menggunakan algoritma CNN sebagai metode klasifikasi untuk menghasilkan citra retina Early Fusion yang dapat di klasifikasikan hasil citra fusi retina.

4) Implementasi

Pada tahapan ini dataset yang digunakan masih belum dibagi menjadi data Train dan data Testing, sehingga diperlukan sebuah proses untuk membagi dataset sesuai dengan tingakatan RD. rasio perbandingan yang digunakan untuk data train dan test yaitu 80:20, 70:30, 60:40.

5) Pengujian dan Analisa

Pada tahap ini akan dilakukan pengujian dan analisa terhadap system yang akan digunakan agar system tersebut bekerja dengan baik.

6) Penulisan Laporan

Hasil penelitian ini didokumentasikan dalam bentuk bakunya yang telah diatur oleh pihak Politeknik Caltex Riau yang terdapat pada Buku Panduan Proyek Akhir Tahun 2017.

1.7 Sistematika Penulisan

Sistematika penulisan laporan proyek akhir ini secara keseluruhan terdiri dari empat bab, masing-masing terdiri dari

beberapa sub bab. Adapun pokok pembahasan dari masing-masing bab tersebut secara garis besar sebagai berikut:

BAB I PENDAHULUAN

Bab ini menguraikan tentang latar belakang masalah, perumusan masalah dan ruang lingkup masalah, tujuan dan manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini menguraikan beberapa hasil penelitian terdahulu dan landasan teori yang diperlukan untuk merancang sistem.

BAB III PERANCANGAN

Bab ini menjelaskan tentang perancangan sistem terdiri dari tahap Dataset, Prosessing, Early Fusion, Modelling, dan Evaluasi.

BAB IV JADWAL DAN PERKIRAAN BIAYA

Bab ini berisi informasi mengenai jadwal pengerjaan proyek akhir dan perkiraan biaya yang dibutuhkan untuk pengerjaan proyek akhir.

BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Penelitian terdahulu merupakan penelitian yang sudah ada sebelumnya dan memeliki objek atau topik yang sama dan relevan dengan yang diteliti oleh penulis saat ini. Penelitian terdahulu memungkinkan penulis untuk membandingkan atau melihat sudut pandang yang diteliti. Dalam penelitian ini, penulis mengambil lima penelitian terdahulu yang sebelumnya telah dilakukan pengujian.

Pada penelitian pertama. Multimodal Information Fusion for Glaucoma and Diabetic Retinopathy Classification, Penelitian ini menggunakan Deep learning dengan pemodelan Early Fusion dengan arsitektur DenseNet121. Model ini dilatih dengan dataset dari citra fundus dan OCT. Hasil penelitian dari model ini mencapai mencapai AUC 0.78 dalam klasifikasi diabetik retinopathy

Pada penelitian kedua. A Multimodal Imaging-Based Deep Learning Model for Detecting Treatment-Requiring Retinal Vascular Diseases: Model Development and Validation Study, Penelitian ini menggunakan model CNN dengan arsitektur EfficientNetB4 untuk mengklasifikasikan RD. Model ini dilatih pada dataset citra fundus, OCT, dan FA. Hasil penelitian menunjukkan bahwa model ini dapat mencapai AUC 0.969 dalam mengklasifikasikan RD.

Pada penelitian ketiga. Deep Fusion of Fundus and OCT Images for Diabetic Retinopathy Classification, Penelitian ini menggunakan model Deep Fusion Network yang telah dimodifikasi untuk mengklasifikasikan RD. Model ini dilatih pada Dataset terdiri dari 1000 pasien dengan diabetik retinopathy. Ynag dimana setiap pasien memiliki citra fundus dan OCT, citra fundus dan OCT dibagi menjadi dataset pelatihan, validasi, dan uji. Hasil penelitian dari model ini mencapai AUC 0.96 dalam klasifikasi diabetik retinopathy. Hasil ini lebih baik dibandingkan dengan model yang hanya menggunakan citra fundus (AUC 0.89) atau citra OCT (AUC 0.92).

Pada penelitian keempat. Multimodal Diabetic Retinopathy Classification Using Fundus, OCT, and FA Images. Penelitian ini menggunakan multimodal fusion network untuk menggabungkan informasi dari citra fundus, OCT, dan FA untuk klasifikasi diabetik

retinopathy. Multimodal fusion network terdiri dari empat subnetwork yaitu subnetwork fundus, subnetwork OCT, subnetwork FA, subnetwork fusi. Hasil penelitian model ini mencapai AUC 0.97 dalam klasifikasi diabetik retinopathy. Hasil ini lebih baik dibandingkan dengan model yang hanya menggunakan citra fundus (AUC 0.89), citra OCT (AUC 0.92), atau citra FA (AUC 0.94).

Pada penelitian kelima. Joint Representation Learning of Fundus and OCT Images for Diabetic Retinopathy Classification, Penelitian ini menggunakan model Joint Representation Learning Network untuk mengklasifikasikan RD. Joint representation learning memungkinkan model untuk menangkap hubungan halus dan komplementer antara informasi fundus dan OCT. Citra fundus memberikan gambaran struktur retina secara global, sedangkan OCT memberikan perspektif mendalam tentang ketebalan dan lapisan retina. Dengan belajar bersama, model dapat memanfaatkan kedua sumber informasi ini secara sinergis, mengungkap pola yang sulit dilihat model sebelumnya. Hasil penelitian menunjukkan bahwa model Joint Representation Learning Network dapat mencapai AUC 0,98 dalam klasifikasi diabetik retinopathy.

Table 2. 1 Penelitian terdahulu

Judul Penelitian	Datas et	Algoritma	Out put	Kekurangan	Kelebihan
Multimodal Information Fusion for Glaucoma and Diabetic Retinopathy Classification (Yihao	Fundus, dan OCT	Deep learning dengan pemodelan Early Fusion (DenseNet 121)	AU C 0.78	Dataset yang digunakan dalam penelitian ini relatif kecil.	Penelitian ini mendapatkan hasil yang bagus dalam hal akurasi klasifikasi glaukoma dan

Li, 2022)					retinopati diabetik.
A Multimodal Imaging –Based Deep Learning Model for Detecting Treatment-Requiring Retinal Vascular Diseases: Model Development and Validation Study (Eugene Yu-Chuan Kang, 2019)	Fundus, OCT, dan fluorescein angiography (FA)	Convolutional neural network (CNN) dengan arsitektur EfficientNetB4	AUC 0.969	Dataset yang digunakan dalam penelitian ini relatif kecil.	Penelitian ini menggunakan heat maps untuk memvisualisasikan model sehingga dapat mengidentifikasi penyakit vaskular retina.
Deep Fusion of	Fundus,	Deep Fusion Network	AUC 0.96	Penelitian ini tidak membanding	Penelitian ini memanfaat

Fundus and OCT Images for Diabetic Retinopathy Classification (Hongyu Zhou, 2022)	dan OCT			gkan kinerja model fusi dengan model yang hanya menggunakan satu jenis citra.	kan fitur spasial dan spektral dari kedua jenis citra
Multimodal Diabetic Retinopathy Classification Using Fundus, OCT, and FA Images (Yuxin Chen, 2023)	Fundus, OCT, dan FA	Multimodal Fusion Network	AUC 0.97	Penelitian ini hanya menggunakan satu dataset yang tersedia (APROS2019) sebagai sumber data validasi.	Penelitian ini memanfaatkan setiap modality dari citra (fundus, OCT, dan FA) memperkaya informasi dan meningkatkan performa klasifikasi.
Joint Representation Learning of Fundus and	Fundus, dan OCT	Joint Representation Learning Network	AUC 0.98	Dataset yang digunakan dalam penelitian ini relatif kecil	penelitian ini berhasil mengidentifikasi dan memetakan lesi pada

OCT Images for Diabetic Retinopathy Classification (Yifan Peng, 2023)					permukaan retina.
Klasifikasi Penyakit Rerinopati Diabetik Menggunakan CNN dengan Multimodal Data Fusion Pada Dataset Citra Retina (Penelitian Saat ini)	Fundus, dan OCT	Convolutional neural network (CNN)			Dataset yang digunakan pada penelitian ini cukup besar. klasifikasi menggunakan dataset baru yang telah difusion dari dataset citra fundus dan OCT

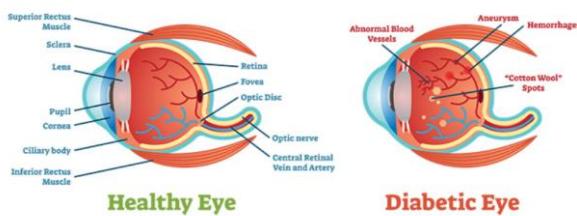
Sedangkan, pada penelitian yang dilakukan oleh penulis adalah Klasifikasi Penyakit Retinopati Diabetik Menggunakan CNN dengan Multimodal Data Fusion Pada Dataset Citra Retina. Penelitian ini bertujuan untuk mengimplementasikan multimodal fusion dataset citra dari fundus dan juga OCT dalam mendeteksi RD dan mengklasifikasikannya.

2.2 Landasan Teori

2.2.1 Retinopati Diabetik (RD)

Retinopati diabetik (RD) adalah salah satu komplikasi serius yang dapat terjadi pada penderita diabetes dan merupakan penyebab utama kebutaan pada orang dewasa. RD berkembang sebagai akibat dari kerusakan pembuluh darah di retina yang disebabkan oleh tingginya kadar gula darah dalam jangka waktu yang lama. Komplikasi ini terkait erat dengan durasi diabetes dan tingkat kontrol glukosa darah yang tidak memadai. Faktor-faktor risiko untuk perkembangan RD melibatkan durasi diabetes, kontrol gula darah yang buruk, tekanan darah tinggi, dan faktor genetik. Penderita diabetes tipe 1 dan tipe 2 berisiko mengalami RD, meskipun risikonya lebih tinggi pada penderita diabetes tipe 2. (Saputri, 2022)

Diabetic Retinopathy



Gambar 2. 1 Retinopati Diabetik.

Sumber:<https://www.kaggle.com/waseemnagahhenes/diabetic-retinopathy-detection-and-classification/notebook>

2.2.2 Pencitraan Retina *Fundus Photography*

Fundus photography, salah satu teknik utama dalam retinal imaging, memungkinkan dokter mata untuk mendokumentasikan kondisi retina dan mendeteksi berbagai penyakit mata, termasuk Retinopati Diabetik (RD). *Fundus photography* menggunakan kamera khusus untuk mengambil gambar retina. Kamera ini memiliki lensa

yang kuat yang dapat fokus pada retina, yang terletak di bagian belakang mata. Kamera fundus biasanya dipasang di mata pasien dengan menggunakan alat khusus yang disebut slit lamp. Slit lamp adalah alat yang digunakan untuk memeriksa mata dengan menggunakan cahaya terfokus. Proses pengambilan gambar *fundus photography* biasanya memakan waktu sekitar 15 menit. Selama proses ini, pasien diminta untuk melihat ke arah cahaya dan tetap diam.

Fundus photography dapat digunakan untuk memprediksi RD. Hal ini karena gambar fundus retina dapat menunjukkan tanda-tanda awal penyakit, seperti: Perdarahan di retina adalah tanda awal RD, Pembuluh darah baru yang abnormal dapat tumbuh di retina pada penderita RD, dan kondisi di mana retina membengkak. (J.F.W. van der Steen, 2012) (A.K. Garg, 2022).



Gambar 2. 2 Hasil Pencitraan Retina Fundus Photography.

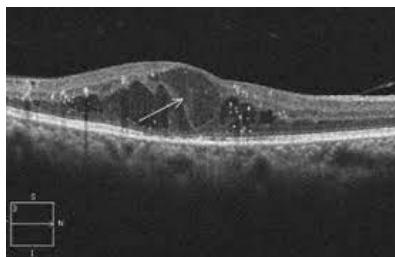
Sumber: <https://www.kaggle.com/competitions/aptos2019-blindness-detection>

2.2.3 *Optical Coherence Tomography (OCT)*

Optical Coherence Tomography (OCT) adalah teknologi pencitraan medis non-invasif yang memungkinkan visualisasi struktur internal dan mikrostruktur jaringan retina. Dengan memanfaatkan sifat interferometri cahaya, *OCT* memberikan gambaran tingkat tinggi tentang lapisan retina, memberikan informasi detail tentang ketebalan lapisan, kerapatan, dan struktur lainnya. *OCT* menggunakan cahaya yang bermodulasi untuk menghasilkan gambar retina. Cahaya yang bermodulasi adalah cahaya yang frekuensinya berubah-ubah. serta menggunakan dua berkas cahaya yang diarahkan ke retina. Satu berkas cahaya adalah berkas cahaya referensi, sedangkan berkas

cahaya lainnya adalah berkas cahaya yang difokuskan ke retina. Cahaya yang dipantulkan dari retina kembali ke detektor. Detektor mengukur perbedaan fase antara berkas cahaya referensi dan berkas cahaya yang dipantulkan.

OCT dapat digunakan untuk memprediksi RD. Hal ini karena *OCT* dapat menghasilkan gambar retina yang menunjukkan tanda-tanda awal penyakit, seperti: Perdarahan di retina adalah tanda awal RD, Pembuluh darah baru yang abnormal dapat tumbuh di retina pada penderita RD, dan kondisi di mana retina membengkak. (A.K. Garg, 2022).

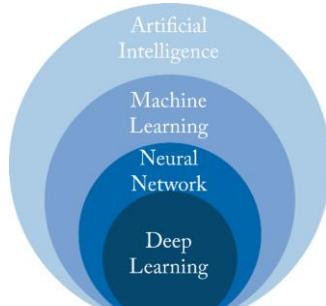


Gambar 2. 3 Hasil Pencitraan Retina optical coherence tomograph (OCT).
Sumber:<https://www.kaggle.com/datasets/paultimothymooney/kerma ny2018>

2.2.4 Deep Learning

Deep learning merupakan suatu kelas dari teknik pembelajaran mesin yang memanfaatkan banyak lapisan dalam pemrosesan informasi untuk ekstraksi dan transformasi, serta untuk analisis pola dan klasifikasi terutama untuk memahami data seperti gambar, suara, dan teks. Deep learning juga subbidang dari machine learning yang menginspirasi oleh struktur dan fungsi otak manusia. Ia menggunakan jaringan syaraf tiruan (*artificial neural networks*) dengan banyak lapisan tersembunyi untuk belajar dari data secara bertahap, sehingga memungkinkan pengenalan pola dan pengambilan keputusan yang kompleks. Sehingga dalam konteks pemahamannya *Deep learning* adalah pembelajaran bertahap dengan banyak lapisan (Bengio, 2016) .Pada Deep Learning lapisan representasi tersebut bernama Neural Networks. Neural Networks memiliki struktur yang bertumpuk yang berarti suatu lapisan akan berada diatas lapisan lain. Neural Networks mengambil konsep dari bidang neurobiology dan terinspirasi dari kemampuan dalam memahami sesuatu seperti yang

dilakukan oleh otak manusia. Meskipun konsep utama Deep Learning dikembangkan atas inspirasi tersebut, tetapi model Deep Learning bukanlah model dari otak manusia.



Gambar 2. 4 Ruang Lingkup Deep Learning

Sumber: <https://images.app.goo.gl/jfJSUH6xggVzpBxLA>

Secara umum, tahapan dalam klasifikasi citra menggunakan deep learning melibatkan langkah-langkah berikut:

- 1) Pemilihan dan Persiapan Data

Pemilihan dan persiapan data adalah langkah awal yang krusial dalam proses klasifikasi citra menggunakan deep learning. Tahapan ini melibatkan pemilihan dataset yang diinginkan dan mempersiapkannya agar siap digunakan. Tahapan ini juga melibatkan eksport data, prepossessing data, dan pembagian data.

- 2) Pembuatan Model Deep Learning

Tahapn ini melibatkan pemilihan arsitektur model yang tepat dan konfigurasi lapisan-lapisan model untuk membangun sebuah jaringan saraf yang dapat memahami dan mengklasifikasi citra dengan akurat. Arsitektur model adalah kerangka kerja atau struktur dasar dari jaringan saraf yang akan digunakan untuk memproses dan mengklasifikasi citra. Arsitektur yang paling umum digunakan untuk klasifikasi citra adalah Convolutional Neural Network (CNN) karena kemampuannya dalam menangkap pola spasial dalam citra.

- 3) Pelatihan Model

Pada tahapan ini merupakan langkah di mana model deep learning belajar untuk mengenali pola dan fitur dari data pelatihan yang Anda berikan. Pola dan fitur tersebut dihitung Setiap iterasi pelatihan (epoch), data pelatihan diumpulkan ke model, kemudian citra diteruskan melalui lapisan-lapisan model, di mana setiap lapisan melakukan operasi matematis tertentu untuk menghasilkan output. Setelah citra melewati model, hasil prediksi model dibandingkan dengan label sebenarnya.

4) Evaluasi Model

Pada tahap ini, model dievaluasi menggunakan data yang tidak pernah dilihat selama proses pelatihan, yang disebut sebagai set data pengujian. Evaluasi dilakukan dengan menggunakan berbagai metrik evaluasi seperti akurasi, presisi, recall, F1-score, dan matriks untuk memahami seberapa baik model dapat memprediksi kelas atau label gambar dengan benar. Hasil evaluasi digunakan untuk menganalisis kekuatan dan kelemahan model. Jika kinerja model tidak memuaskan, ini adalah waktu yang tepat untuk melakukan penyesuaian seperti tuning hyperparameter, mengubah arsitektur model, atau menambahkan lebih banyak data pelatihan atau validasi. Evaluasi model memberikan gambaran tentang seberapa baik model dapat digunakan dalam aplikasi praktis, dan dapat membantu mengidentifikasi area untuk peningkatan lebih lanjut

5) Tuning Model dan Evaluasi Lanjutan

Analisis hasil evaluasi ini meliputi pemahaman mendalam tentang pola kesalahan yang dilakukan oleh model, identifikasi kelas-kelas yang sulit diprediksi, dan pemahaman tentang faktor-faktor yang mempengaruhi kinerja model. Berdasarkan analisis ini, penyesuaian model dapat dilakukan. Ini bisa berupa pengaturan kembali hyperparameter, modifikasi arsitektur model, atau bahkan penggunaan teknik-teknik lanjutan seperti transfer learning atau fine-tuning. Selain itu, evaluasi lanjutan juga dapat melibatkan penggunaan teknik validasi silang (cross-validation) untuk mendapatkan

perkiraan kinerja yang lebih andal dari model. Dengan validasi lanjutan, kita dapat memastikan bahwa model tidak hanya bekerja dengan baik pada satu set data tertentu, tetapi juga dapat menggeneralisasi dengan baik ke data baru.

6) Implementasi dan Penerapan

Implementasi model memerlukan beberapa langkah. Pertama, model harus dipersiapkan agar sesuai dengan lingkungan di mana ia akan digunakan. Ini mungkin melibatkan konversi model ke format yang sesuai atau menyesuaikan model dengan infrastruktur yang ada. Setelah itu, model diterapkan pada data baru dalam lingkungan produksi. Proses ini mungkin memerlukan pengkodean antarmuka atau integrasi dengan sistem yang ada. Setelah model diimplementasikan, langkah selanjutnya adalah pengujian dan validasi. Ini melibatkan menguji model dengan data baru dan memastikan bahwa model memberikan hasil yang diharapkan dalam situasi dunia nyata. Jika diperlukan, model dapat disesuaikan atau diperbarui berdasarkan hasil pengujian ini.

2.2.5 *Convolutional Neural Network (CNN)*

Convolutional Neural Networks (CNN) adalah jenis model pembelajaran mendalam yang telah mendapatkan popularitas signifikan dalam bidang pembelajaran mesin dan visi komputer. CNN dirancang untuk secara otomatis dan adaptif mempelajari hierarki spasial fitur dari data masukan. Mereka sangat cocok untuk memproses data berstruktur grid, seperti gambar dan data deret waktu, karena kemampuannya dalam menangkap dependensi spasial dengan efektif (Zewen Li, 2021).

CNN terdiri dari lapisan konvolusi, pooling, dan terhubung penuh. Lapisan konvolusi menggunakan filter yang dipelajari untuk mengekstraksi fitur, menghasilkan peta fitur. Lapisan pooling mengecilkan peta fitur untuk mengurangi dimensi spasial. Lapisan terhubung penuh memproses fitur tingkat tinggi untuk membuat prediksi. Inovasi seperti konvolusi deformabel dan konvolusi kelompok dapat diterapkan untuk meningkatkan adaptabilitas dan mengurangi kompleksitas model. Secara umum, CNN cocok untuk

data berstruktur grid seperti gambar dan deret waktu, digunakan luas dalam visi komputer dan pemrosesan bahasa alami, dan terus menjadi fokus penelitian dengan prospek pengembangan yang menjanjikan (Zewen Li, 2021) . Kekuatan CNN terletak pada kemampuannya untuk secara otomatis belajar fitur relevan dari data gambar tanpa memerlukan fitur yang ditentukan sebelumnya. Arsitektur berlapis memungkinkan CNN untuk menangkap hierarki fitur, dari tingkat rendah hingga tingkat tinggi, memberikan representasi yang kaya dari konten gambar. Selain itu, sifat invarian spasial CNN membuatnya resisten terhadap transformasi spasial kecil seperti pergeseran, rotasi, dan skala,

2.2.6 Multimodal Data vs Unimodal Data

Multimodal data mengacu pada data yang berasal dari berbagai jenis mode atau sumber informasi. Mode dalam konteks ini merujuk pada jenis atau bentuk data yang berbeda, seperti teks, gambar, suara, video, sensor, dan lainnya. Pemahaman mendalam tentang multimodal data memerlukan penanganan informasi dari berbagai sumber untuk menghasilkan gambaran yang lebih kaya dan holistic. Kemampuan untuk menyatukan data yang berbeda dari beragam sensor ke dalam satu model adalah karakteristik utama dari model multimodal. Fitur ini memungkinkan pembuatan prediksi yang lebih dinamis. Menggunakan beberapa sensor untuk mengamati fenomena yang sama memberikan pemahaman yang lebih lengkap, yang mendarah ke wawasan yang lebih cerdas (Notomoro, 2023). Multimodal data umumnya lebih unggul daripada unimodal data pada klasifikasi citra seperti:

- 1) Informasi yang Lebih Kaya.

Multimodal data memberikan representasi yang lebih kaya dan lebih komprehensif dari suatu objek atau adegan daripada unimodal data. Hal ini karena multimodal data menggabungkan informasi dari berbagai sumber, seperti gambar, teks, audio, dan data sensor. Informasi yang lebih kaya ini membantu model untuk lebih memahami konten gambar dan meningkatkan akurasi klasifikasi.

- 2) Peningkatan Kualitas Analisis.

Penggabungan data dari berbagai mode dapat meningkatkan ketepatan dan akurasi analisis. Kombinasi

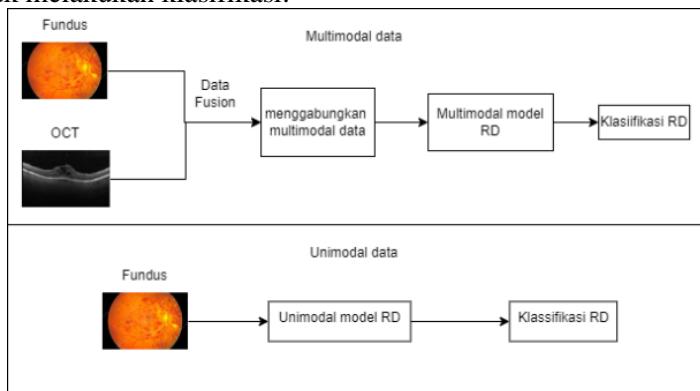
kekuatan masing-masing mode dapat memberikan informasi yang lebih mendalam.

3) Interpretasi yang Lebih Baik.

Multimodal data dapat membantu untuk meningkatkan interpretasi model. Hal ini karena informasi dari berbagai modalitas dapat digunakan untuk menjelaskan mengapa model membuat keputusan tertentu. Contohnya, jika model mengklasifikasikan gambar sebagai "kucing", informasi teks dapat menjelaskan mengapa model membuat keputusan tersebut, seperti dengan menyebutkan ciri-ciri fisik kucing yang terlihat pada gambar.

Sedangkan Unimodal Data mengacu dengan hanya menggunakan satu jenis data, atau modalitas. Beberapa contoh modalitas adalah teks, gambar, audio, dan video. Algoritma pembelajaran mesin tradisional sebagian besar Unimodal Data dirancang untuk bekerja hanya dengan satu jenis data input. Misalnya, jaringan saraf konvolusi menganalisis data gambar sementara jaringan saraf berulang menganalisis data berurutan seperti teks. (Curtis, 2024).

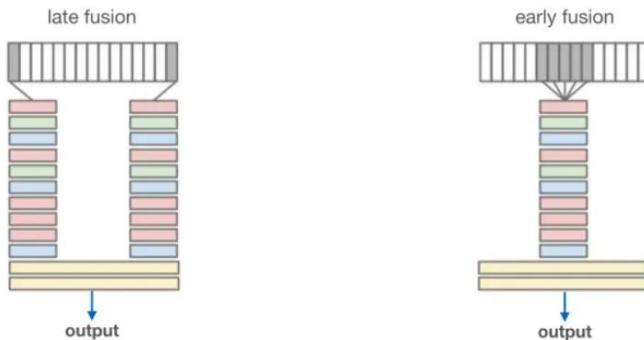
Sehingga dalam klasifikasi gambar, multimodal data dapat menghasilkan akurasi klasifikasi yang lebih tinggi daripada unimodal data, yang hanya menggunakan satu jenis data, seperti gambar saja, untuk melakukan klasifikasi.



Gambar 2. 5 Perbandingan Multimodal Data vs Unimodal Data

2.2.7 Teknik Fusion

Fusion adalah teknik untuk menggabungkan beberapa data menjadi satu dengan tujuan data yang telah digabung dapat diproses dan menjadi representasi penggabungan beberapa modal yang berbeda. Teknik fusion selalu digunakan dalam penelitian yang menggunakan lebih dari satu modal. Teknik fusion yang paling sering ditemukan yaitu Early Fusion.



Gambar 2. 6 Ilustrasi Teknik Early Fusion

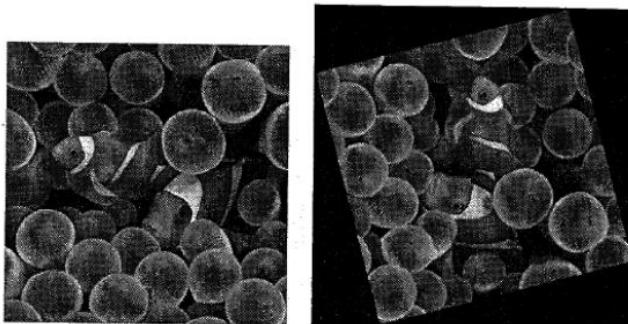
Sumber: <https://chaozhangchn.medium.com/performance-comparison-between-early-fusion-and-late-fusion-5f9d88ffce66>

Dalam early fusion, data dari berbagai modalitas digabungkan pada tahap awal proses. Ini terlihat dari cara data dari berbagai sumber langsung dikombinasikan menjadi satu representasi gabungan sebelum diproses lebih lanjut. Data dari berbagai modalitas atau sumber, yang mungkin berupa teks, gambar, dan suara, dikombinasikan menjadi satu vektor fitur gabungan. Di gambar, kotak-kotak berwarna di bagian atas mewakili fitur-fitur dari berbagai modalitas yang langsung digabungkan menjadi satu set fitur komprehensif yang ditunjukkan dengan kotak yang disatukan. Setelah data digabungkan, representasi gabungan ini kemudian diproses oleh model pembelajaran mesin sebagai satu kesatuan. Ini berarti bahwa model menerima dan mengolah satu set fitur besar yang mengandung informasi dari semua modalitas yang digabungkan. Hasil dari proses ini adalah output dari model yang didasarkan pada analisis dari representasi gabungan. Output ini bisa berupa prediksi atau klasifikasi berdasarkan data multimodal yang telah digabungkan sejak awal. Secara visual, perbedaan utama antara late fusion dan early fusion di

gambar ini adalah titik di mana data dari berbagai modalitas digabungkan. Dalam late fusion, data dari setiap modalitas diproses secara terpisah terlebih dahulu dan hasilnya baru digabungkan di tahap akhir. Sedangkan dalam early fusion, data digabungkan segera setelah diambil, sebelum diproses lebih lanjut oleh model. Dengan early fusion, model dapat langsung menangkap keterkaitan dan interaksi antara data dari berbagai modalitas, karena semua informasi digabungkan sejak awal. Namun, ini juga berarti bahwa representasi gabungan bisa menjadi sangat kompleks dan berukuran besar, yang bisa menambah tantangan dalam hal komputasi dan penyimpanan.

2.2.8 Augmentasi

Augmentasi citra merupakan teknik yang digunakan untuk memperluas dan memperkaya dataset citra dengan menciptakan variasi baru dari citra asli melalui berbagai transformasi. Teknik ini bertujuan untuk meningkatkan kemampuan generalisasi model pembelajaran mesin, terutama dalam konteks deep learning, dengan menambahkan variasi yang mungkin terjadi dalam data nyata tanpa perlu mengumpulkan data tambahan (Laraswati, 2023). Salah satu Teknik augmentasi adalah Rotasi yang dimana teknik augmentasi citra yang melibatkan pemutaran gambar dengan sudut tertentu di sekitar titik pusatnya. Proses ini membantu menciptakan variasi orientasi dari gambar asli, yang memungkinkan model pembelajaran mesin untuk menjadi lebih tahan terhadap perubahan sudut dan orientasi saat menghadapi data baru.

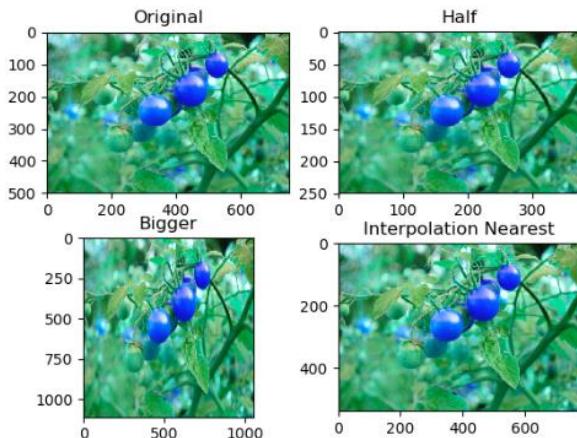


Gambar 2. 7 Augmentasi

Sumber: <https://koleksibukukuliah.blogspot.com/2017/07/operasi-geometri-pada-pengolahan-citra.html>

2.2.9 Resize

Resize adalah proses mengubah ukuran gambar dengan memperbesar atau memperkecil dimensi (lebar dan tinggi) gambar tersebut. Proses ini sering digunakan dalam berbagai aplikasi pengolahan citra dan pembelajaran mesin untuk menormalkan ukuran gambar, sehingga setiap gambar dalam dataset memiliki dimensi yang konsisten. Cara kerja resize pada citra melibatkan beberapa langkah. Pertama, menentukan dimensi baru yang diinginkan untuk gambar, baik sebagai ukuran absolut (misalnya, 224x224 piksel) atau sebagai ukuran relatif berdasarkan faktor skala (misalnya, setengah dari ukuran asli). Kedua, melakukan interpolasi piksel untuk menghitung nilai-nilai piksel pada lokasi baru. Ada beberapa metode interpolasi yang umum digunakan, seperti nearest-neighbor interpolation, bilinear interpolation, dan bicubic interpolation. Metode nearest-neighbor memilih nilai piksel terdekat tanpa perhitungan tambahan, bilinear interpolation mengambil rata-rata tertimbang dari piksel terdekat di empat titik sekitar, dan bicubic interpolation menggunakan rata-rata tertimbang dari 16 piksel terdekat, menghasilkan hasil yang lebih halus.



Gambar 2. 8 Contoh Resize

Sumber: <https://www.myxxgirl.com/sexfoto/merubah-ukuran-resize-citra-menggunakan-opencv-python-di-google-colab.htm>

2.2.10 Grayscale

Grayscale adalah proses konversi gambar berwarna menjadi gambar hitam-putih dengan berbagai tingkatan abu-abu. Dalam gambar berwarna, setiap piksel diwakili oleh kombinasi tiga kanal warna dasar: merah (red), hijau (green), dan biru (blue), yang biasa disebut sebagai model warna RGB (AMARA, 2023). Setiap kanal ini memiliki nilai intensitas yang berbeda, yang secara kolektif menciptakan warna akhir untuk setiap piksel. Pada gambar grayscale, setiap piksel hanya memiliki satu nilai intensitas yang mewakili tingkat kecerahan dari hitam ke putih. Nilai ini biasanya berkisar antara 0 hingga 255, di mana 0 merepresentasikan hitam penuh dan 255 merepresentasikan putih penuh. Proses konversi gambar berwarna ke grayscale biasanya melibatkan pengambilan nilai intensitas untuk setiap piksel dalam gambar berwarna. Nilai intensitas grayscale dihitung dengan menggunakan rata-rata tertimbang dari kanal merah, hijau, dan biru.



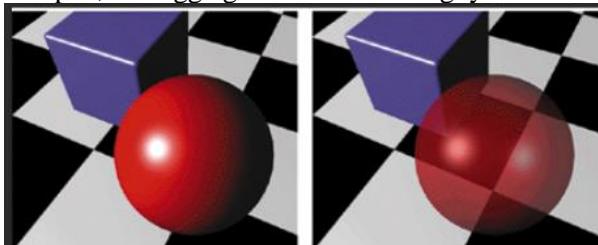
Gambar 2. 9 Contoh Grayslce

Sumber: <https://hendika-blogs.blogspot.com/2013/11/pengolahan-citra-digital-grayscale-pada.html>

2.2.11 Alpha Blending

Alpha blending adalah teknik dalam grafika komputer yang digunakan untuk mengatur tingkat transparansi atau opasitas piksel pada gambar. Setiap piksel dalam gambar memiliki komponen alpha yang menentukan seberapa transparan atau opaknya piksel tersebut. Nilai alpha biasanya berkisar antara 0 dan 1, di mana 0 berarti piksel sepenuhnya transparan dan 1 berarti piksel sepenuhnya opak. Dengan

menggunakan alpha blending, transparansi dapat diatur dengan cara menggabungkan nilai warna piksel dengan nilai alpha untuk menghasilkan efek transparansi. Misalnya, ketika nilai alpha rendah, piksel akan lebih transparan, memungkinkan gambar di belakangnya terlihat melalui piksel tersebut. Sebaliknya, nilai alpha tinggi membuat piksel lebih opak, sehingga gambar di belakangnya akan tertutup.



Gambar 2. 10 Contoh *Alpha Blending*

Sumber: <https://www.pc当地.com/encyclopedia/term/alpha-blending>

2.2.12 *Concatenate*

Concatenate adalah proses menggabungkan dua atau lebih elemen data menjadi satu kesatuan. Dalam konteks pemrograman, concatenate biasanya merujuk pada penggabungan string, array, atau struktur data lain. Misalnya, dalam bahasa pemrograman Python, Anda dapat menggabungkan dua string menjadi satu dengan menggunakan operator +. Dalam pengolahan citra, concatenate berarti menggabungkan gambar secara vertikal atau horizontal. Misalnya, Anda dapat menyusun dua gambar berdampingan atau bertumpuk untuk membuat satu gambar besar. Dalam matematika dan aljabar linier, concatenate merujuk pada penggabungan dua atau lebih vektor atau matriks menjadi satu matriks atau vektor yang lebih besar. Misalnya, menggabungkan dua vektor kolom menjadi satu vektor kolom yang lebih panjang.

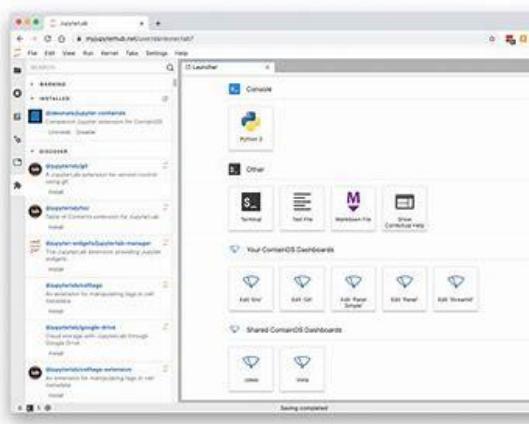
2.2.13 *Python*

Python adalah bahasa pemrograman tingkat tinggi yang populer, digunakan secara luas dalam berbagai bidang, termasuk pengembangan perangkat lunak, analisis data, kecerdasan buatan, pengembangan web, dan sebagainya. Python juga memiliki library yang lengkap sehingga memungkinkan programmer untuk membuat

aplikasi yang mutakhir dengan menggunakan source code yang tampak sederhana (Perkovic, 2012)

2.2.14 Jupyter

Jupyter adalah lingkungan pengembangan yang populer di kalangan ilmuwan data, peneliti, dan pengembang yang memungkinkan pengembangan interaktif dan pembuatan dokumentasi yang dinamis. Jupyter Notebook pertama kali dikembangkan oleh Fernando Perez dan IPython Project pada tahun 2011. Jupyter Notebook awalnya disebut IPython Notebook, tetapi kemudian diganti namanya menjadi Jupyter Notebook pada tahun 2015. Jupyter Notebook terdiri dari dua komponen utama, yaitu kernel dan notebook server. Kernel adalah aplikasi yang menjalankan kode Python. Notebook server adalah aplikasi yang menampilkan notebook. Kernel adalah aplikasi yang menjalankan kode Python. Kernel dapat berjalan di berbagai platform, termasuk Linux, macOS, dan Windows. Sedangkan Notebook server adalah aplikasi yang menampilkan notebook. Notebook server dapat berjalan di berbagai platform, termasuk Linux, macOS, dan Windows.



Gambar 2. 11 Tampilan Jupyter

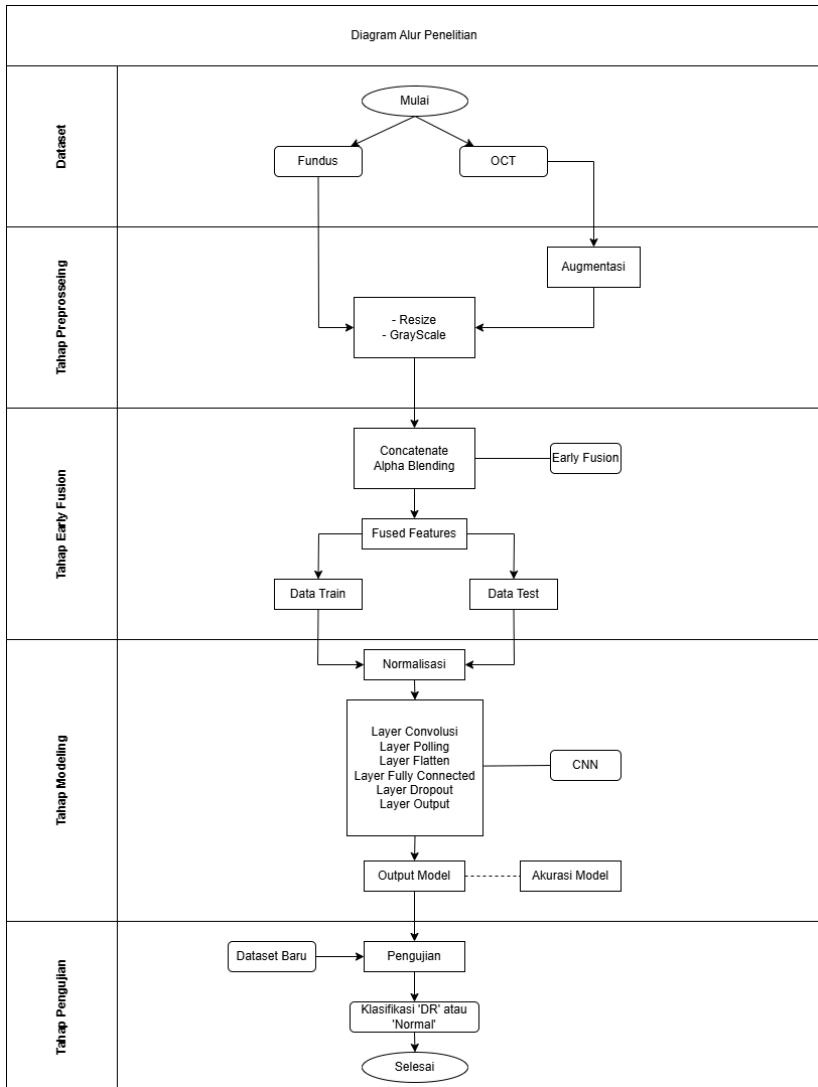
BAB III

PERANCANGAN

Gambar fundus dan *OCT* retina disediakan sebagai data inputan dan memproses klasifikasi sehingga mendapatkan hasil deteksi retina terkena Retinopati Diabetik dan non Diabetic Retinopathy. Peneliti melakukan praproses gambar citra retina dan menerapkan klasifikasi Teknik. Peneliti membagi memagi proses menjadi 5 langkah yaitu: Dataset, Prepossessing, *Early fusion*, Modelling, Evaluasi Model, dan Pengujian Klasifikasi Retinopati Diabetik. Preprocessing termasuk ekstraksi kontras warna citra fundus dan *OCT*. dalam *Early Fusion* peneliti akan menggabungkan seluruh data citra dan augmentasi sehingga hasil dari layer tersebut akan dilakukan fusion dari citra fundus dan *OCT* hingga menghasilkan dataset baru, pada tahapan Modelling, peneliti akan menggunakan arsitektur CNN untuk menklasifikasi dataset baru hingga menghasilkan klasifikasi Normal dan RD. Dan pada tahap Evaluasi, mengevaluasi kinerja dari arsitektur menggunakan data testing sehingga menghasilkan nilai *accuracy*, *Sensitivity*, *Specificity*, dan *AUC*.

3.1 Tahapan Alur Klasifikasi Retinopati Diabetik

Tahapan yang dilakukan pada penelitian ini digambarkan pada melalui diagram alir pada Gambar berikut ini:



Gambar 3. 1 Diagram Alur Penelitian

3.2 Langkah-langkah Klasifikasi Retinopati Diabetik

Pendekatan deteksi Diabetik Retinopai yang diusulkan oleh penelitian ini, dijabarkan dalam mengikuti Langkah-langkah berikut ini:

3.2.1 Dataset

Pada tahapan dataset, peneliti akan melakukan pengumpulan dataset yang akan digunakan pada penelitian ini. Dataset yang digunakan berupa citra retina fundus yang diperoleh dari Kaggle (<https://www.kaggle.com/datasets/pkdarabi/diagnosis-of-diabetic-retinopathy>) dan juga citra retina *OCT* yang diperoleh dari Google Drive(https://drive.google.com/drive/folders/14OVRuOiFl2jICh54m17niU1RhtWX9SdU?usp=drive_link).

Table 3. 1 Dataset Fundus

Dataset Fundus	Total
DR	1163
Normal	1144

Table 3. 2 Dataset OCT

Dataset OCT	Total
DR	206
Normal	107

3.2.2 Preposessing

Pada tahap prepossessing, akan dilakukan deteksi dataset, augmentasi, *resize*, dan *grayscale*. Tahapan ini akan akan diuraikan pada penjelasan berikut ini:

1) Deteksi Dataset.

Deteksi dataset merupakan suatu pendekatan pada bidang pengolahan citra yang bertujuan untuk mengenali dan membaca keseluruhan dataset. Dengan menerapkan deteksi dataset, proses pembacaan keseluruhan dataset dapat terbaca dan memudahkan pada tahapan fitur selanjutnya.

2) Augmentasi.

Augmentasi dataset diperlukan utnuk meningkatkan jumlah dan keragaman data pelatihan dengan membuat variasi kecil pada data pelatihan yang ada. Tujuan

utamanya adalah untuk meningkatkan kinerja dan generalisasi model, mengurangi overfitting, dan membuat model lebih tahan terhadap variasi dalam data uji. Disini peneliti akan menggunakan teknik rotasi sebagai metode augmentasi pada dataset citra OCT, dilakukan augmentasi dengan pemutaran citra sebanyak 9 kali. menggunakan rumus berikut ini:

Rotasi.

$$Jumlah\ Rotasi = \frac{Jumlah\ Dataset\ Akhir - Jumlah\ Dataset\ Awal}{Jumlah\ Dataset\ Awal} \quad (1)$$

Sehingga dataset OCT, RD dan Normal yang digunakan pada penelitian ini masing-masing berjumlah 1.000 dataset.

3) *Resize.*

Resize merupakan merupakan proses pengubahan dimensi atau ukuran citra dari piksel awalnya menjadi ukuran piksel yang baru. Pada tahapan ini semua dataset akan diubah ukuran citra menjadi 224x224 piksel, yang bertujuan agar lebih membantu untuk stabilitas selama pelatihan model yang dapat mengurangi resiko konvergensi yang lambat selama pelatihan model.

4) *Grayscale Citra.*

Teknik grayscale adalah metode untuk mengonversi citra berwarna menjadi citra dengan skala keabuan. Dalam citra berwarna, setiap piksel biasanya direpresentasikan oleh tiga komponen warna: merah, hijau, dan biru (dalam mode warna RGB). Tahapan ini. Peneliti akan mengubah citra warna dari dataset fundus menjadi grayscale dengan rentang nilai intensitas dari 0 hingga 255, dimana 0 mewakili warna hitam dan 255 mewakili warna putih. Rentang ini mencerminkan intensitas piksel yang berkisar dari yang paling gelap hingga yang paling terang dalam citra, agar dataset fundus dapat dapat menyamai grayscale bawaan dari dataset OCT.

3.2.3 Early Fusion

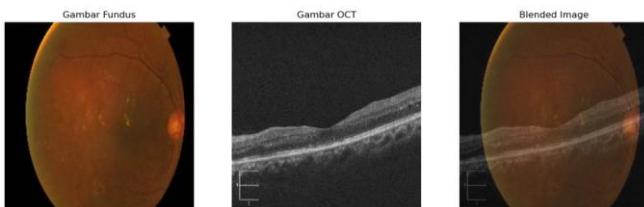
Pada tahapan ini, dilakukan pendekatan dengan metode Early Fusion, dengan melakukan penggabungan antara citra fundus dan OCT. Tahapan ini juga akan melibatkan metode alpha blending, dan penggabungan gambar dengan concatenate. Tahapan metode tersebut akan dijabarkan pada uraian berikut ini:

1) Concatenate.

Concatinate adalah teknik penggabungan dua atau lebih gambar menjadi satu gambar, baik secara horizontal maupun vertical tanpa pengolahan tambahan antara objectnya. Sehingga penggabungan dari citra fundus dan OCT dapat mempertahankan citra aslinya.

2) Alpha Blending.

Alpha blending adalah penggunaan nilai alpha, yang merupakan komponen dari setiap piksel dalam citra RGBA (Red, Green, Blue, Alpha). Nilai alpha menentukan seberapa transparan piksel tersebut: nilai alpha tinggi menunjukkan piksel yang lebih tidak transparan (atau lebih "solid"), sementara nilai alpha rendah menunjukkan piksel yang lebih transparan (atau lebih "translusen"). Nilai rentang alpha yang dilakukan pada penelitian ini dari 0-1 sebagai transparasi citranya. Sehingga pada tahapan ini, peneliti ingin menjadikan gambar citra fundus sebagai citra yang solid dengan nilai alpha 0,7 untuk citra fundus, dan gambar citra OCT sebagai citra traslusen dengan nilai alpha 0,3 untuk citra OCT.



Gambar 3. 2 Teknik *Concatenate + Alpha Blending*

3) Pembagian Dataset

Pada tahapan ini dataset yang digunakan masih belum dibagi menjadi data Train dan data Testing, sehingga diperlukan sebuah proses untuk membagi dataset sesuai dengan tingakatan RD dan Normal. rasio perbandingan yang digunakan yaitu 80:20, 70:30 dan 60:40. Yang dimana terbagi untuk data train dan testing yang digunakan untuk melatih dan menguji model.

Dengan metode yang dijelaskan oleh peneliti, memungkinkan untuk mempertahankan informasi dari setiap citra saat proses penggabungan dataset untuk menghasilkan dataset fused features.

3.2.4 Modelling

Pada tahapan ini, peneliti akan melakukan normalisasi data dan menggunakan arsitektur algoritma CNN pada tahapan modelling. Yang diuraikan pada penjelasan berikut ini

1) Normalisasi

Normalisasi adalah proses mengubah nilai-nilai dari suatu dataset sehingga memiliki skala atau rentang tertentu. Tujuan normalisasi adalah untuk mengubah rentang intensitas piksel yang berada ke dalam distribusi normal agar citra menjadi lebih optimal pada saat diproses. Pada tahap ini, citra berwarna RGB (Red, Green, Blue) akan dinormalisasi ke dalam piksel [0, 1].

2) Arsitektur CNN

Setelah data train dan test dinormalisasi peneliti akan menggunakan arsitektur algoritma CNN pada tahapan modelling. Arsitektur CNN yang terdiri dari beberapa layer yaitu, Layer Convoluti, Layer Pooling, Layer Flatten, Layer Fully Connected, Layer Dropout, Layer Output, yang diuraikan pada penjelasan berikut ini:

Pada layer convolutional, akan dilakukan operasi konvolusi untuk mengekstrasi fitur dari data input menggunakan kernel konvolusi. Hasil dari operasi konvolusi menghasilkan peta fitur/feature map, yang dimana setiap nilai dalam peta fitur mewakili respon filter terhadap fitur atau pola tertentu dalam data yang masuk. Sederhananya, layer convolutional membantu model

belajar melihat pola dan fitur-fitur dalam gambar, seperti tepi atau bentuk, yang nantinya akan membantu model memahami gambar secara lebih baik

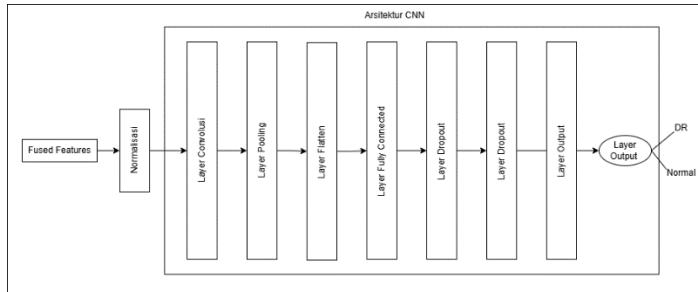
Kemudian pada tahapan layer pooling, berfungsi untuk mereduksi dimensi spasial dari feature map yang dihasilkan oleh layer convolutional. Reduksi dimensi ini membantu mengurangi jumlah parameter dan komputasi dalam jaringan, sambil mempertahankan informasi penting yang diambil dari layer convolutional.

Flatten layer berfungsi untuk mengubah dimensi multi-dimensi dari feature map yang dihasilkan oleh layer sebelumnya menjadi satu dimensi. Transformasi ini penting untuk menghubungkan output dari layer convolutional atau pooling ke layer dense (fully connected). Dengan mengubah data menjadi vektor satu dimensi, flatten layer memungkinkan jaringan untuk mengintegrasikan informasi spasial yang kompleks dari feature map ke dalam proses klasifikasi atau regresi yang lebih sederhana.

Kemudian pada tahapan layer full connected setelah menggabungkan firut dari setiap modalitas peneliti akan beberapa layer connected untuk memperoses vector firut gabungan dari citra dengan aktivasi ReLU.

Layer Dropout adalah teknik yang digunakan dalam jaringan saraf untuk mengurangi overfitting. Overfitting terjadi ketika model yang dibuat terlalu kompleks / sangat baik, tetapi gagal untuk menggeneralisasi dengan baik pada data baru yang tidak terlihat sebelumnya.

Kemudian pada tahapan selanjutnya yaitu layer output, Output layer adalah lapisan terakhir dalam sebuah jaringan saraf tiruan yang berfungsi untuk menghasilkan prediksi atau keputusan akhir dari jaringan berdasarkan input yang telah diproses melalui lapisan-lapisan sebelumnya. Output layer mengubah representasi internal dari jaringan menjadi format yang bisa digunakan untuk prediksi Berikut rancangan model CNN tersebut:



Gambar 3. 3 Rancangan Arsitektur CNN

Dan model disimpan dalam bentuk HDF5 yang akan digunakan untuk klasifikasi

3.2.5 Akurasi Model

Kemudian pada tahap akurasi model, penelitian ini akan dievaluasi dari model CNN yang telah dijalankan menggunakan perhitungan *confusion matrix* berdasarkan *metrics* yang digunakan yaitu *accuracy*, *Sensitivity*, *Specificity*, dan *AUC* (Area Under the Curve), yang akan dijabarkan pada uraian berikut ini:

1. *Accuracy* adalah salah satu metrik evaluasi yang umum digunakan untuk mengukur kinerja model klasifikasi. Metrik ini memberikan gambaran tentang sejauh mana model dapat memprediksi dengan benar kelas atau label yang tepat. Secara umum, akurasi dihitung sebagai rasio antara jumlah prediksi yang benar (positif dan negatif) dibagi dengan total jumlah sampel.

Accuracy

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

2. *Sensitivity*, adalah metrik yang mengukur seberapa baik model dapat mengidentifikasi sampel positif dari semua sampel yang benar-benar positif dalam dataset. Secara khusus, sensitivitas mengukur kemampuan model untuk mengklasifikasikan dengan benar semua kasus positif yang sebenarnya.

Sensitivity

$$Sensitivity = \frac{TN}{TN+FP} \quad (3)$$

3. *Specificity* memberikan informasi tentang seberapa baik model dapat menghindari kesalahan dalam memprediksi kelas negatif. Semakin tinggi nilai Specificity, semakin sedikit sampel kelas negatif yang salah diprediksi sebagai kelas positif oleh model.

Specificity

$$Specificity = \frac{TN}{TN+FP} \quad (4)$$

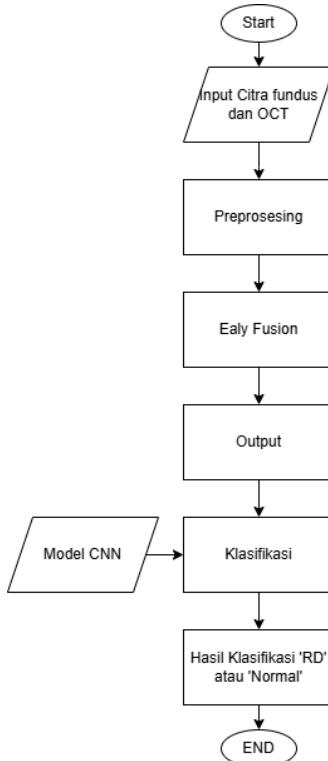
4. AUC adalah model klasifikasi memberikan informasi yang berguna tentang kemampuan model dalam membedakan antara kelas positif dan negatif, dan dapat membantu dalam pembandingan dan seleksi model. Semakin tinggi nilai AUC, semakin baik kinerja modelnya.

AUC

$$AUC = \frac{1}{2} \left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP} \right) \quad (5)$$

3.2.6 Testing

Proses testing yaitu berupa tahapan klasifikasi dengan menguji data citra validasi dan membandingkannya dengan model hasil training data citra latih yang disimpan pada database.



Gambar 3. 4 Flowchart proses Testing Model

Berdasarkan gambar 3.4 diatas, Tahapan pertama adalah menginputkan data citra fundus dan oct. Citra akan melalui tahap preprocessing dan *early fusion* terlebih dahulu sebelum sehingga menghasilkan citra baru fusion, citra baru fusion tersebut akan di proses klasifikasi CNN menggunakan model yang sudah dilatih. Setelah citra selesai diklasifikasi, akan menghasilkan klasifikasi dari RD atau Normal.

BAB IV

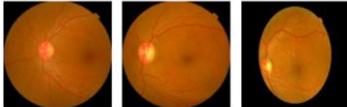
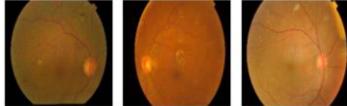
PENGUJIAN DAN ANALISIS

4.1 Dataset

4.1.1 Pengumpulan *Dataset*

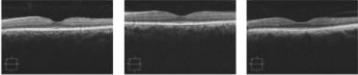
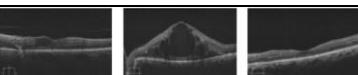
Dataset yang digunakan berupa citra retina fundus yang diperoleh dari Kaggle¹ dan juga citra retina OCT yang diperoleh dari Google Drive².

Tabel 4. 1 Dataset Fundus dan OCT

No	Dataset	Keterangan	Jumlah Data
1	  	Fundus Normal	1144
2	  	Fundus RD	1163

¹<https://www.kaggle.com/datasets/pkdarabi/diagnosis-of-diabetic-retinopathy>

²<https://drive.google.com/drive/folders/14OVRuOiFl2jICh54m17niU1RhtWX9SdU?usp=drivelink>

3		OCT Normal	107
4		OCT RD	206
			

4.2 . Prosesing

4.2.1 Augmentasi Dataset

Dilakukan augmentasi yang bertujuan untuk menghasilkan data tambahan dari data yang sudah ada. Disini peneliti mengaugmentasi data dari dataset OCT, dengan melakukan rotasi pada setiap gambar sebanyak 9 kali, dengan sudut rotasi yang ditentukan dari 0 hingga 360 derajat searah jarum jam.

```

source_dir = 'Training_Testing/PA_almun/data ori/oct/OCT DR'
target_dir = 'Training_Testing/PA_almun/data ori/oct_augmentasi/DR'

if not os.path.exists(target_dir):
    os.makedirs(target_dir)

# Menghitung sudut rotasi
rotation_angles = np.linspace(0, 360, 10, endpoint=False) # Membuat 9 sudut, mengabaikan 360 karena sama dengan 0

augmented_images_count = 0 # Penghitung untuk gambar yang telah diaugmentasi

for img_name in os.listdir(source_dir):
    img_path = os.path.join(source_dir, img_name)
    for i, angle in enumerate(rotation_angles):
        save_path = os.path.join(target_dir, f'oct_{int(angle)}_{(img_name)}')
        if rotate_image(img_path, angle, save_path):
            augmented_images_count += 1
        if augmented_images_count >= 1000:
            print("Target 1000 ocr_DR gambar telah berhasil diaugmentasi.")
            break # Keluar dari loop jika target tercapai
        if augmented_images_count >= 1000:
            break # Keluar dari loop luar jika target tercapai

if augmented_images_count < 1000:
    print(f"Proses selesai. Total gambar yang diaugmentasi: {augmented_images_count}.")
else:
    print("Proses augmentasi gambar selesai dengan sukses.")

```

Gambar 4. 1 Code Program Augmentasi Dataset.

Kode program ditulis ini menggunakan pustaka cv2 dari OpenCV dan numpy untuk menangani proses rotasi dan manajemen file.

Program dimulai dengan mendefinisikan fungsi `rotate_image`, yang bertugas untuk memutar gambar pada sudut tertentu. Fungsi ini menerima tiga parameter: `image_path` (jalur gambar yang akan diputar), `angle` (sudut rotasi dalam derajat), dan `save_path` (jalur untuk menyimpan gambar hasil rotasi).

Di dalam fungsi, gambar dibaca dari `image_path` menggunakan `cv2.imread`. Jika gambar tidak berhasil dibaca (misalnya, jika file tidak ada atau rusak), fungsi akan menghasilkan pengecualian dengan pesan error. Selanjutnya, fungsi menghitung dimensi gambar dan menentukan titik pusat rotasi. Dengan menggunakan `cv2.getRotationMatrix2D`, matriks rotasi dibuat berdasarkan sudut yang diberikan. Gambar kemudian diputar dengan `cv2.warpAffine` menggunakan matriks rotasi ini. Hasil rotasi disimpan ke `save_path` menggunakan `cv2.imwrite`. Jika proses berhasil, fungsi mengembalikan `True`; jika tidak, fungsi mencetak pesan error dan mengembalikan `False`.

Program kemudian mengatur direktori sumber (`source_dir`) dan direktori tujuan (`target_dir`). Jika direktori tujuan belum ada, program akan membuatnya.

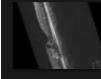
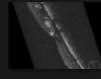
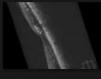
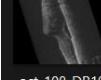
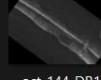
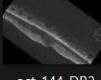
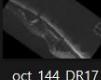
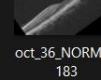
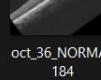
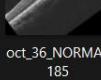
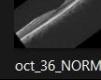
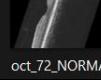
Selanjutnya, sudut rotasi dihitung menggunakan `np.linspace`, menghasilkan 9 sudut mulai dari 0 hingga 360 derajat (dengan 360

derajat diabaikan karena sama dengan 0 derajat). Program kemudian memulai loop untuk memproses setiap gambar dalam direktori sumber. Untuk setiap gambar, program akan memutar gambar pada setiap sudut yang telah ditentukan dan menyimpan hasilnya ke direktori tujuan dengan nama file yang mencakup sudut rotasi.

Program melacak jumlah gambar yang telah diproses menggunakan augmented_images_count. Jika jumlah gambar yang telah diproses mencapai 1000, program akan mencetak pesan bahwa target 1000 gambar telah tercapai dan menghentikan proses rotasi lebih lanjut.

Dengan jumlah dataset yang telah mencukupi dari citra Fundus dan OCT, dilakukan Preposesing, berikut hasil augmentasi dataset OCT:

Tabel 4. 2 Hasil Augmentasi OCT

Keterangan	Dataset OCT Baru								
OCT_DR	        								
OCT_Normal	        								

4.2.2 *Resize* Citra.

Data citra yang didapat memiliki ukuran pixel yang berbeda-beda. Oleh karena itu dilakukan proses resize ukuran pixel, yang dimana pada tahap ini bertujuan untuk menyamakan ukuran data citra pada saat Penggabungan citra dan juga untuk mempermudah pada tahapan *training*.

```
# Ukuran target  
target_size = (224, 224)
```

Gambar 4. 2 Kode Program Resize

Pada tahap resize ini dilakukan perubahan ukuran pixel data citra fundus dan OCT menjadi 224x224

4.2.3 *Grayscale*.

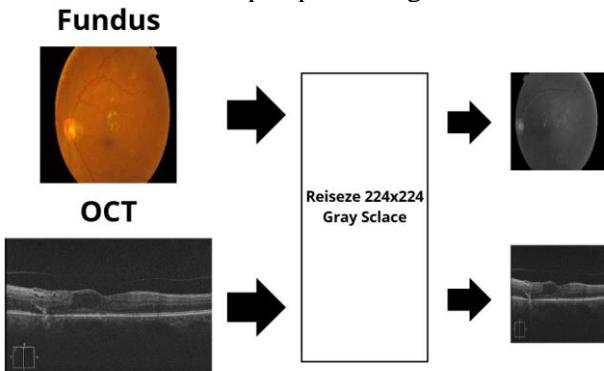
Data citra yang didapatkan memiliki berbagai spektrum warna (RGB) yang berbeda-beda. Oleh karena itu, dilakukan proses konversi ke grayscale. Tahap ini bertujuan untuk menyederhanakan data citra dengan mengurangi kompleksitas warna menjadi hanya satu kanal intensitas cahaya, sehingga mempermudah proses pengolahan lebih lanjut penggabungan citra dan proses *training*.

```
# Fungsi untuk resize dan convert ke grayscale  
def process_images(input_dir, output_dir):  
    if not os.path.exists(output_dir):  
        os.makedirs(output_dir)  
  
    for filename in os.listdir(input_dir):  
        if filename.endswith('.jpg') or filename.endswith('.png') or filename.endswith('.jpeg'):br/>            img_path = os.path.join(input_dir, filename)  
            img = Image.open(img_path)  
            img = img.resize(target_size).convert('L')  
            output_path = os.path.join(output_dir, filename)  
            img.save(output_path)
```

Gambar 4. 3 Kode program Grayscale

Pada tahap konversi grayscale ini, citra berwarna (fundus dan OCT) yang telah diresize kemudian dikonversi ke format grayscale dengan metode .convert('L'). Dalam mode grayscale ('L'), setiap piksel dalam gambar mewakili nilai intensitas cahaya dari 0 (hitam) hingga 255 (putih). Proses konversi ini menghilangkan informasi warna dan menghasilkan gambar dengan satu saluran intensitas. Formula konversi standar yang digunakan mempertimbangkan sensitivitas

retina terhadap warna hijau, sehingga intensitas hijau diberi bobot lebih besar dibandingkan warna merah dan biru.. Konversi ini dilakukan dengan memanfaatkan fungsi cv2.cvtColor dari OpenCV. Berikut adalah hasil dari tahapan prosessing:

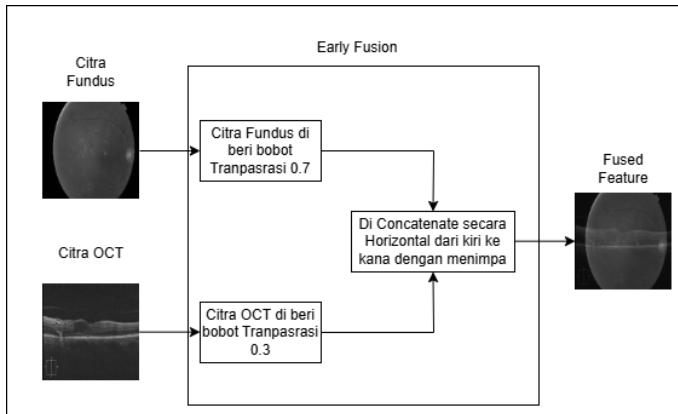


Gambar 4. 4 Preposesing Citra Fundus dan OCT.

4.3 Early Fusion

4.3.1 Concatenate + Alpha Blending.

Pada tahapan ini, gambar fundus dan OCT diberikan bobot dengan nilai alpha untuk transparasi pada kedua dataset. Perbandingan gambar fundus adalah 0.7 dan gambar OCT adalah 0.3. Kemudian dilakukan penggabungan gambar dengan np.concatenate yang di custom dengan pendekatan dari *Alpha Blending*, dengan cara enambahkan saluran alpha ke kedua gambar menggunakan add_alpha_channel, Kemudian menentukan lebar untuk tumpang tindih (lebar dari perbangiannya 1:1) kemudian memisahkan bagian kiri dan kanan dari gambar pertama dan melapisi gambar kedua di bagian kiri dengan bobot (alpha2). Terakhir, menggabungkan bagian yang dilapisi dengan sisa bagian kanan dari gambar pertama dan mengembalikan gambar yang digabungkan.



Gambar 4. 5 Proses Early Fusion

```

import os
import cv2
import numpy as np
import matplotlib.pyplot as plt

def add_alpha_channel(image):
    b_channel, g_channel, r_channel = cv2.split(image)
    alpha_channel = np.ones(b_channel.shape, dtype=b_channel.dtype) * 255
    return cv2.merge((b_channel, g_channel, r_channel, alpha_channel))

# Fungsi untuk menggabungkan dua gambar dengan np.concatenate
def concatenate_images(image1, image2, alpha1=0.7, alpha2=0.3):
    image1_with_alpha = add_alpha_channel(image1)
    image2_with_alpha = add_alpha_channel(image2)

    width_to_overlay = min(image1_with_alpha.shape[1], image2_with_alpha.shape[1])

    image1_left = image1_with_alpha[:, :width_to_overlay]
    image1_right = image1_with_alpha[:, width_to_overlay:]
    image2_overlay = image2_with_alpha[:, :width_to_overlay]

    blended_overlay = cv2.addWeighted(image1_left, alpha1, image2_overlay, alpha2, 0)
    concatenated_image = np.concatenate((blended_overlay, image1_right), axis=1)

    return concatenated_image

# Path ke folder gambar
fundus_dr_folder = 'data garslece/fundus/DR'
oct_dr_folder = 'data garslece/oct/DR'
output_folder = 'penelitian 3/DR'

# Membaca semua gambar dari folder DR fundus
fundus_dr_images = []
fundus_dr_filenames = []
for filename in os.listdir(fundus_dr_folder):
    img_path = os.path.join(fundus_dr_folder, filename)
    if os.path.isfile(img_path):
        img = cv2.imread(img_path)
        fundus_dr_images.append(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        fundus_dr_filenames.append(filename)

```

```

# Membaca semua gambar dari folder DR oct
oct_dr_images = []
oct_dr_filenames = []
for filename in os.listdir(oct_dr_folder):
    img_path = os.path.join(oct_dr_folder, filename)
    if os.path.isfile(img_path):
        img = cv2.imread(img_path)
        oct_dr_images.append(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        oct_dr_filenames.append(filename)

# Mengatur jumlah gambar yang akan diolah
num_images_to_merge = 1000 # Ubah sesuai dengan jumlah gambar yang ingin Anda proses

# Menggabungkan beberapa gambar dari folder DR fundus dan DR oct dan menyimpannya ke folder output
concat_images = []
num_images_to_merge = min(num_images_to_merge, min(len(fundus_dr_images), len(oct_dr_images)))
for i in range(num_images_to_merge):
    concat_image = concatenate_images(fundus_dr_images[i], oct_dr_images[i])
    concat_images.append(concat_image)

# Menyimpan gambar hasil penggabungan ke folder output
output_filename = f'fusion_DR_{i+1}.png'
output_path = os.path.join(output_folder, output_filename)
cv2.imwrite(output_path, cv2.cvtColor(concat_image, cv2.COLOR_RGB2BGR))
print(f'Gambar {output_filename} berhasil disimpan di {output_path}')

# Menampilkan hasil penggabungan beberapa contoh gambar
num_images_to_show = min(3, num_images_to_merge)
plt.figure(figsize=(15, 5*num_images_to_show))

for i in range(num_images_to_show):
    plt.subplot(num_images_to_show, 3, i*3 + 1)
    plt.imshow(fundus_dr_images[i])
    plt.title('fundus')
    plt.axis('off')

    plt.subplot(num_images_to_show, 3, i*3 + 2)
    plt.imshow(oct_dr_images[i])
    plt.title('OCT')
    plt.axis('off')

    plt.subplot(num_images_to_show, 3, i*3 + 3)
    plt.imshow(concat_images[i])
    plt.title('Hasil Penggabungan')
    plt.axis('off')

plt.tight_layout()

```

Gambar 4. 6 Kode Program Early Fusion

Dalam penelitian ini, peneliti menggunakan beberapa pustaka Python, yaitu os, cv2, numpy, dan matplotlib, untuk memproses dan menggabungkan gambar dari dua sumber yang berbeda: fundus DR dan OCT.

Proses dimulai dengan mendefinisikan fungsi add_alpha_channel yang bertujuan untuk menambahkan saluran alpha pada gambar. Penambahan saluran alpha ini penting karena memberikan informasi tentang transparansi gambar. Fungsi ini memisahkan saluran warna biru, hijau, dan merah dari gambar, kemudian menambahkan saluran alpha dengan nilai 255, yang menunjukkan opasitas penuh. Setelah itu, saluran-saluran tersebut

digabungkan kembali untuk menghasilkan gambar dengan empat saluran (BGRA).

Selanjutnya, peneliti mengimplementasikan fungsi concatenate_images, yang digunakan untuk menggabungkan dua gambar secara horizontal. Dalam fungsi ini, kedua gambar terlebih dahulu ditambahkan saluran alpha dengan citra fundus diberi bobot alpha 0.7 dan citra OCT diberi bobot alpha 0.3. Fungsi ini kemudian menentukan lebar area yang akan digabungkan berdasarkan ukuran gambar yang lebih sempit. Gambar pertama dibagi menjadi bagian kiri dan kanan, sedangkan bagian kiri dari gambar pertama dicampur dengan bagian kiri dari gambar kedua menggunakan metode cv2.addWeighted, yang memungkinkan pencampuran warna dengan bobot tertentu. Hasil campuran ini kemudian digabungkan dengan bagian kanan gambar pertama untuk membentuk gambar akhir dengan np.concatenate.

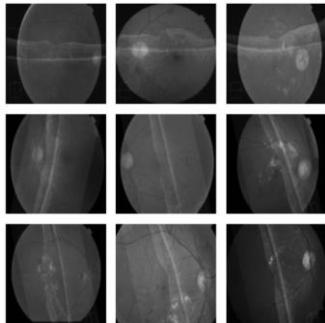
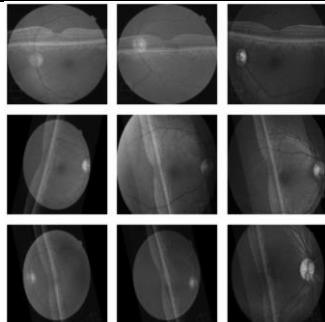
Setelah itu, gambar-gambar dari kedua folder (fundus dan OCT) dibaca. Proses pembacaan dilakukan dengan menggunakan cv2.imread, dan gambar-gambar tersebut dikonversi dari format BGR, yang merupakan format default OpenCV, ke format RGB untuk memastikan kompatibilitas dengan tampilan visual menggunakan matplotlib.

Selanjutnya, mengatur jumlah gambar yang akan diproses, dengan batas maksimal sebanyak 1000 gambar. Gambar-gambar dari kedua sumber tersebut kemudian digabungkan sesuai dengan jumlah yang telah ditentukan. Hasil dari penggabungan setiap pasangan gambar disimpan ke dalam folder output dengan nama file yang terstruktur. Proses ini melibatkan penyimpanan gambar hasil penggabungan menggunakan cv2.imwrite.

Kemudian, menampilkan beberapa contoh gambar fundus, OCT, dan hasil penggabungan mereka. Penggunaan matplotlib memungkinkan peneliti untuk menampilkan gambar-gambar ini dalam format grid yang terorganisir. Setiap gambar ditampilkan dengan judul yang sesuai, sehingga memudahkan identifikasi antara gambar fundus, OCT, dan hasil penggabungan.

Berikut adalah tabel penggabungan early fusion sehingga menjadi dataset barunya:

Tabel 4. 3 Hasil Early Fusion

Dataset	Gambar Ealy Fusion	Dataset Baru
Fundus DR (1000 Gambar) + OCT DR (1000 Gambar)		Fusion_DR
Fundus Normal (1000 Gambar) + OCT Normal (1000 Gambar)		Fusion_Normal

4.3.2 Pembagian Dataset.

Setelah dataset gambar disimpan berdasarkan labelnya, dataset akan dibagi menjadi data training dan data testing. Pada proses pembagian data dilakukan dalam penulisan kode program (coding) dengan menggunakan bahasa pemrograman python dan library dari sklearn.

```

import os
import shutil
from sklearn.model_selection import train_test_split

# Path ke folder dataset
folder_DR = 'penelitian 3/DR'
folder_normal = 'penelitian 3/Normal'

# Path ke folder output
output_train_DR = 'penelitian 3/data 80 20/train/DR'
output_train_normal = 'penelitian 3/data 80 20/train/normal'
output_test_DR = 'penelitian 3/data 80 20/test/DR'
output_test_normal = 'penelitian 3/data 80 20/test/normal'

# Membuat folder output jika belum ada
os.makedirs(output_train_DR, exist_ok=True)
os.makedirs(output_train_normal, exist_ok=True)
os.makedirs(output_test_DR, exist_ok=True)
os.makedirs(output_test_normal, exist_ok=True)

# Fungsi untuk membagi data
def split_data(folder, output_train, output_test, test_size=0.2):
    # Mengambil semua file gambar dalam folder
    images = [f for f in os.listdir(folder) if os.path.isfile(os.path.join(folder, f))]

    # Membagi data menjadi train dan test
    train_images, test_images = train_test_split(images, test_size=test_size, random_state=42)

    # Menyalin file gambar ke folder train
    for image in train_images:
        shutil.copy(os.path.join(folder, image), os.path.join(output_train, image))

    # Menyalin file gambar ke folder test
    for image in test_images:
        shutil.copy(os.path.join(folder, image), os.path.join(output_test, image))

    # Membagi data untuk folder DR
    split_data(folder_DR, output_train_DR, output_test_DR)

    # Membagi data untuk folder normal
    split_data(folder_normal, output_train_normal, output_test_normal)

```

Gambar 4. 7 Kode Program Pembagian Dataset.

Program ini menggunakan pustaka Python os, shutil, dan train_test_split dari sklearn.model_selection.

Program dimulai dengan mengimpor pustaka yang diperlukan. Pustaka os digunakan untuk operasi sistem file seperti membuat folder dan memeriksa file, sedangkan shutil digunakan untuk menyalin file dari satu folder ke folder lain. train_test_split dari sklearn.model_selection digunakan untuk membagi dataset menjadi set pelatihan dan pengujian.

Jalur direktori untuk dataset gambar didefinisikan. Direktori sumber gambar untuk kategori "DR" dan "Normal" ditetapkan, bersama dengan jalur direktori tujuan untuk gambar train dan test.

Program kemudian memastikan bahwa direktori tujuan ada dengan menggunakan os.makedirs, yang membuat folder jika folder tersebut belum ada. Jika folder sudah ada, parameter exist_ok=True memastikan bahwa tidak ada kesalahan yang dihasilkan.

Fungsi split_data digunakan untuk membagi dataset gambar. Fungsi ini menerima direktori sumber gambar, direktori tujuan untuk gambar train, direktori tujuan untuk gambar test, dan proporsi data untuk set pengujian sebagai parameter. Di dalam fungsi ini, semua file gambar diambil dari folder sumber dan dibagi menjadi dua set menggunakan train_test_split. Gambar-gambar dari set train dan test kemudian disalin ke direktori yang sesuai menggunakan shutil.copy.

Terakhir, fungsi split_data dipanggil untuk masing-masing direktori gambar "DR" dan "Normal", sehingga gambar-gambar dalam direktori tersebut dibagi dan disalin ke folder train dan test yang telah ditentukan.

Berikut hasil distribusi pembagian dataset:

Tabel 4. 4 Distribusi data train dan test

Skenario Pembagian	Training	Testing
80:20	1600	400
70:30	1400	600
60:40	1200	800
50:50	1000	1000

4.4 Implementasi Model.

4.4.1 Normalisasi

Proses normalisasi dilakukan dengan menggunakan ImageDataGenerator dari library keras, data train dan test yang sudah inputkan akan dinormalisasi, sehingga nilai piksel gambar dalam train_batches dan test_batches berada dalam rentang 0 hingga 1.

```
train_steps = np.ceil(num_train_samples / train_batch_size)
test_steps = np.ceil(num_test_samples / test_batch_size)

datagen = ImageDataGenerator(rescale=1./255)

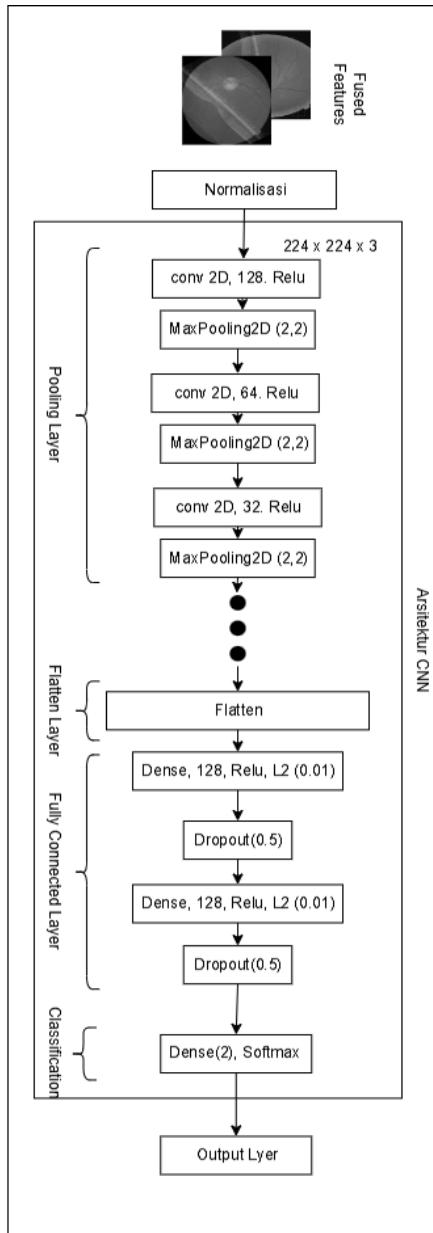
train_batches = datagen.flow_from_directory(train_path,
                                             target_size=(224, 224),
                                             batch_size=train_batch_size,
                                             classes=['DR', 'normal'])

test_batches = datagen.flow_from_directory(test_path,
                                             target_size=(224, 224),
                                             batch_size=test_batch_size,
                                             classes=['DR', 'normal'],
                                             shuffle=False)
```

Gambar 4. 8 Proses Normalisasi

4.4.2 Model CNN

Pada tahap pembuatan model, proses training dan testing dilakukan setelah data dinormalisasi. Proses ini menggunakan platform Jupyter Notebook dengan Bahasa pemrograman Python dan berbagai library, seperti TensorFlow, scikit-learn (sklearn), Matplotlib, OpenCV, dan Keras.



Gambar 4. 9 Arsitektur CNN

```

# Define the CNN model
model_cnn = Sequential([
    Conv2D(128, (3, 3), activation='relu', input_shape=(image_size, image_size, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(2, activation='softmax', kernel_regularizer=l2(0.01))
])

# Compile the model
model_cnn.compile(optimizer=Adam(learning_rate=0.001),
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])

# Fit the model with early stopping
history_cnn = model_cnn.fit(train_batches,
                             validation_data=test_batches,
                             epochs=100)

```

Gambar 4. 10 Kode Program CNN

Model pada Gambar 4.8 dibangun dengan menggunakan arsitektur Convolutional Neural Network (CNN). Model ini memiliki beberapa lapisan utama yaitu Lapisan Convolutional 2D (Conv2D), Lapisan MaxPooling, Lapisan Flatten, Lapisan Dense, dan Lapisan Dropout.

Pada layer Conv2D, terdapat tiga lapisan dengan masing-masing memiliki 128, 64, dan 32 filter serta ukuran kernel 3x3. Setiap layer Conv2D menggunakan fungsi aktivasi ReLU untuk memperkenalkan non-linearitas ke dalam model. Pada lapisan konvolusi pertama, ukuran gambar yang awalnya 224x224x3 berubah menjadi 222x222x128. Kemudian, setelah layer MaxPooling2D dengan ukuran pooling 2x2, ukuran gambar menjadi 111x111x128. Pada lapisan konvolusi kedua dengan 64 filter dan kernel 3x3, ukuran gambar menjadi 109x109x64 setelah proses konvolusi. Setelah itu, lapisan MaxPooling2D mengurangi ukuran gambar menjadi 54x54x64. Lapisan konvolusi ketiga memiliki 32 filter dengan kernel 3x3 yang mengubah ukuran gambar menjadi 52x52x32. Setelah melalui layer MaxPooling2D terakhir, ukuran gambar menjadi 26x26x32. Output dari layer konvolusi dan pooling kemudian diratakan menggunakan layer Flatten sebelum memasuki layer Dense.

Layer Flatten mengubah data dari bentuk dua dimensi menjadi satu dimensi agar dapat diproses oleh layer fully connected (Dense).

Lapisan fully connected pada model ini terdiri dari tiga lapisan. Lapisan pertama memiliki 128 neuron dengan fungsi aktivasi ReLU dan menggunakan regularisasi kernel L2 untuk mencegah overfitting. Setelah lapisan ini, terdapat lapisan Dropout dengan rate 0.5 yang bertujuan untuk mengurangi overfitting dengan mengabaikan (drop) 50% neuron selama pelatihan. Lapisan kedua fully connected memiliki 64 neuron, juga dengan fungsi aktivasi ReLU dan regularisasi kernel L2. Seperti sebelumnya, lapisan ini diikuti oleh lapisan Dropout dengan rate 0.5 untuk mengurangi overfitting.

Lapisan terakhir adalah lapisan output dengan 2 neuron dan fungsi aktivasi softmax, yang menghasilkan probabilitas kelas untuk dua kategori yaitu RD dan Normal. Regularisasi kernel L2 juga diterapkan pada lapisan ini untuk memastikan model tidak terlalu overfit pada data pelatihan.

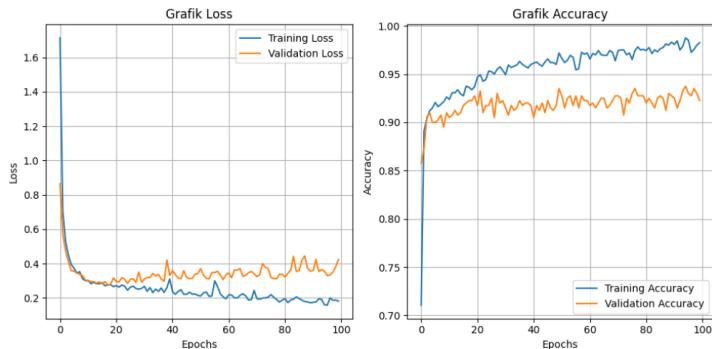
4.4.3 Training Model

Setelah merancang arsitektur model, dilakukan proses training dengan beberapa percobaan hyperparameter yaitu optimizer menggunakan adam dengan learning rate 0.001, dan 0.0001. Pemilihan epoch dilakukan dengan percobaan menggunakan epoch 100. Berikut tabel percobaan training pada dataset.

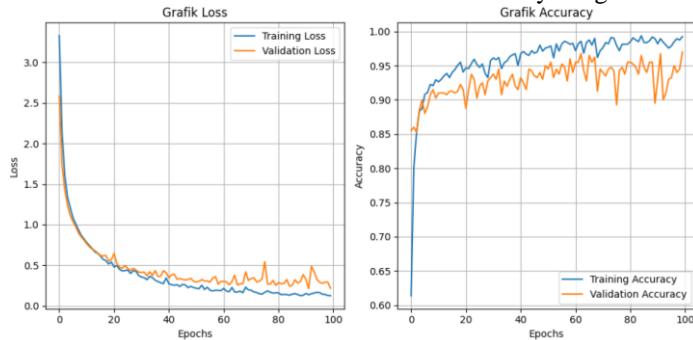
Tabel 4. 5 Training Dataset

No	Model		Learning Rate	Training Loos	Training Akurasi	Validasi Loos	Validasi Akurasi
1	80:20	Model CNN	0.001	0.1854	0.9793	0.4235	0.9225
2			0.0001	0.1254	0.9923	0.2173	0.9700
3			0.001	0.1743	0.9743	0.4354	0.9183
4			0.0001	0.1644	0.9829	0.4149	0.9267
5			0.001	0.1493	0.9850	0.3477	0.9362
6			0.0001	0.1588	0.9808	0.4380	0.8988
7			0.001	0.1587	0.9910	0.5322	0.9030
8			0.0001	0.1684	0.9910	0.4353	0.9120

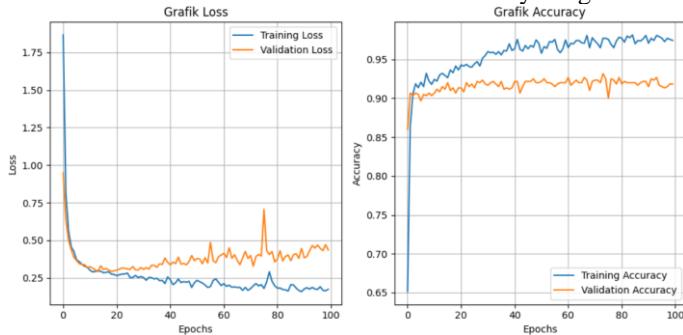
Berikut adalah grafik model *loss* dan model *accuracy* dari hasil training pada tabel 4.4 dengan arsitektur model CNN.



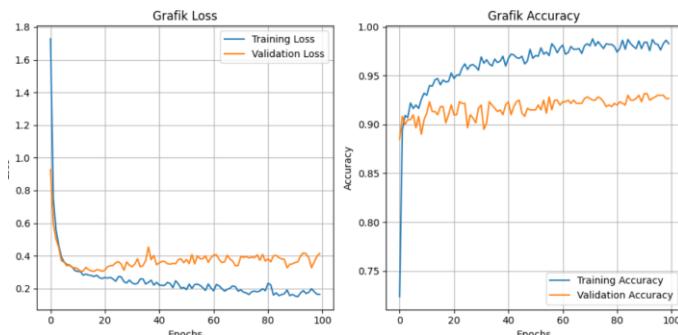
Gambar 4. 11 80:20 Grafik Model Loss dan Accuracy dengan LR 0.001



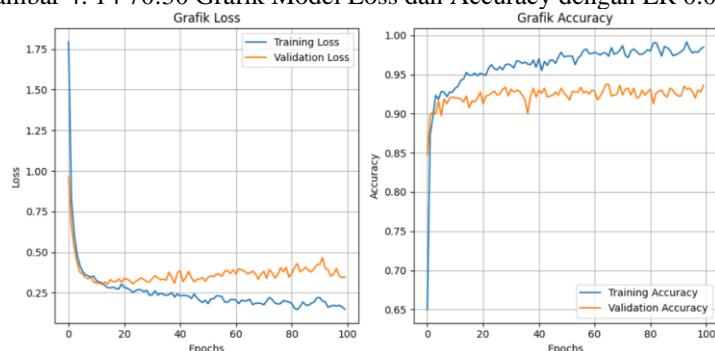
Gambar 4. 12 80:20 Grafik Model Loss dan Accuracy dengan LR 0.0001



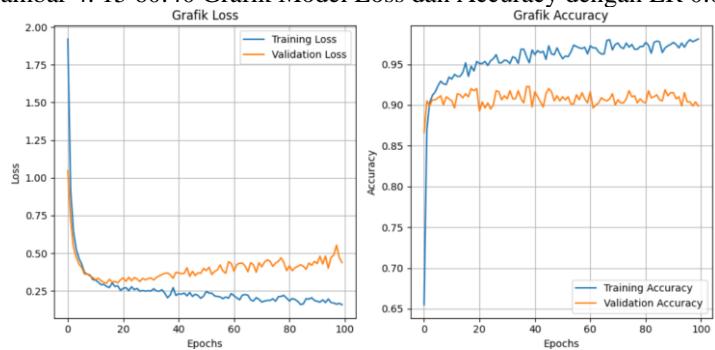
Gambar 4. 13 70:30 Grafik Model Loss dan Accuracy dengan LR 0.001



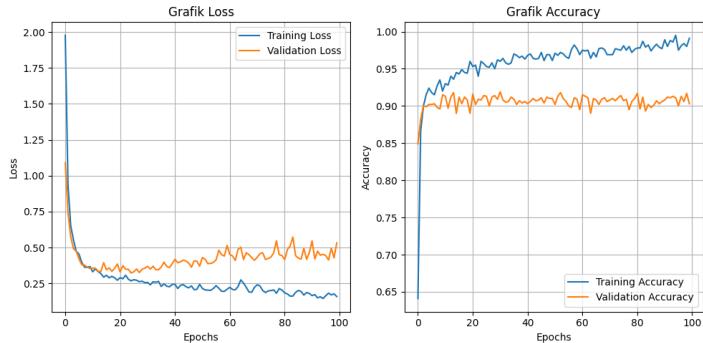
Gambar 4. 14 70:30 Grafik Model Loss dan Accuracy dengan LR 0.0001



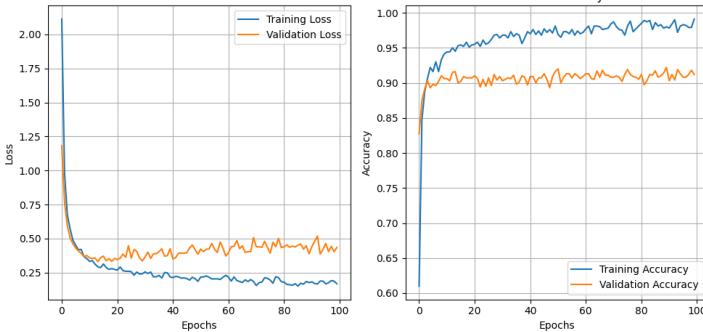
Gambar 4. 15 60:40 Grafik Model Loss dan Accuracy dengan LR 0.001



Gambar 4. 16 60:40 Grafik Model Loss dan Accuracy dengan LR 0.0001



Gambar 4. 17 50:50 Grafik Model Loss dan Accuracy dengan LR 0.001



Gambar 4. 18 50:50 Grafik Model Loss dan Accuracy dengan LR 0.0001

Tabel 4.4 merupakan hasil evaluasi dari berbagai skenario percobaan. Setiap baris tabel merupakan satu scenario percobaan yang berbeda, dengan memuat informasi rasio pembagian data, jenis arsitektur model, learning rate, dan jumlah epoch yang digunakan. Hasil evaluasi model termasuk training loss, training akurasi, validasi loss dan validasi akurasi dicatat pada setiap scenario percobaan. Training loss dan validasi loss mengukur sejauh mana prediksi model dari data train dan validasi mendekati label yang sebenarnya. Training akurasi dan validasi akurasi mengukur sejauh mana model mampu mengklasifikasikan data pelatihan dengan benar.

Berikut hasil dari Accuracy, Specificity, Sensitivity, AUC yang didapat dari percobaan table 4.4.

Tabel 4. 6 Validasi Akurasi

No	Model		Learning Rate	Accuracy	Specificity	Sensitivity	AUC	F1-score
1	80:20	Model CNN	0.001	0.91	0.86	0.98	0.98	0.92
2			0.0001	0.92	0.96	0.98	0.99	0.96
3			0.001	0.91	0.92	0.91	0.97	0.91
4			0.0001	0.91	0.91	0.94	0.98	0.92
5			0.001	0.92	0.93	0.94	0.98	0.93
6			0.0001	0.9	0.88	0.91	0.97	0.89
7			0.001	0.9	0.85	0.95	0.96	0.9
8			0.0001	0.9	0.88	0.94	0.97	0.91

Learning rate 0.001 memberikan performa yang bervariasi tergantung pada rasio pembagian data. Pada pembagian data 80:20, model menunjukkan sensitivity yang sangat tinggi (0.98) namun specificity lebih rendah (0.86), menunjukkan kecenderungan model untuk overfitting pada kelas positif karena belajar terlalu cepat dari data latih yang besar. Pada pembagian data 70:30, model menunjukkan performa yang lebih seimbang dengan specificity (0.92) dan sensitivity (0.91), yang menunjukkan bahwa dengan lebih banyak data uji, model dapat belajar lebih baik dan mengurangi overfitting. Pada pembagian data 60:40, model memberikan performa yang baik dengan specificity (0.93) dan F1-score (0.93), menunjukkan bahwa model dapat memanfaatkan data latih yang lebih besar dengan belajar lebih cepat dan mampu menangkap variasi data dengan baik. Pada pembagian data 50:50, model menunjukkan sensitivity yang tinggi (0.95) tetapi specificity lebih rendah (0.85), menunjukkan bahwa model mungkin terlalu fokus pada kelas positif, mengorbankan kemampuan untuk mengenali kelas negatif.

Learning rate 0.0001 secara konsisten memberikan performa yang lebih baik terutama pada pembagian data 80:20 dan 70:30. Pada pembagian data 80:20, learning rate 0.0001 memberikan hasil terbaik dengan specificity (0.96), AUC (0.99), dan F1-score (0.96). Hal ini disebabkan oleh model yang belajar lebih hati-hati dengan learning rate yang lebih kecil, sehingga menghindari overfitting dan menghasilkan performa yang lebih baik dalam mengenali kelas negatif. Pada pembagian data 70:30, learning rate 0.0001 juga menunjukkan keseimbangan yang baik antara specificity (0.91) dan sensitivity (0.94), menghasilkan F1-score (0.92) yang tinggi. Learning rate yang lebih kecil membantu model belajar dengan lebih detail, menangkap pola dari data latih yang cukup besar tanpa overfitting.

Namun, pada pembagian data 60:40, learning rate 0.0001 tidak memberikan performa sebaik learning rate 0.001, menunjukkan bahwa dengan lebih banyak data latih, model membutuhkan kecepatan pembelajaran yang lebih tinggi untuk menangkap pola dengan baik. Pada pembagian data 50:50, learning rate 0.0001 menunjukkan sedikit peningkatan dalam specificity (0.88) dan F1-score (0.91) dibandingkan dengan learning rate 0.001, menunjukkan bahwa model dengan learning rate kecil lebih baik dalam mengenali kelas negatif meskipun jumlah data uji lebih besar.

Secara keseluruhan, pembagian data rasio 80:20 dengan learning rate 0.0001 memberikan hasil terbaik secara keseluruhan, dengan metrik evaluasi tertinggi, terutama specificity (0.96), AUC (0.99), dan F1-score (0.96). Hal ini menunjukkan bahwa model memiliki kemampuan yang baik untuk mengidentifikasi kelas negatif dan mampu menjaga keseimbangan antara precision dan recall.

4.5 Validasi

Pada tahap ini, peneliti berusaha membuktikan bahwa metode early fusion lebih unggul dibandingkan dengan penggunaan dataset Unimodal. Untuk membuktikannya, dilakukan pengujian dengan menggunakan dataset fundus dan OCT, yang keduanya dilatih menggunakan model CNN yang sama. Berikut adalah hasil validasi akurasi dari dataset fundus dan OCT:

Tabel 4. 7 Validasi Akurasi Fundus dan OCT

No	Dataset	Model		Learning Rate	Accuracy	Specificity	Sensitivity	AUC	F1-score		
1	Fundus	80:20	Model CNN	0.001	0.91	0.87	0.93	0.96	0.93		
2				0.0001	0.9	0.94	0.97	0.99	0.95		
3	OCT			0.001	0.85	0.91	0.94	0.96	0.95		
4				0.0001	0.89	0.97	0.94	0.99	0.95		

Selanjutnya, peneliti melakukan uji coba perbandingan transparansi alpha blending pada teknik early fusion yang sebelumnya telah diterapkan. Dalam eksperimen ini, peneliti mengatur rasio transparansi dengan bobot berbeda: 0,6 untuk citra fundus dan 0,4 untuk citra OCT. Hasil validasi akurasi yang diperoleh dari percobaan ini adalah sebagai berikut:

Tabel 4. 8 Validasi Akurasi Dari Alpha Blending Dengan Bobot Fundus 0,6 dan OCT 0,4

No	Model		Learning Rate	Accuracy	Specificity	Sensitivity	AUC	F1-score
1	80:20	Model CNN	0.001	0.9	0.9	0.94	0.97	0.92
2			0.0001	0.92	0.92	0.96	0.98	0.94
3			0.001	0.91	0.9	0.96	0.98	0.93
4			0.0001	0.91	0.92	0.94	0.98	0.93
5			0.001	0.91	0.93	0.91	0.97	0.92
6			0.0001	0.91	0.93	0.92	0.97	0.92
7			0.001	0.89	0.85	0.91	0.91	0.9
8			0.0001	0.87	0.88	0.9	0.89	0.91

Dari hasil percobaan di atas, terlihat bahwa teknik early fusion berhasil memberikan akurasi yang lebih baik dibandingkan dengan penggunaan data unimodal. Selain itu, penggunaan transparansi alpha blending dengan bobot 0,7 untuk fundus dan 0,3 untuk OCT terbukti membrikan hasil akurasi yang baik selama pelatihan model CNN. Temuan ini sesuai dengan data yang ditampilkan pada Tabel 4.6.

4.6 Implementasi Pengujian

4.6.1 Inputan Citra.

Inputan citra untuk implementasi pengujian diperoleh dari Kaggle³, yang berisi validasi citra fundus. Sementara itu, citra OCT didapatkan dari dataset yang sama pada proses penelitian yaitu Table 4.1 dikarenakan keterbatasan data yang diperoleh oleh peneliti

4.6.2 Preprocessing.

Dilakukan tahapan uji coba klasifikasi dengan cara membuat kembali tahapan *preprocessing*, tahapan tersebut mencakup *Resize*, *Grayscale*, *Alpha Blending*, *Concatinate*, dan juga dengan beberapa tambahan fungsi klasifikasi. Kemudian dilakukan pemanggilan model dengan kurasi yang terbaik dari table 4.5 yang sudah dilatih dengan format HDF5 (Hierarchical Data Format version 5).

³<https://www.kaggle.com/datasets/pkdarabi/diagnosis-of-diabetic-retinopathy>

```

import numpy as np
import cv2
from tkinter import Tk, Button, Label, filedialog, messagebox, StringVar, Frame
from PIL import Image, ImageTk
from tensorflow.keras.models import load_model

# Fungsi untuk menambahkan saluran alfa ke gambar
def add_alpha_channel(image):
    b, g, r = cv2.split(image)
    alpha = np.ones(b.shape, dtype=b.dtype) * 255 # Membuat saluran alfa penuh (tidak transparan)
    return cv2.merge((b, g, r, alpha))

# Fungsi untuk mengubah gambar menjadi grayscale dan menambahkan saluran alfa
def convert_to_grayscale_with_alpha(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    alpha = np.ones(gray.shape, dtype=gray.dtype) * 255 # Membuat saluran alfa penuh (tidak transparan)
    return cv2.merge((gray, gray, gray, alpha))

# Fungsi untuk memuat gambar dengan pengecekan
def load_image_with_check(path):
    image = cv2.imread(path)
    if image is None:
        messagebox.showerror("Error", f"Gambar pada path {path} tidak berhasil dimuat.")
    return image

# Fungsi untuk memilih file gambar
def select_file():
    file_path = filedialog.askopenfilename()
    return file_path

def show_image(image, label, frame_width, frame_height):
    image = cv2.resize(image, (frame_width, frame_height))
    image = Image.fromarray(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    image_tk = ImageTk.PhotoImage(image)
    label.config(image=image_tk)
    label.image = image_tk

def load_gambar1():
    global gambar1
    file_path = select_file()
    gambar1 = load_image_with_check(file_path)
    if gambar1 is not None:
        show_image(gambar1, label_gambar1_image, 224, 224)

def load_gambar2():
    global gambar2
    file_path = select_file()
    gambar2 = load_image_with_check(file_path)
    if gambar2 is not None:
        show_image(gambar2, label_gambar2_image, 224, 224)

def process_and_predict():
    global gambar1, gambar2, concatenate

    if gambar1 is None or gambar2 is None:
        messagebox.showerror("Error", "Silakan muat kedua gambar terlebih dahulu.")
        return

    # Mengubah ukuran gambar menjadi 224x224
    gambar1_resized = cv2.resize(gambar1, (224, 224))
    gambar2_resized = cv2.resize(gambar2, (224, 224))

    # Mengubah ukuran gambar menjadi grayscale dan menambahkan saluran alfa
    gambar1_gray_with_alpha = convert_to_grayscale_with_alpha(gambar1_resized)
    gambar2_gray_with_alpha = convert_to_grayscale_with_alpha(gambar2_resized)

    # Menentukan Lebar area yang akan ditimpak
    width_to_overlay = 224

    # Mengubah ukuran gambar pertama menjadi dua bagian
    gambar1_left_with_alpha = gambar1_gray_with_alpha[:, :width_to_overlay]
    gambar1_right_with_alpha = gambar1_gray_with_alpha[:, width_to_overlay:]

    # Mengubah ukuran gambar kedua menjadi bagian yang akan ditimpak
    gambar2_overlay_with_alpha = gambar2_gray_with_alpha[:, :width_to_overlay]

    # Mengatur transparansi pada bagian yang akan digabungkan
    alpha_gambar1 = 0.7 # 70% transparansi
    alpha_gambar2 = 0.3 # 30% transparansi

    # Melakukan blending pada bagian yang akan digabungkan
    blended_overlay = cv2.addWeighted(gambar1_left_with_alpha, alpha_gambar1, gambar2_overlay_with_alpha, alpha_gambar2, 0)

    # Menggabungkan dengan menempatkan bagian kiri gambar pertama dengan hasil blending
    concatenate = np.concatenate([blended_overlay, gambar1_right_with_alpha], axis=1)

    # Preprocessing gambar untuk model (sesuai dengan preprocessing saat melatih model)
    concatenate_preprocessed = concatenate[:, :, :] / 255.0 # Normalisasi gambar

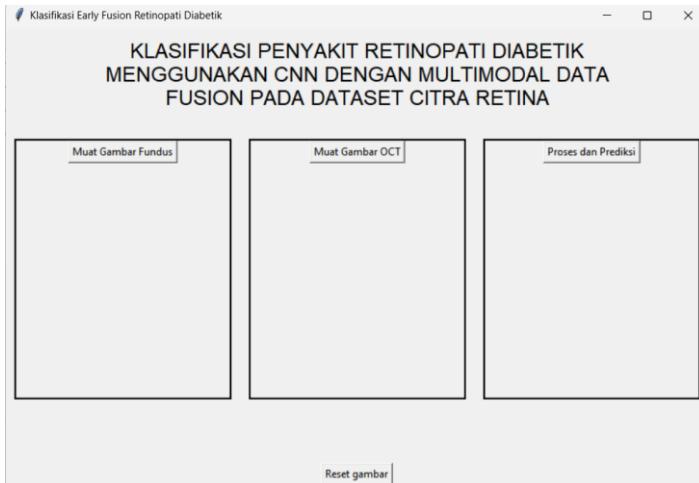
    # Memuat model
    model = load_model('Training_Testing/PA_alumn/CNN_Exception_model.h5')

```

Gambar 4. 19 Proses Klasifikasi

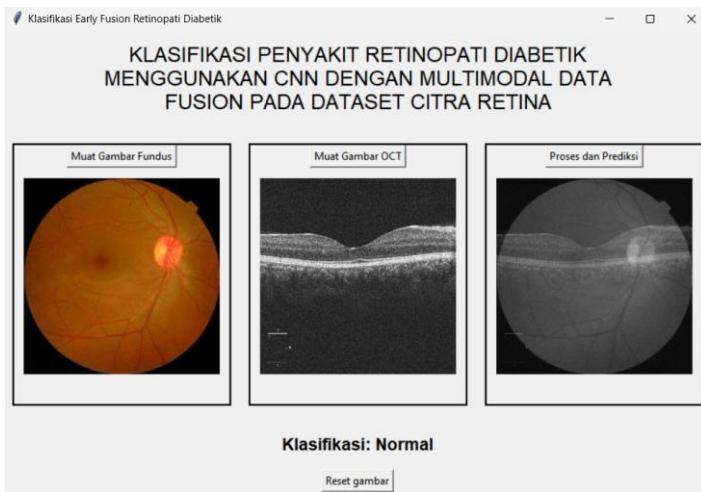
4.6.3 Inputan Citra.

Pada tahapan selanjutnya dilakukan penginputan citra pada melalui library pemograman Tkinter (Python's standard GUI library) untuk membuat antarmuka pengguna grafis (GUI). Tkinter adalah toolkit antarmuka pengguna standar untuk Python, yang menyediakan berbagai macam widget untuk membangun aplikasi GUI, seperti tombol, label, dan kotak teks. Berikut tampilan Ketika program dijalankan:



Gambar 4. 20 Tampilan antarmuka klasifikasi Retinopati Diabetik.

Kemudian, pilih *button* muat gambar fundus untuk melimilih gambar fundus dari folder dan pilih *button* muat gambar OCT untuk melimilih gambar OCT dari folder gambar. Ketika sudah menginputkan gambar fundus dan OCT maka selanjutnya pilih *button* Proses dan Prediksi untuk menampilkan hasil gambar yang telah digabungkan dengan teknik Early Fusion beserta hasil klasifikasinya. Berikut hasilnya:



Gambar 4. 21 Hasil Klasifikasi Retinopati Diabetik

4.7 Analisis

Berdasarkan hasil percobaan *training* model pada tabel 4.5, tabel 4.6, dan grafik pada gambar 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, dan 4.18 diketahui bahwa learning rate 0.0001 cenderung memberikan performa yang lebih baik pada skenario pembagian data 80:20 dan 70:30, terutama pada Specificity dan F1-score. Namun, pada skenario 60:40, *learning rate* 0.001 memberikan peforma yang baik juga, hal ini disebabkan penggunaan *learning rate* 0.0001 cenderung membuat model belajar lebih lambat dan hati-hati, yang dapat menghindari *overfitting* terutama pada set data pelatihan yang lebih besar seperti pada data pembagian dengan rasio 80:20 dan 70:30 yang dapat menghasilkan Specificity dan F1-score yang lebih baik. Sedangkan pada learning rate 0.001 dari scenario pembagian data 60:40 menghasilkan akurasi yang baik daripada scenario pembagian lainnya, hal tersebut dikarenakan model belajar lebih cepat dari variasi data yang lebih besar dalam set pelatihan yang lebih kecil. Sehingga pada penelitian ini menghasilkan kesimpulan bahwa data pembagian rasio 80:20 dengan *Learning Rate* 0.0001 memberikan hasil terbaik secara keseluruhan, dengan metrik evaluasi tertinggi, terutama Specificity (0.96), AUC (0.99), dan F1-score (0.96). Hal ini menunjukkan bahwa model memiliki kemampuan yang baik untuk mengidentifikasi kelas negatif dan mampu menjaga keseimbangan antara precision dan recall.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pengujian dan analisis yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

1. klasifikasi Early Fusion dari penggabungan citra fundus dan OCT berhasil dibagun dengan menggunakan *Convolutional Neural Network*.
2. Bedasarkan hasil proses *training*, pembagian dataset dari rasio 80:20, dengan *learning rate* 0.0001 dan epoch 100 menghasilkan peforma terbaik dengan mendapatkan nilai rata-rata validasi akurasi tertinggi sebesar 92%.

5.2 Saran

Adapun saran untuk pengembangan proyek akhir selanjutnya adalah:

1. Pada proses pengumpulan dataset, penelitian selanjutnya dapat melakukan penambahan dataset yang lebih banyak terutama dataset dari citra OCT, sehingga menghasilkan akurasi yang lebih baik lagi.
2. Pengembangan model arsitektur CNN dengan transfer learning seperti Xception, ResNet, VGGNet, Inception. Hal ini dikarenakan pada pemodelan dengan CNN saja cenderung membuat lebih banyak layer yang kurang dipakai sehingga menghasilkan grafik yang sering fluktasi atau overfitting dalam pemodelannya.
3. Pengembangan fusion dari citra fundus dan OCT retinopati diabetic dengan pendekatan teknik Late Fusion dapat menjadi solusi klasifikasi lebih baik.

DAFTAR PUSTAKA

- Gayathri, S., Gopi, V. P., & Palanisamy, P. (2021). Diabetic retinopathy classification based on multipath CNN and machine learning classifiers. *Physical and Engineering Sciences in Medicine*, 44(3), 639-653. <https://doi.org/10.1007/s13246-021-01012-3>.
- Mansour, M. (2018). Deep learning for medical image analysis: A review. *Journal of Healthcare Engineering*, 2018, 1-16.
- Hua, J., Lu, Y., Li, Y., & Wu, Y. (2022). Deep learning for medical image analysis: A review. *Computational and Mathematical Methods in Medicine*, 2022, 1-17.
- Wu, Y.-H., Huang, C.-W., & Kuo, Y.-H. (2022). A multi-modal deep learning approach for diabetic retinopathy screening. *Biomedical Optics Express*, 13(2), 1048-1060.
- Hu, X.-Q., He, L.-Y., & Li, X.-F. (2021). Multi-modal deep learning for early detection of diabetic retinopathy. *IEEE Transactions on Biomedical Engineering*, 68(12), 3316-3324.
- Singh, S., Kumar, P., & Kumar, D. (2020). A deep learning approach for diabetic retinopathy detection using multi-modal retinal imaging. *Journal of Medical Imaging*, 7(4), 046501.
- A.K. Garg, S. S. (2022). Fundus Photography for the Detection of Diabetic Retinopathy Using Machine Learning . *IEEE Access*.
- Chollet, F. (2017). *Deep Learning with Python*. Manning Publications
- Curtis, A. (2024, 01 09). *Multimodal AI Explained*. Retrieved from splunk:
https://www.splunk.com/en_us/blog/learn/multimodal-ai.html

Dr. Anita Dhanorkar, B. (2024). *What Is the Purpose of Fundus Photography?* Retrieved from MedicineNet : https://www.medicinenet.com/heart_health_palpitations_causes/article.htm

Eugene Yu-Chuan Kang, L. Y.-L.-H.-Y.-P.-Z.-F.-C. (2019). A Multimodal Imaging-Based Deep Learning Model for Detecting Treatment-Requiring Retinal Vascular Diseases: Model Development and Validation Study. *IEEE Access*.

Hongyu Zhou, Y. P. (2022). Deep Fusion of Fundus and OCT Images for Diabetic Retinopathy Classification. *Computer Methods and Programs in Biomedicine*.

Izzati, S. F. (2023, 11 02). *Waspadai Komplikasi Diabetes pada Mata, Bisa Picu Kebutaan.* Retrieved from detikHealth: <https://health.detik.com/berita-detikhealth/d-7016289/waspadai-komplikasi-diabetes-pada-mata-bisa-picu-kebutaan>

J.F.W. van der Steen, D. v. (2012). Fundus Photography: A Review of Techniques and Applications . *Eye*.

Kevyn Alifian Hernanda Wibowo, A. (2023). Klasifikasi Diabetic Retinopathy Menggunakan CNN dengan Arsitektur yang Dimodifikasi . *Indonesian Journal of Mathematics and Natural Sciences*, 8.

Kumar, S. K. (2018). Diabetic Retinopathy Detection by Extracting Area and Number of Microaneurysm from Colour Fundus Image. *5th International Conference on Signal Processing and Integrated Networks (SPIN)* , (pp. 359-364.).

M.D., M. J. (2023, 10 21). *Optical Coherence Tomography.* Retrieved from EyeWiki: https://eyewiki.org/Optical_Coherence_Tomography

- Notomoro. (2023, 10 30). *A Deep Dive into Multimodal Models: Enhancing AI Applications*. Retrieved from Webisoft Articles: <https://webisoft.com/articles/multimodal-model/>
- Perkovic, L. (2012). *Introduction to computing using Python*. English: J. Wiley & Sons, Hoboken, NJ.
- S.K. Jain, R. J. (2001). *Handbook of Image Processing*. Academic Press .
- Sagala, L. O. (2022). Classification Of Diabetic Retinopathy by Deep learning. 4.
- Sailellah, H. R. (2023, April 28). "Pengertian OS Linux: Sejarah, Fungsi, Kelebihan, dan Kekurangan". Retrieved from Telkom University Blog: <https://it.telkomuniversity.ac.id/pengertian-os-linux/>
- Saputri, N. (2022, 2 23). *hello sehat*. Retrieved from Retinopati Diabetik: <https://hellosehat.com/diabetes/komplikasi-diabetes/retinopati-diabetik/>
- Trivusi. (2022, 09 16). *Normalisasi Data: Pengertian, Tujuan, dan Metodenya*. Retrieved from Trivusi: <https://www.trivusi.web.id/2022/09/normalisasi-data.html>
- Yifan Peng, J. Z. (2023). Joint Representation Learning of Fundus and OCT Images for Diabetic Retinopathy Classification. *IEEE Transactions on Medical Imaging*.
- Yihao Li, M. E.-H. (2022). Multimodal Information Fusion for Glaucoma and Diabetic Retinopathy Classification. 11.
- Yuxin Chen, Y. P. (2023). Multimodal Diabetic Retinopathy Classification Using Fundus, OCT, and FA Images. *Artificial Intelligence in Medicine*.

Zewen Li, F. L. (2021). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Xplore logo*, 6999 - 7019.

Zhong-Qiu Zhao, P. Z.-t. (2019, 04 16). Object Detection with Deep Learning: A Review. *Cornell University*.

LAMPIRAN

Keterangan	Kode Program
Augemntasi Dataset OCT	<pre> source_dir = 'Training_Testing/PA_alumn/data ori/oct/OCT DR' target_dir = 'Training_Testing/PA_alumn/data ori/oct_augmentasi/DR' if not os.path.exists(target_dir): os.makedirs(target_dir) # Menghitung sudut rotasi rotation_angles = np.linspace(0, 360, 10, endpoint=False) # Membuat 9 sudut, mengabaikan 360 karena sama dengan 0 augmented_images_count = 0 # Penghitung untuk gambar yang telah diaugmentasi for img_name in os.listdir(source_dir): img_path = os.path.join(source_dir, img_name) for i, angle in enumerate(rotation_angles): save_path = os.path.join(target_dir, f'oct_{int(angle)}_{img_name}') if rotate_image(img_path, angle, save_path): augmented_images_count += 1 if augmented_images_count >= 1000: print("Target 1000 ocr_DR gambar telah berhasil diaugmentasi.") break # Keluar dari loop jika target tercapai if augmented_images_count >= 1000: break # Keluar dari loop luar jika target tercapai if augmented_images_count < 1000: print(f"Proses selesai. Total gambar yang diaugmentasi: {augmented_images_count}.") else: print("Proses augmentasi gambar selesai dengan sukses.") </pre>
Early Fusion DR	<pre> import os import cv2 import numpy as np import matplotlib.pyplot as plt def add_alpha_channel(image): b_channel, g_channel, r_channel = cv2.split(image) alpha_channel = np.ones(b_channel.shape, dtype=b_channel.dtype) * 255 return cv2.merge((b_channel, g_channel, r_channel, alpha_channel)) # Fungsi untuk menggabungkan dua gambar dengan np.concatenate def concatenate_images(image1, image2, alpha1=0.6, alpha2=0.4): image1_with_alpha = add_alpha_channel(image1) image2_with_alpha = add_alpha_channel(image2) width_to_overlay = min(image1_with_alpha.shape[1], image2_with_alpha.shape[1]) image1_left = image1_with_alpha[:, :width_to_overlay] image1_right = image1_with_alpha[:, width_to_overlay:] image2_overlay = image2_with_alpha[:, :width_to_overlay] blended_overlay = cv2.addWeighted(image1_left, alpha1, image2_overlay, alpha2, 0) concatenated_image = np.concatenate((blended_overlay, image1_right), axis=1) return concatenated_image # Path ke folder gambar fundus_dr_folder = 'data garslece/fundus/DR' oct_dr_folder = 'data garslece/oct/DR' output_folder = 'revisiBukJuni/DR' # Membuat folder output jika belum ada if not os.path.exists(output_folder): os.makedirs(output_folder) # Membaca semua gambar dari folder DR fundus fundus_dr_images = [] fundus_dr_filenames = [] for filename in os.listdir(fundus_dr_folder): img_path = os.path.join(fundus_dr_folder, filename) if os.path.isfile(img_path): img = cv2.imread(img_path) fundus_dr_images.append(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)) fundus_dr_filenames.append(filename) </pre>

	<pre> # Membaca semua gambar dari folder DR oct oct_dr_images = [] oct_dr_filenames = [] for filename in os.listdir(oct_dr_folder): img_path = os.path.join(oct_dr_folder, filename) if os.path.isfile(img_path): img = cv2.imread(img_path) oct_dr_images.append(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)) oct_dr_filenames.append(filename) # Mengatur jumlah gambar yang akan diolah num_images_to_merge = 1000 # Ubah sesuai dengan jumlah gambar yang ingin Anda proses # Menggabungkan beberapa gambar dari folder DR fundus dan DR oct dan menyimpannya ke folder output concat_images = [] num_images_to_merge = min(num_images_to_merge, min(len(fundus_dr_images), len(oct_dr_images))) for i in range(num_images_to_merge): concat_image = concatenate_images(fundus_dr_images[i], oct_dr_images[i]) concat_images.append(concat_image) # Menampilkan gambar hasil penggabungan ke folder output output_filename = f'fusion_DR_{i+1}.png' output_path = os.path.join(output_folder, output_filename) cv2.imwrite(output_path, cv2.cvtColor(concat_image, cv2.COLOR_RGB2BGR)) print(f'Gambar {output_filename} berhasil disimpan di {output_path}.') # Menampilkan hasil penggabungan beberapa contoh gambar num_images_to_show = min(3, num_images_to_merge) plt.figure(figsize=(15, 5*num_images_to_show)) for i in range(num_images_to_show): plt.subplot(num_images_to_show, 3, i*3 + 1) plt.imshow(fundus_dr_images[i]) plt.title('Fundus') plt.axis('off') plt.subplot(num_images_to_show, 3, i*3 + 2) plt.imshow(oct_dr_images[i]) plt.title('OCT') plt.axis('off') plt.subplot(num_images_to_show, 3, i*3 + 3) plt.imshow(concat_images[i]) plt.title('Hasil Penggabungan') plt.axis('off') plt.tight_layout() plt.show() </pre>
Early Fusion Normal	<pre> import os import cv2 import numpy as np import matplotlib.pyplot as plt def add_alpha_channel(image): b_channel, g_channel, r_channel = cv2.split(image) alpha_channel = np.ones(b_channel.shape, dtype=b_channel.dtype) * 255 return cv2.merge((b_channel, g_channel, r_channel, alpha_channel)) # Fungsi untuk menggabungkan dua gambar dengan np.concatenate def concatenate_images(image1, image2, alpha1=0.6, alpha2=0.4): image1_with_alpha = add_alpha_channel(image1) image2_with_alpha = add_alpha_channel(image2) width_to_overlay = min(image1_with_alpha.shape[1], image2_with_alpha.shape[1]) image1_left = image1_with_alpha[:, :width_to_overlay] image1_right = image1_with_alpha[:, width_to_overlay:] image2_overlay = image2_with_alpha[:, :width_to_overlay] blended_overlay = cv2.addWeighted(image1_left, alpha1, image2_overlay, alpha2, 0) concatenated_image = np.concatenate((blended_overlay, image1_right), axis=1) return concatenated_image # Path ke folder gambar fundus_normal_folder = 'data_garslece/fundus/Normal' oct_normal_folder = 'data_garslece/oct/Normal' output_folder = 'revisiBukUji/Normal' # Membuat folder output jika belum ada if not os.path.exists(output_folder): os.makedirs(output_folder) # Membaca semua gambar dari folder Normal fundus fundus_normal_images = [] fundus_normal_filenames = [] for filename in os.listdir(fundus_normal_folder): img_path = os.path.join(fundus_normal_folder, filename) if os.path.isfile(img_path): img = cv2.imread(img_path) fundus_normal_images.append(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)) fundus_normal_filenames.append(filename) </pre>

	<pre> # Membaca semua gambar dari folder Normal oct oct_normal_images = [] oct_normal_filenames = [] for filename in os.listdir(oct_normal_folder): img_path = os.path.join(oct_normal_folder, filename) if os.path.isfile(img_path): img = cv2.imread(img_path) oct_normal_images.append(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)) oct_normal_filenames.append(filename) # Mengatur jumlah gambar yang akan dialoh num_images_to_merge = 1000 # Ubah sesuai dengan jumlah gambar yang ingin Anda proses # Menggabungkan beberapa gambar dari folder Normal fundus dan Normal oct dan menyimpannya ke folder output concat_images = [] num_images_to_merge = min(num_images_to_merge, min(len(fundus_normal_images), len(oct_normal_images))) for i in range(num_images_to_merge): concat_image = concatenate_images(fundus_normal_images[i], oct_normal_images[i]) concat_images.append(concat_image) # Menyimpan gambar hasil penggabungan ke folder output output_filename = f'fusion_Normal_{i+1}.png' output_path = os.path.join(output_folder, output_filename) cv2.imwrite(output_path, cv2.cvtColor(concat_image, cv2.COLOR_RGB2BGR)) print(f'Gambar {output_filename} berhasil disimpan di {output_path}') # Menampilkan hasil penggabungan beberapa contoh gambar num_images_to_show = min(3, num_images_to_merge) plt.figure(figsize=(15, 5*num_images_to_show)) for i in range(num_images_to_show): plt.subplot(num_images_to_show, 3, i*3 + 1) plt.imshow(fundus_normal_images[i]) plt.title('fundus') plt.axis('off') plt.subplot(num_images_to_show, 3, i*3 + 2) plt.imshow(oct_normal_images[i]) plt.title('OCT') plt.axis('off') plt.subplot(num_images_to_show, 3, i*3 + 3) plt.imshow(concat_images[i]) plt.title('Hasil Penggabungan') plt.axis('off') plt.tight_layout() plt.show() </pre>
Resize dan GrayScale OCT	<pre> import os from PIL import Image # Tentukan direktori utama input dan output input_base_dir = 'data ori/oct_augmentasi' output_base_dir = 'data garslece/oct' sub_dirs = ['DR', 'Normal'] # Ukuran target target_size = (224, 224) # Fungsi untuk resize dan convert ke grayscale def process_images(input_dir, output_dir): if not os.path.exists(output_dir): os.makedirs(output_dir) for filename in os.listdir(input_dir): if filename.endswith('.jpg') or filename.endswith('.png') or filename.endswith('.jpeg'): img_path = os.path.join(input_dir, filename) img = Image.open(img_path) img = img.resize(target_size).convert('L') output_path = os.path.join(output_dir, filename) img.save(output_path) # Proses untuk tiap sub direktori for sub_dir in sub_dirs: input_dir = os.path.join(input_base_dir, sub_dir) output_dir = os.path.join(output_base_dir, sub_dir) process_images(input_dir, output_dir) print("Proses selesai!") </pre>

<h3>Resize dan GrayScale DR</h3>	<pre> import os from PIL import Image # Tentukan direktori utama input dan output input_base_dir = 'data ori/fundus' output_base_dir = 'data garslece/fundus' sub_dirs = ['DR', 'Normal'] # Ukuran target target_size = (224, 224) # Fungsi untuk resize dan convert ke grayscale def process_images(input_dir, output_dir): if not os.path.exists(output_dir): os.makedirs(output_dir) for filename in os.listdir(input_dir): if filename.endswith('.jpg') or filename.endswith('.png') or filename.endswith('.jpeg'): img_path = os.path.join(input_dir, filename) img = Image.open(img_path) img = img.resize(target_size).convert('L') output_path = os.path.join(output_dir, filename) img.save(output_path) # Proses untuk tiap sub direktori for sub_dir in sub_dirs: input_dir = os.path.join(input_base_dir, sub_dir) output_dir = os.path.join(output_base_dir, sub_dir) process_images(input_dir, output_dir) print("Proses selesai!") </pre>
<h3>Split Data Train dan Test 80:20</h3>	<pre> import os import shutil from sklearn.model_selection import train_test_split # Path ke folder dataset folder_DR = 'revisiBukJuni/DR' folder_normal = 'revisiBukJuni/Normal' # Path ke folder output output_train_DR = 'revisiBukJuni/data 80 20/train/DR' output_train_normal = 'revisiBukJuni/data 80 20/train/normal' output_test_DR = 'revisiBukJuni/data 80 20/test/DR' output_test_normal = 'revisiBukJuni/data 80 20/test/normal' # Membuat folder output jika belum ada os.makedirs(output_train_DR, exist_ok=True) os.makedirs(output_train_normal, exist_ok=True) os.makedirs(output_test_DR, exist_ok=True) os.makedirs(output_test_normal, exist_ok=True) # Fungsi untuk membagi data def split_data(folder, output_train, output_test, test_size=0.2): # Mengambil semua file gambar dalam folder images = [f for f in os.listdir(folder) if os.path.isfile(os.path.join(folder, f))] # Membagi data menjadi train dan test train_images, test_images = train_test_split(images, test_size=test_size, random_state=42) # Menyalin file gambar ke folder train for image in train_images: shutil.copy(os.path.join(folder, image), os.path.join(output_train, image)) # Menyalin file gambar ke folder test for image in test_images: shutil.copy(os.path.join(folder, image), os.path.join(output_test, image)) # Membagi data untuk folder DR split_data(folder_DR, output_train_DR, output_test_DR) # Membagi data untuk folder normal split_data(folder_normal, output_train_normal, output_test_normal) </pre>

Split Data Train dan Test 70:30	<pre> import os import shutil from sklearn.model_selection import train_test_split # Path ke folder dataset folder_DR = 'revisibukJuni/DR' folder_normal = 'revisibukJuni/Normal' # Path ke folder output output_train_DR = 'revisibukJuni/data 60 40/train/DR' output_train_normal = 'revisibukJuni/data 60 40/train/normal' output_test_DR = 'revisibukJuni/data 60 40/test/DR' output_test_normal = 'revisibukJuni/data 60 40/test/normal' # Membuat folder output jika belum ada os.makedirs(output_train_DR, exist_ok=True) os.makedirs(output_train_normal, exist_ok=True) os.makedirs(output_test_DR, exist_ok=True) os.makedirs(output_test_normal, exist_ok=True) # Fungsi untuk membagi data def split_data(folder, output_train, output_test, test_size=0.4): # Mengambil semua file gambar dalam folder images = [f for f in os.listdir(folder) if os.path.isfile(os.path.join(folder, f))] # Membagi data menjadi train dan test train_images, test_images = train_test_split(images, test_size=test_size, random_state=42) # Menyalin file gambar ke folder train for image in train_images: shutil.copy(os.path.join(folder, image), os.path.join(output_train, image)) # Menyalin file gambar ke folder test for image in test_images: shutil.copy(os.path.join(folder, image), os.path.join(output_test, image)) # Membagi data untuk folder DR split_data(folder_DR, output_train_DR, output_test_DR) # Membagi data untuk folder normal split_data(folder_normal, output_train_normal, output_test_normal) </pre>
Split Data Train dan Test 60:40	

<p>Split Data Train dan Test 50:50</p>	<pre> import os import shutil from sklearn.model_selection import train_test_split # Path ke folder dataset folder_DR = 'revisiBukJuni/DR' folder_normal = 'revisiBukJuni/Normal' # Path ke folder output output_train_DR = 'revisiBukJuni/data 50 50/train/DR' output_train_normal = 'revisiBukJuni/data 50 50/train/normal' output_test_DR = 'revisiBukJuni/data 50 50/test/DR' output_test_normal = 'revisiBukJuni/data 50 50/test/normal' # Membuat folder output jika belum ada os.makedirs(output_train_DR, exist_ok=True) os.makedirs(output_train_normal, exist_ok=True) os.makedirs(output_test_DR, exist_ok=True) os.makedirs(output_test_normal, exist_ok=True) # Fungsi untuk membagi data def split_data(folder, output_train, output_test, test_size=0.5): # Mengambil semua file gambar dalam folder images = [f for f in os.listdir(folder) if os.path.isfile(os.path.join(folder, f))] # Membagi data menjadi train dan test train_images, test_images = train_test_split(images, test_size=test_size, random_state=42) # Menyalin file gambar ke folder train for image in train_images: shutil.copy(os.path.join(folder, image), os.path.join(output_train, image)) # Menyalin file gambar ke folder test for image in test_images: shutil.copy(os.path.join(folder, image), os.path.join(output_test, image)) # Membagi data untuk folder DR split_data(folder_DR, output_train_DR, output_test_DR) # Membagi data untuk folder normal split_data(folder_normal, output_train_normal, output_test_normal) </pre>
---	---

<p>Library CNN</p>	<pre>[] from tensorflow.keras.models import Sequential, Model from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization, GlobalAveragePooling2D from tensorflow.keras.optimizers import Adam from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array from tensorflow.keras.applications import Xception, NASNetLarge, InceptionResNetV2, ResNet152V2, imagenet_utils from tensorflow.keras.regularizers import l2 from tensorflow.keras.initializers import GlorotUniform from tensorflow.keras.utils import to_categorical import matplotlib.pyplot as plt import numpy as np import os import pandas as pd from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score, accuracy_score, f1_score from keras.applications.efficientnet_v2 import EfficientNetV2L from datetime import datetime import keras</pre>
---------------------------	---

Normalisasi dataset 80:20	<pre>[] # Path ke direktori data train dan validasi train_path = '/content/drive/MyDrive/Dataset/data_80_20/train' test_path = '/content/drive/MyDrive/Dataset/data_80_20/test' [] num_train_samples = 2000 num_test_samples = 400 train_batch_size = 32 test_batch_size = 32 image_size = 224 [] train_steps = np.ceil(num_train_samples / train_batch_size) test_steps = np.ceil(num_test_samples / test_batch_size) datagen = ImageDataGenerator(rescale=1./255) train_batches = datagen.flow_from_directory(train_path, target_size=(image_size, image_size), batch_size=train_batch_size, classes=['DR', 'normal']) test_batches = datagen.flow_from_directory(test_path, target_size=(image_size, image_size), batch_size=test_batch_size, classes=['DR', 'normal'], shuffle=False)</pre>
Model CNN LR.0.001 Dataset 80:20	<p>▼ CNN => LR.0001 EPOCH 100</p> <pre>[] # Define the CNN model model_cnn = Sequential([Conv2D(128, (3, 3), activation='relu', input_shape=(224, 224, 3)), MaxPooling2D((2, 2)), Conv2D(64, (3, 3), activation='relu'), MaxPooling2D((2, 2)), Conv2D(32, (3, 3), activation='relu'), MaxPooling2D((2, 2)), Flatten(), Dense(128, activation='relu', kernel_regularizer=l2(0.01)), Dropout(0.5), Dense(64, activation='relu', kernel_regularizer=l2(0.01)), Dropout(0.5), Dense(2, activation='softmax', kernel_regularizer=l2(0.01))]) # Compile the model model_cnn.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy']) # Fit the model with early stopping history_cnn = model_cnn.fit(train_batches, validation_data=test_batches, epochs=100, steps_per_epoch=len(train_batches), validation_steps=len(test_batches)) # Simpan model ke dalam file HDFS os.makedirs('/content/drive/MyDrive/Dataset/hasil/LR 0.001 EPOCH 100', exist_ok=True) model_cnn.save('/content/drive/MyDrive/Dataset/hasil/LR 0.001 EPOCH 100/CNN_model.h5')</pre>

```

[] # Convert the history to a DataFrame
history_df_cnn = pd.DataFrame(history_cnn.history)

# Define the CSV file path
csv_folder_path_cnn = '/content/drive/MyDrive/Dataset/hasil/LR 0.001 EPOCH 100/History' # Update with your desired folder path
os.makedirs(csv_folder_path_cnn, exist_ok=True) # Ensure the folder exists

# rata-rata accuracy dari semua epoch yang dijalankan
average_accuracy_cnn = history_df_cnn['val_accuracy'].mean()

# Add a column for epoch times
history_df_cnn['epoch_time'] = (pd.to_datetime(datetime.utcnow()) - pd.to_datetime(datetime.utcnow())).total_seconds()

# Generate nama file CSV dengan informasi waktu, jumlah epoch, dan durasi model dijalankan
current_time_cnn = datetime.now().strftime('%.Y-%m-%d %H%M%S')
epoch_count_cnn = len(history_cnn.history['loss'])
duration_cnn = history_df_cnn['epoch_time'].sum()

csv_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_CNN_duration{int(duration_cnn)}s_training_history.csv'

#path file CSV disimpan
csv_folder_path_cnn = '/content/drive/MyDrive/Dataset/hasil/LR 0.001 EPOCH 100/History' # Ganti dengan path folder yang diinginkan
csv_filepath_cnn = os.path.join(csv_folder_path_cnn, csv_filename_cnn)

# Save the DataFrame to CSV
history_df_cnn.to_csv(csv_filepath_cnn, index=False)

❸ import os
import matplotlib.pyplot as plt
import numpy as np

# Define the folder path for saving plots
plot_folder_path_cnn = '/content/drive/MyDrive/Dataset/hasil/LR 0.001 EPOCH 100/Grafik'

# Create the directory if it doesn't exist
os.makedirs(plot_folder_path_cnn, exist_ok=True)

# Create a figure with specified size
plt.figure(figsize=(10, 5))

# Plot Training Loss
plt.subplot(1, 2, 1)
plt.plot(history_df_cnn['loss'], label='Training Loss')
plt.plot(history_df_cnn['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafik Loss')
plt.legend()
plt.grid(True)

# Plot Validation Accuracy
plt.subplot(1, 2, 2)
plt.plot(history_df_cnn['accuracy'], label='Training Accuracy')
plt.plot(history_df_cnn['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Grafik Accuracy')
plt.legend()
plt.grid(True)

# Adjust layout
plt.tight_layout()

# Save the combined plot
combined_plot_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_combined_plot.png'
combined_plot_filepath_cnn = os.path.join(plot_folder_path_cnn, combined_plot_filename_cnn)
plt.savefig(combined_plot_filepath_cnn)

# Show the plot
plt.show()

```

```

[ ] # Evaluate the model
    test_batches.reset()
    y_pred_cnn = model.cnn.predict(test_batches, steps=len(test_batches), verbose=1)
    y_pred_classes_cnn = np.argmax(y_pred_cnn, axis=1)
    y_true_cnn = test_batches.classes

    # Confusion Matrix and Classification Report
    conf_matrix_cnn = confusion_matrix(y_true_cnn, y_pred_classes_cnn)
    class_labels = list(train_batches.class_indices.keys())

    print("Confusion Matrix:")
    print(conf_matrix_cnn)

    print("Classification Report:")
    class_report_cnn = classification_report(y_true_cnn, y_pred_classes_cnn, target_names=class_labels)
    print(class_report_cnn)

    # Specificity, Sensitivity, and AUC
    tn, fp, fn, tp = conf_matrix_cnn.ravel()
    specificity = tn / (tn + fp)
    sensitivity = tp / (tp + fn)
    auc = roc_auc_score(y_true_cnn, y_pred_cnn[:, 1])

    print(f"Sensitivity: {sensitivity}")
    print(f"Specificity: {specificity}")
    print(f"AUC: {auc}")

    # Calculate F1-score
    f1score = f1_score(y_true_cnn, y_pred_classes_cnn, average='weighted')
    print(f"F1-score: {f1score}")

    print(f"Accuracy: {average_accuracy_cnn}")

[ ] import os
import numpy as np

# Define the folder path for saving TXT file
txt_folder_path_cnn = '/content/drive/MyDrive/Dataset/hasil/LR_0.001 EPOCH 100/accuracy'

# Create the directory if it doesn't exist
os.makedirs(txt_folder_path_cnn, exist_ok=True)

# Define the filename
txt_filename_cnn = f"(current_time_cnn)_epochs{epoch_count_cnn}_duration{int(duration_cnn)}confusion matrix.txt"
txt_filepath_cnn = os.path.join(txt_folder_path_cnn, txt_filename_cnn)

# Save the information to TXT file
with open(txt_filepath_cnn, 'w') as txt_file_cnn:
    # Write Confusion Matrix
    txt_file_cnn.write("Confusion Matrix:\n")
    txt_file_cnn.write(np.array2string(conf_matrix_cnn, separator=', ') + '\n\n')

    # Write Classification Report
    txt_file_cnn.write("Classification Report:\n")
    txt_file_cnn.write(class_report_cnn + '\n\n')

    # Write Specificity, Sensitivity, AUC, and Accuracy
    txt_file_cnn.write(f"Specificity: {specificity}\n")
    txt_file_cnn.write(f"Sensitivity: {sensitivity}\n")
    txt_file_cnn.write(f"AUC: {auc}\n")
    txt_file_cnn.write(f"F1-score: {f1score}\n")
    txt_file_cnn.write(f"Accuracy: {average_accuracy_cnn}\n")

```

Model CNN LR.0.0001 Dataset 80:20

```
✗ CNN => LR 0.0001 EPOCH 100

[ ] # Define the CNN model
model_cnn = Sequential([
    Conv2D(128, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(2, activation='softmax', kernel_regularizer=l2(0.01))
])

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

# Fit the model with early stopping
history_cnn = model_cnn.fit(train_batches,
                             validation_data=test_batches,
                             epochs=100,
                             steps_per_epoch=len(train_batches),
                             validation_steps=len(test_batches))

# Simpan model ke dalam file HDF5
os.makedirs('/content/drive/MyDrive/Dataset/hasil/LR 0.0001 EPOCH 100', exist_ok=True)
model_cnn.save('/content/drive/MyDrive/Dataset/hasil/LR 0.0001 EPOCH 100/CNN_model.h5')

[ ] # Convert the history to a DataFrame
history_df_cnn = pd.DataFrame(history_cnn.history)

# Define the CSV file path
csv_folder_path_cnn = '/content/drive/MyDrive/Dataset/hasil/LR 0.0001 EPOCH 100/History' # Update with your desired folder path
os.makedirs(csv_folder_path_cnn, exist_ok=True) # Ensure the folder exists

# rata-rata accuracy dari semua epoch yang dijalankan
average_accuracy_cnn = history_df_cnn['val_accuracy'].mean()

# Add a column for epoch times
history_df_cnn['epoch_time'] = (pd.to_datetime(datetime.utcnow()) - pd.to_datetime(datetime.utcnow())).total_seconds()

# Generate nama file CSV dengan informasi waktu, jumlah epoch, dan durasi model dijalankan
current_time_cnn = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
epoch_count_cnn = len(history_cnn.history['loss'])
duration_cnn = history_df_cnn['epoch_time'].sum()

csv_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_CNN_duration{int(duration_cnn)}s_training_history.csv'

#path file CSV disimpan
csv_folder_path_cnn = '/content/drive/MyDrive/Dataset/hasil/LR 0.0001 EPOCH 100/History' # Ganti dengan path folder yang diinginkan
csv_filepath_cnn = os.path.join(csv_folder_path_cnn, csv_filename_cnn)

# Save the DataFrame to CSV
history_df_cnn.to_csv(csv_filepath_cnn, index=False)
```

```

[ ] import os
import matplotlib.pyplot as plt
import numpy as np

# Define the folder path for saving plots
plot_folder_path_cnn = '/content/drive/MyDrive/Dataset/hasil/LR 0.0001 EPOCH 100/Grafik'

# Create the directory if it doesn't exist
os.makedirs(plot_folder_path_cnn, exist_ok=True)

# Create a figure with specified size
plt.figure(figsize=(10, 5))

# Plot Training Loss
plt.subplot(1, 2, 1)
plt.plot(history_df_cnn['loss'], label='Training Loss')
plt.plot(history_df_cnn['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafik Loss')
plt.legend()
plt.grid(True)

# Plot Validation Accuracy
plt.subplot(1, 2, 2)
plt.plot(history_df_cnn['accuracy'], label='Training Accuracy')
plt.plot(history_df_cnn['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Grafik Accuracy')
plt.legend()
plt.grid(True)

# Adjust layout
plt.tight_layout()

# Save the combined plot
combined_plot_filename_cnn = f"{current_time_cnn}_epochs{epoch_count_cnn}_combined_plot.png"
combined_plot_filepath_cnn = os.path.join(plot_folder_path_cnn, combined_plot_filename_cnn)
plt.savefig(combined_plot_filepath_cnn)

# Show the plot
plt.show()

[ ] # Evaluate the model
test_batches.reset_()
y_pred_cnn = model_cnn.predict(test_batches, steps=len(test_batches), verbose=1)
y_pred_classes_cnn = np.argmax(y_pred_cnn, axis=1)
y_true_cnn = test_batches.classes

# Confusion Matrix and Classification Report
conf_matrix_cnn = confusion_matrix(y_true_cnn, y_pred_classes_cnn)
class_labels = list(train_batches.class_indices.keys())

print("Confusion Matrix:")
print(conf_matrix_cnn)

print("Classification Report:")
class_report_cnn = classification_report(y_true_cnn, y_pred_classes_cnn, target_names=class_labels)
print(class_report_cnn)

# Specificity, Sensitivity, and AUC
tn, fp, fn, tp = conf_matrix_cnn.ravel()
specificity = tn / (tn + fp)
sensitivity = tp / (tp + fn)
auc = roc_auc_score(y_true_cnn, y_pred_cnn[:, 1])

print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"AUC: {auc}")

# Calculate F1-score
f1score = f1_score(y_true_cnn, y_pred_classes_cnn, average='weighted')
print(f"F1-score: {f1score}")

print(f"Accuracy: {average_accuracy_cnn}")

```

	<pre>[] import os import numpy as np # Define the folder path for saving TXT file txt_folder_path_cnn = '/content/drive/MyDrive/Dataset/hasil/LR_0.0001 EPOCH 100/Accuracy' # Create the directory if it doesn't exist os.makedirs(txt_folder_path_cnn, exist_ok=True) # Define the filename txt_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_duration{int(duration_cnn)}confusion matrix.txt' txt_filepath_cnn = os.path.join(txt_folder_path_cnn, txt_filename_cnn) # Save the information to TXT file with open(txt_filepath_cnn, 'w') as txt_file_cnn: # Write Confusion Matrix txt_file_cnn.write("Confusion Matrix:\n") txt_file_cnn.write(np.array2string(conf_matrix_cnn, separator=', ') + '\n\n') # Write Classification Report txt_file_cnn.write("Classification Report:\n") txt_file_cnn.write(class_report_cnn + '\n\n') # Write Specificity, Sensitivity, AUC, and Accuracy txt_file_cnn.write(f"Specificity: {specificity}\n") txt_file_cnn.write(f"Sensitivity: {sensitivity}\n") txt_file_cnn.write(f"AUC: {auc}\n") txt_file_cnn.write(f"F1-score: {f1score}\n") txt_file_cnn.write(f"Accuracy: {average_accuracy_cnn}\n")</pre>
Normalisasi dataset 70:30	<pre>[] # Path ke direktori data train dan validasi train_path = '/content/drive/MyDrive/dataset/data_70_30/train' test_path = '/content/drive/MyDrive/dataset/data_70_30/test' [] num_train_samples = 2000 num_test_samples = 600 train_batch_size = 32 test_batch_size = 32 [] train_steps = np.ceil(num_train_samples / train_batch_size) test_steps = np.ceil(num_test_samples / test_batch_size) datagen = ImageDataGenerator(rescale=1./255) train_batches = datagen.flow_from_directory(train_path, target_size=(224, 224), batch_size=train_batch_size, classes=['DR', 'normal']) test_batches = datagen.flow_from_directory(test_path, target_size=(224, 224), batch_size=test_batch_size, classes=['DR', 'normal'], shuffle=False)</pre>

Model CNN

LR.0.001

Dataset

70:30

✓ CNN => LR 0.001 EPOCH 100

```
[ ] # Define the CNN model
model_cnn = Sequential([
    Conv2D(128, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(2, activation='softmax', kernel_regularizer=l2(0.01))
])

# Compile the model
model_cnn.compile(optimizer=Adam(learning_rate=0.001),
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])

# Assuming train_batches and test_batches are already defined
history_cnn = model_cnn.fit(train_batches,
                            validation_data=test_batches,
                            epochs=100)

# Save the model to an HDF5 file
os.makedirs('/content/drive/MyDrive/dataset/hasil 70 30/LR 0.001 EPOCH 100', exist_ok=True)
model_cnn.save('/content/drive/MyDrive/dataset/hasil 70 30/LR 0.001 EPOCH 100/CNN_model_improved.h5')

[ ] # Convert the history to a DataFrame
history_df_cnn = pd.DataFrame(history_cnn.history)

# Define the CSV file path
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil 70 30/LR 0.001 EPOCH 100/History' # Update with your desired folder path
os.makedirs(csv_folder_path_cnn, exist_ok=True) # Ensure the folder exists

# rata-rata accuracy dari semua epoch yang dijalankan
average_accuracy_cnn = history_df_cnn['val_accuracy'].mean()

# Add a column for epoch times
history_df_cnn['epoch_time'] = (pd.to_datetime(datetime.utcnow()) - pd.to_datetime(datetime.utcnow())).total_seconds()

# Generate name file CSV dengan informasi waktu, jumlah epoch, dan durasi model dijalankan
current_time_cnn = datetime.now().strftime("%Y-%m-%d %H%M%S")
epoch_count_cnn = len(history_cnn.history['loss'])
duration_cnn = history_df_cnn['epoch_time'].sum()

csv_filename_cnn = f'{current_time_cnn}_epochs_{epoch_count_cnn}_CNN_duration_{int(duration_cnn)}s_training_history.csv'

#path file CSV disimpan
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil 70 30/LR 0.001 EPOCH 100/History' # Ganti dengan path folder yang diinginkan
csv_filepath_cnn = os.path.join(csv_folder_path_cnn, csv_filename_cnn)

# Save the DataFrame to CSV
history_df_cnn.to_csv(csv_filepath_cnn, index=False)
```

```

[ ] import os
import matplotlib.pyplot as plt
import numpy as np

# Define the folder path for saving plots
plot_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_70_30/LR 0.001 EPOCH 100/Grafik'

# Create the directory if it doesn't exist
os.makedirs(plot_folder_path_cnn, exist_ok=True)

# Create a figure with specified size
plt.figure(figsize=(10, 5))

# Plot Training Loss
plt.subplot(1, 2, 1)
plt.plot(history_df_cnn['loss'], label='Training Loss')
plt.plot(history_df_cnn['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafik Loss')
plt.legend()
plt.grid(True)

# Plot Validation Accuracy
plt.subplot(1, 2, 2)
plt.plot(history_df_cnn['accuracy'], label='Training Accuracy')
plt.plot(history_df_cnn['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Grafik Accuracy')
plt.legend()
plt.grid(True)

# Adjust layout
plt.tight_layout()

# Save the combined plot
combined_plot_filename_cnn = f"{current_time_cnn}_epochs{epoch_count_cnn}_combined_plot.png"
combined_plot_filepath_cnn = os.path.join(plot_folder_path_cnn, combined_plot_filename_cnn)
plt.savefig(combined_plot_filepath_cnn)

# Show the plot
plt.show()

[ ] # Evaluate the model
test_batches.reset()
y_pred_cnn = model_cnn.predict(test_batches, steps=len(test_batches), verbose=1)
y_pred_classes_cnn = np.argmax(y_pred_cnn, axis=1)
y_true_cnn = test_batches.classes

# Confusion Matrix and Classification Report
conf_matrix_cnn = confusion_matrix(y_true_cnn, y_pred_classes_cnn)
class_labels = list(train_batches.class_indices.keys())

print("Confusion Matrix:")
print(conf_matrix_cnn)

print("Classification Report:")
class_report_cnn = classification_report(y_true_cnn, y_pred_classes_cnn, target_names=class_labels)
print(class_report_cnn)

# Specificity, Sensitivity, and AUC
tn, fp, fn, tp = conf_matrix_cnn.ravel()
specificity = tn / (tn + fp)
sensitivity = tp / (tp + fn)
auc = roc_auc_score(y_true_cnn, y_pred_cnn[:, 1])

print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"AUC: {auc}")

# Calculate F1-score
fiscore = f1_score(y_true_cnn, y_pred_classes_cnn, average='weighted')
print(f"F1-score: {fiscore}")

print(f"Accuracy: {average_accuracy_cnn}")

```

	<pre>[] import os import numpy as np # Define the folder path for saving TXT file txt_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_70_30/LR 0.001 EPOCH 100/Accuracy' # Create the directory if it doesn't exist os.makedirs(txt_folder_path_cnn, exist_ok=True) # Define the filename txt_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_duration{int(duration_cnn)}confusion matrix.txt' txt_filepath_cnn = os.path.join(txt_folder_path_cnn, txt_filename_cnn) # Save the information to TXT file with open(txt_filepath_cnn, 'w') as txt_file_cnn: # Write Confusion Matrix txt_file_cnn.write("Confusion Matrix:\n") txt_file_cnn.write(np.array2string(conf_matrix_cnn, separator=', ') + '\n\n') # Write Classification Report txt_file_cnn.write("Classification Report:\n") txt_file_cnn.write(class_report_cnn + '\n\n') # Write Specificity, Sensitivity, AUC, and Accuracy txt_file_cnn.write(f"Specificity: {specificity}\n") txt_file_cnn.write(f"Sensitivity: {sensitivity}\n") txt_file_cnn.write(f"AUC: {auc}\n") txt_file_cnn.write(f"F1-score: {f1score}\n") txt_file_cnn.write(f"Accuracy: {average_accuracy_cnn}\n")</pre>
Model CNN LR.0.0001 Dataset 70:30	<pre>[] # Define the CNN model model_cnn = Sequential([Conv2D(128, (3, 3), activation='relu', input_shape=(224, 224, 3)), MaxPooling2D((2, 2)), Conv2D(64, (3, 3), activation='relu'), MaxPooling2D((2, 2)), Conv2D(32, (3, 3), activation='relu'), MaxPooling2D((2, 2)), Flatten(), Dense(128, activation='relu', kernel_regularizer=l2(0.01)), Dropout(0.5), Dense(64, activation='relu', kernel_regularizer=l2(0.01)), Dropout(0.5), Dense(2, activation='softmax', kernel_regularizer=l2(0.01))]) # Compile the model model_cnn.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy']) history_cnn = model_cnn.fit(train_batches, validation_data=test_batches, epochs=100) # Simpan model ke dalam file HDF5 os.makedirs('/content/drive/MyDrive/dataset/hasil_70_30/LR 0.0001 EPOCH 100', exist_ok=True) model_cnn.save('/content/drive/MyDrive/dataset/hasil_70_30/LR 0.0001 EPOCH 100/CNN_model.h5')</pre>

```

[ ] # Convert the history to a DataFrame
history_df_cnn = pd.DataFrame(history_cnn.history)

# Define the CSV file path
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_70_30/LR 0.0001 EPOCH 100/History' # Update with your desired folder path
os.makedirs(csv_folder_path_cnn, exist_ok=True) # Ensure the folder exists

# rata-rata accuracy dari semua epoch yang dijalankan
average_accuracy_cnn = history_df_cnn['val_accuracy'].mean()

# Add a column for epoch times
history_df_cnn['epoch_time'] = (pd.to_datetime(datetime.utcnow()) - pd.to_datetime(datetime.utcnow()))/total_seconds()

# Generate nama file CSV dengan informasi waktu, jumlah epoch, dan durasi model dijalankan
current_time_cnn = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
epoch_count_cnn = len(history_cnn.history['loss'])
duration_cnn = history_df_cnn['epoch_time'].sum()

csv_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_Duration{int(duration_cnn)}s_training_history.csv'

#path file CSV disimpan
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_70_30/LR 0.0001 EPOCH 100/History' # Ganti dengan path folder yang diinginkan
csv_filepath_cnn = os.path.join(csv_folder_path_cnn, csv_filename_cnn)

# Save the DataFrame to CSV
history_df_cnn.to_csv(csv_filepath_cnn, index=False)

[ ] import os
import matplotlib.pyplot as plt
import numpy as np

# Define the folder path for saving plots
plot_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_70_30/LR 0.0001 EPOCH 100/Grafik'

# Create the directory if it doesn't exist
os.makedirs(plot_folder_path_cnn, exist_ok=True)

# Create a figure with specified size
plt.figure(figsize=(10, 5))

# Plot Training Loss
plt.subplot(1, 2, 1)
plt.plot(history_df_cnn['loss'], label='Training Loss')
plt.plot(history_df_cnn['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafik Loss')
plt.legend()
plt.grid(True)

# Plot Validation Accuracy
plt.subplot(1, 2, 2)
plt.plot(history_df_cnn['accuracy'], label='Training Accuracy')
plt.plot(history_df_cnn['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Grafik Accuracy')
plt.legend()
plt.grid(True)

# Adjust layout
plt.tight_layout()

# Save the combined plot
combined_plot_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_combined_plot.png'
combined_plot_filepath_cnn = os.path.join(plot_folder_path_cnn, combined_plot_filename_cnn)
plt.savefig(combined_plot_filepath_cnn)

# Show the plot
plt.show()

```

```

[ ] test_batches.reset() # Reset the generator to the beginning
y_pred_cnn = model_cnn.predict(test_batches, steps=len(test_batches), verbose=1)
y_pred_classes_cnn = np.argmax(y_pred_cnn, axis=1)
y_true_cnn = test_batches.classes

# Get the accuracy from the history
test_accuracy_cnn = history_cnn.history['val_accuracy']

from sklearn.metrics import f1_score

# Confusion Matrix
conf_matrix_cnn = confusion_matrix(y_true_cnn, y_pred_classes_cnn)
print("Confusion Matrix:")
print(conf_matrix_cnn)

# Classification Report
class_report_cnn = classification_report(y_true_cnn, y_pred_classes_cnn)
print("Classification Report:")
print(class_report_cnn)

# Specificity, Sensitivity, and AUC
tn, fp, fn, tp = conf_matrix_cnn.ravel()
specificity = tn / (tn + fp)
sensitivity = tp / (tp + fn)
auc = roc_auc_score(y_true_cnn, y_pred_cnn[:, 1], multi_class='ovr') # Adjust AUC calculation for multiple classes

print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"AUC: {auc}")

# Calculate F1-score
fiscore = f1_score(y_true_cnn, y_pred_classes_cnn, average='weighted')
print(f"F1-score: {fiscore}")

print(f"Accuracy: {accuracy_cnn}")

[ ] import os
import numpy as np

# Define the folder path for saving TXT file
txt_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_70_30/LR_0.0001_EPOCH_100/Accuracy'

# Create the directory if it doesn't exist
os.makedirs(txt_folder_path_cnn, exist_ok=True)

# Define the filename
txt_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_duration{int(duration_cnn)}confusion matrix.txt'
txt_filepath_cnn = os.path.join(txt_folder_path_cnn, txt_filename_cnn)

# Save the information to TXT file
with open(txt_filepath_cnn, 'w') as txt_file_cnn:
    # Write Confusion Matrix
    txt_file_cnn.write("Confusion Matrix:\n")
    txt_file_cnn.write(np.array2string(conf_matrix_cnn, separator=', ') + '\n\n')

    # Write Classification Report
    txt_file_cnn.write("Classification Report:\n")
    txt_file_cnn.write(class_report_cnn + '\n\n')

    # Write Specificity, Sensitivity, AUC, and Accuracy
    txt_file_cnn.write(f"Specificity: {specificity}\n")
    txt_file_cnn.write(f"Sensitivity: {sensitivity}\n")
    txt_file_cnn.write(f"AUC: {auc}\n")
    txt_file_cnn.write(f"F1-score: {fiscore}\n")
    txt_file_cnn.write(f"Accuracy: {accuracy_cnn}\n")

```

Normalisasi dataset 60:40	<pre>[] # Path ke direktori data train dan validasi train_path = '/content/drive/MyDrive/dataset/data_60_40/train' test_path = '/content/drive/MyDrive/dataset/data_60_40/test' [] num_train_samples = 2000 num_test_samples = 800 train_batch_size = 32 test_batch_size = 32 [] train_steps = np.ceil(num_train_samples / train_batch_size) test_steps = np.ceil(num_test_samples / test_batch_size) datagen = ImageDataGenerator(rescale=1./255) train_batches = datagen.flow_from_directory(train_path, target_size=(224, 224), batch_size=train_batch_size, classes=['DR', 'normal']) test_batches = datagen.flow_from_directory(test_path, target_size=(224, 224), batch_size=test_batch_size, classes=['DR', 'normal'], shuffle=False)</pre>
Model CNN LR.0.001 Dataset 60:40	<p>▼ CNN => LR 0.001 EPOCH 100</p> <pre>[] # Define the CNN model model_cnn = Sequential([Conv2D(128, (3, 3), activation='relu', input_shape=(224, 224, 3)), MaxPooling2D((2, 2)), Conv2D(64, (3, 3), activation='relu'), MaxPooling2D((2, 2)), Conv2D(32, (3, 3), activation='relu'), MaxPooling2D((2, 2)), Flatten(), Dense(128, activation='relu', kernel_regularizer=l2(0.01)), Dropout(0.5), Dense(64, activation='relu', kernel_regularizer=l2(0.01)), Dropout(0.5), Dense(2, activation='softmax', kernel_regularizer=l2(0.01))]) # Compile the model model_cnn.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy']) history_cnn = model_cnn.fit(train_batches, validation_data=test_batches, epochs=100) # Simpan model ke dalam file HDF5 os.makedirs('/content/drive/MyDrive/dataset/hasil_60_40/LR 0.001 EPOCH 100', exist_ok=True) model_cnn.save('/content/drive/MyDrive/dataset/hasil_60_40/LR 0.001 EPOCH 100/CNN_model.h5')</pre>

```

[ ] # Convert the history to a DataFrame
history_df_cnn = pd.DataFrame(history_cnn.history)

# Define the CSV file path
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil 60 40/LR 0.001 EPOCH 100/History'
os.makedirs(csv_folder_path_cnn, exist_ok=True) # Ensure the folder exists

# rata-rata accuracy dari semua epoch yang dijalankan
average_accuracy_cnn = history_df_cnn['val_accuracy'].mean()

# Add a column for epoch times
history_df_cnn['epoch_time'] = (pd.to_datetime(datetime.utcnow()) - pd.to_datetime(datetime.utcnow())).total_seconds()

# Generate nama file CSV dengan informasi waktu, jumlah epoch, dan durasi model dijalankan
current_time_cnn = datetime.now().strftime("%Y-%m-%d %H%M%S")
epoch_count_cnn = len(history_cnn.history['loss'])
duration_cnn = history_df_cnn['epoch_time'].sum()

csv_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_CNN_duration{int(duration_cnn)}s_training_history.csv'

#path file CSV disimpan
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil 60 40/LR 0.001 EPOCH 100/History'
csv_filepath_cnn = os.path.join(csv_folder_path_cnn, csv_filename_cnn)

# Save the DataFrame to CSV
history_df_cnn.to_csv(csv_filepath_cnn, index=False)

[ ] import os
import matplotlib.pyplot as plt
import numpy as np

# Define the folder path for saving plots
plot_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil 60 40/LR 0.001 EPOCH 100/Grafik'

# Create the directory if it doesn't exist
os.makedirs(plot_folder_path_cnn, exist_ok=True)

# Create a figure with specified size
plt.figure(figsize=(10, 5))

# Plot Training Loss
plt.subplot(1, 2, 1)
plt.plot(history_df_cnn['loss'], label='Training Loss')
plt.plot(history_df_cnn['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafik Loss')
plt.legend()
plt.grid(True)

# Plot Validation Accuracy
plt.subplot(1, 2, 2)
plt.plot(history_df_cnn['accuracy'], label='Training Accuracy')
plt.plot(history_df_cnn['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Grafik Accuracy')
plt.legend()
plt.grid(True)

# Adjust layout
plt.tight_layout()

# Save the combined plot
combined_plot_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_combined_plot.png'
combined_plot_filepath_cnn = os.path.join(plot_folder_path_cnn, combined_plot_filename_cnn)
plt.savefig(combined_plot_filepath_cnn)

# Show the plot
plt.show()

```

```

[ ] # Evaluate the model
test_batches.reset()
y_pred_cnn = model_cnn.predict(test_batches, steps=len(test_batches), verbose=1)
y_pred_classes_cnn = np.argmax(y_pred_cnn, axis=1)
y_true_cnn = test_batches.classes

# Confusion Matrix and Classification Report
conf_matrix_cnn = confusion_matrix(y_true_cnn, y_pred_classes_cnn)
class_labels = list(train_batches.class_indices.keys())

print("Confusion Matrix:")
print(conf_matrix_cnn)

print("Classification Report:")
class_report_cnn = classification_report(y_true_cnn, y_pred_classes_cnn, target_names=class_labels)
print(class_report_cnn)

# Specificity, Sensitivity, and AUC
tn, fp, fn, tp = conf_matrix_cnn.ravel()
specificity = tn / (tn + fp)
sensitivity = tp / (tp + fn)
auc = roc_auc_score(y_true_cnn, y_pred_cnn[:, 1])

print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"AUC: {auc}")

# Calculate F1-score
f1score = f1_score(y_true_cnn, y_pred_classes_cnn, average='weighted')
print(f"F1-score: {f1score}")

print(f"Accuracy: {average_accuracy_cnn}")

[ ] import os
import numpy as np

# Define the folder path for saving TXT file
txt_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_60_40/LR_0.001_EPOCH_100/Accuracy'

# Create the directory if it doesn't exist
os.makedirs(txt_folder_path_cnn, exist_ok=True)

# Define the filename
txt_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_duration{int(duration_cnn)}confusion matrix.txt'
txt_filepath_cnn = os.path.join(txt_folder_path_cnn, txt_filename_cnn)

# Save the information to TXT file
with open(txt_filepath_cnn, 'w') as txt_file_cnn:
    # Write Confusion Matrix
    txt_file_cnn.write("Confusion Matrix:\n")
    txt_file_cnn.write(np.array2string(conf_matrix_cnn, separator=', ') + '\n\n')

    # Write Classification Report
    txt_file_cnn.write("Classification Report:\n")
    txt_file_cnn.write(class_report_cnn + '\n\n')

    # Write Specificity, Sensitivity, AUC, and Accuracy
    txt_file_cnn.write(f"Specificity: {specificity}\n")
    txt_file_cnn.write(f"Sensitivity: {sensitivity}\n")
    txt_file_cnn.write(f"AUC: {auc}\n")
    txt_file_cnn.write(f"F1-score: {f1score}\n")
    txt_file_cnn.write(f"Accuracy: {average_accuracy_cnn}\n")

```

Model CNN LR.0.0001 Dataset 60:40

```
✗ CNN => LR 0.0001 EPOCH 100

[ ] # Define the CNN model
model_cnn = Sequential([
    Conv2D(128, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(2, activation='softmax', kernel_regularizer=l2(0.01))
])

# Compile the model
model_cnn.compile(optimizer=Adam(learning_rate=0.001),
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])

history_cnn = model_cnn.fit(train_batches,
                            validation_data=test_batches,
                            epochs=100)

# Simpan model ke dalam file HDF5
os.makedirs('/content/drive/MyDrive/dataset/hasil_60_40/LR 0.0001 EPOCH 100', exist_ok=True)
model_cnn.save('/content/drive/MyDrive/dataset/hasil_60_40/LR 0.0001 EPOCH 100/CNN_model.h5')

[ ] # Convert the history to a DataFrame
history_df_cnn = pd.DataFrame(history_cnn.history)

# Define the CSV file path
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_60_40/LR 0.0001 EPOCH 100/History' # Update with your desired folder path
os.makedirs(csv_folder_path_cnn, exist_ok=True) # Ensure the folder exists

# Rata-rata accuracy dari semua epoch yang dijalankan
average_accuracy_cnn = history_df_cnn['val_accuracy'].mean()

# Add a column for epoch times
history_df_cnn['epoch_time'] = (pd.to_datetime(datetime.utcnow()) - pd.to_datetime(datetime.utcnow())).total_seconds()

# Generate nama file CSV dengan informasi waktu, jumlah epoch, dan durasi model dijalankan
current_time_cnn = datetime.now().strftime("%Y-%m-%d %H%M%S")
epoch_count_cnn = len(history_cnn.history['loss'])
duration_cnn = history_df_cnn['epoch_time'].sum()

csv_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_CNN_duration{int(duration_cnn)}s_training_history.csv'

#path file CSV disimpan
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_60_40/LR 0.0001 EPOCH 100/History' # Ganti dengan path folder yang diinginkan
csv_filepath_cnn = os.path.join(csv_folder_path_cnn, csv_filename_cnn)

# Save the DataFrame to CSV
history_df_cnn.to_csv(csv_filepath_cnn, index=False)
```

```

[ ] import os
import matplotlib.pyplot as plt
import numpy as np

# Define the folder path for saving plots
plot_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_60_40/LR_0.0001 EPOCH 100/Grafik'

# Create the directory if it doesn't exist
os.makedirs(plot_folder_path_cnn, exist_ok=True)

# Create a figure with specified size
plt.figure(figsize=(10, 5))

# Plot Training Loss
plt.subplot(1, 2, 1)
plt.plot(history_df_cnn['loss'], label='Training Loss')
plt.plot(history_df_cnn['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafik Loss')
plt.legend()
plt.grid(True)

# Plot Validation Accuracy
plt.subplot(1, 2, 2)
plt.plot(history_df_cnn['accuracy'], label='Training Accuracy')
plt.plot(history_df_cnn['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Grafik Accuracy')
plt.legend()
plt.grid(True)

# Adjust layout
plt.tight_layout()

# Save the combined plot
combined_plot_filename_cnn = f"{current_time_cnn}_epochs{epoch_count_cnn}_combined_plot.png"
combined_plot_filepath_cnn = os.path.join(plot_folder_path_cnn, combined_plot_filename_cnn)
plt.savefig(combined_plot_filepath_cnn)

# Show the plot
plt.show()

[ ] # Evaluate the model
test_batches.reset()
y_pred_cnn = model_cnn.predict(test_batches, steps=len(test_batches), verbose=1)
y_pred_classes_cnn = np.argmax(y_pred_cnn, axis=1)
y_true_cnn = test_batches.classes

# Confusion Matrix and Classification Report
conf_matrix_cnn = confusion_matrix(y_true_cnn, y_pred_classes_cnn)
class_labels = list(train_batches.class_indices.keys())

print("Confusion Matrix:")
print(conf_matrix_cnn)

print("Classification Report:")
class_report_cnn = classification_report(y_true_cnn, y_pred_classes_cnn, target_names=class_labels)
print(class_report_cnn)

# Specificity, Sensitivity, and AUC
tn, fp, fn, tp = conf_matrix_cnn.ravel()
specificity = tn / (tn + fp)
sensitivity = tp / (tp + fn)
auc = roc_auc_score(y_true_cnn, y_pred_cnn[:, 1])

print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"AUC: {auc}")

# Calculate F1-score
f1score = f1_score(y_true_cnn, y_pred_classes_cnn, average='weighted')
print(f"F1-score: {f1score}")

print(f"Accuracy: {accuracy_cnn}")

```

	<pre>[] import os import numpy as np # Define the folder path for saving TXT file txt_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_60_40_0.0001_EPOCH 100/Accuracy' # Create the directory if it doesn't exist os.makedirs(txt_folder_path_cnn, exist_ok=True) # Define the filename txt_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_duration{int(duration_cnn)}confusion matrix.txt' txt_filepath_cnn = os.path.join(txt_folder_path_cnn, txt_filename_cnn) # Save the information to TXT file with open(txt_filepath_cnn, 'w') as txt_file_cnn: # Write Confusion Matrix txt_file_cnn.write("Confusion Matrix:\n") txt_file_cnn.write(np.array2string(conf_matrix_cnn, separator=', ') + '\n\n') # Write Classification Report txt_file_cnn.write("Classification Report:\n") txt_file_cnn.write(class_report_cnn + '\n\n') # Write Specificity, Sensitivity, AUC, and Accuracy txt_file_cnn.write(f"Specificity: {specificity}\n") txt_file_cnn.write(f"Sensitivity: {sensitivity}\n") txt_file_cnn.write(f"AUC: {auc}\n") txt_file_cnn.write(f"F1-score: {f1score}\n") txt_file_cnn.write(f"Accuracy: {average_accuracy_cnn}\n")</pre>
Normalisasi dataset 50:50	<pre>[] # Path ke direktori data train dan validasi train_path = '/content/drive/MyDrive/dataset/data_50_50/train' test_path = '/content/drive/MyDrive/dataset/data_50_50/test' [] num_train_samples = 1000 num_test_samples = 1000 train_batch_size = 32 test_batch_size = 32 [] train_steps = np.ceil(num_train_samples / train_batch_size) test_steps = np.ceil(num_test_samples / test_batch_size) datagen = ImageDataGenerator(rescale=1./255) train_batches = datagen.flow_from_directory(train_path, target_size=(224, 224), batch_size=train_batch_size, classes=['DR', 'normal']) test_batches = datagen.flow_from_directory(test_path, target_size=(224, 224), batch_size=test_batch_size, classes=['DR', 'normal'], shuffle=False)</pre>

Model CNN LR.0.001 Dataset 50:50

▼ CNN => LR 0.001 EPOCH 100

```
[ ] # Define the CNN model
model_cnn = Sequential([
    Conv2D(128, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(2, activation='softmax', kernel_regularizer=l2(0.01))
])

# Compile the model
model_cnn.compile(optimizer=Adam(learning_rate=0.001),
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])

# Assuming train_batches and test_batches are already defined
history_cnn = model_cnn.fit(train_batches,
                            validation_data=test_batches,
                            epochs=100)

# Save the model to an HDF5 file
os.makedirs('/content/drive/MyDrive/dataset/hasil_50_50/LR 0.001 EPOCH 100', exist_ok=True)
model_cnn.save('/content/drive/MyDrive/dataset/hasil_50_50/LR 0.001 EPOCH 100/CNN_model_improved.h5')

[ ] # Convert the history to a DataFrame
history_df_cnn = pd.DataFrame(history_cnn.history)

# Define the CSV file path
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_50_50/LR 0.001 EPOCH 100/History' # Update with your desired folder path
os.makedirs(csv_folder_path_cnn, exist_ok=True) # Ensure the folder exists

# rata-rata accuracy dari semua epoch yang dijalankan
average_accuracy_cnn = history_df_cnn['val_accuracy'].mean()

# Add a column for epoch times
history_df_cnn['epoch_time'] = (pd.to_datetime(datetime.utcnow()) - pd.to_datetime(datetime.utcnow()))/total_seconds()

# Generate nama file CSV dengan informasi waktu, jumlah epoch, dan durasi model dijalankan
current_time_cnn = datetime.now().strftime("%Y-%m-%d %H%M%S")
epoch_count_cnn = len(history_cnn.history['loss'])
duration_cnn = history_df_cnn['epoch_time'].sum()

csv_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_CNN_duration{int(duration_cnn)}s_training_history.csv'

#path file CSV disimpan
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_50_50/LR 0.001 EPOCH 100/History' # Ganti dengan path folder yang diinginkan
csv_filepath_cnn = os.path.join(csv_folder_path_cnn, csv_filename_cnn)

# Save the DataFrame to CSV
history_df_cnn.to_csv(csv_filepath_cnn, index=False)
```

```

[ ] import os
import matplotlib.pyplot as plt
import numpy as np

# Define the folder path for saving plots
plot_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_50_50/LR 0.001 EPOCH 100/Grafik'

# Create the directory if it doesn't exist
os.makedirs(plot_folder_path_cnn, exist_ok=True)

# Create a figure with specified size
plt.figure(figsize=(10, 5))

# Plot Training Loss
plt.subplot(1, 2, 1)
plt.plot(history_df_cnn['loss'], label='Training Loss')
plt.plot(history_df_cnn['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafik Loss')
plt.legend()
plt.grid(True)

# Plot Validation Accuracy
plt.subplot(1, 2, 2)
plt.plot(history_df_cnn['accuracy'], label='Training Accuracy')
plt.plot(history_df_cnn['val_accuracy'], label='Validation Accuracy')
plt.xlabel("Accuracy")
plt.ylabel('Accuracy')
plt.title('Grafik Accuracy')
plt.legend()
plt.grid(True)

# Adjust layout
plt.tight_layout()

# Save the combined plot
combined_plot_filename_cnn = f"{current_time_cnn}_epoch{epoch_count_cnn}_combined_plot.png"
combined_plot_filepath_cnn = os.path.join(plot_folder_path_cnn, combined_plot_filename_cnn)
plt.savefig(combined_plot_filepath_cnn)

# Show the plot
plt.show()

[ ] # Evaluate the model
test_batches.reset()
y_pred_cnn = model_cnn.predict(test_batches, steps=len(test_batches), verbose=1)
y_pred_classes_cnn = np.argmax(y_pred_cnn, axis=1)
y_true_cnn = test_batches.classes

# Confusion Matrix and Classification Report
conf_matrix_cnn = confusion_matrix(y_true_cnn, y_pred_classes_cnn)
class_labels = list(train_batches.class_indices.keys())

print("Confusion Matrix:")
print(conf_matrix_cnn)

print("Classification Report:")
class_report_cnn = classification_report(y_true_cnn, y_pred_classes_cnn, target_names=class_labels)
print(class_report_cnn)

# Specificity, Sensitivity, and AUC
tn, fp, fn, tp = conf_matrix_cnn.ravel()
specificity = tn / (tn + fp)
sensitivity = tp / (tp + fn)
auc = roc_auc_score(y_true_cnn, y_pred_cnn[:, 1])

print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"AUC: {auc}")

# Calculate F1-score
f1score = f1_score(y_true_cnn, y_pred_classes_cnn, average='weighted')
print(f"F1-score: {f1score}")

print(f"Accuracy: {accuracy_cnn}")

```

	<pre>[] import os import numpy as np # Define the folder path for saving TXT file txt_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_50_50/LR 0.001 EPOCH 100/Accuracy' # Create the directory if it doesn't exist os.makedirs(txt_folder_path_cnn, exist_ok=True) # Define the filename txt_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_duration{int(duration_cnn)}confusion matrix.txt' txt_filepath_cnn = os.path.join(txt_folder_path_cnn, txt_filename_cnn) # Save the information to TXT file with open(txt_filepath_cnn, 'w') as txt_file_cnn: # Write Confusion Matrix txt_file_cnn.write("Confusion Matrix:\n") txt_file_cnn.write(np.array2string(conf_matrix_cnn, separator=', ') + '\n\n') # Write Classification Report txt_file_cnn.write("Classification Report:\n") txt_file_cnn.write(class_report_cnn + '\n\n') # Write Specificity, Sensitivity, AUC, and Accuracy txt_file_cnn.write(f"Specificity: {specificity}\n") txt_file_cnn.write(f"Sensitivity: {sensitivity}\n") txt_file_cnn.write(f"AUC: {auc}\n") txt_file_cnn.write(f"F1-score: {f1score}\n") txt_file_cnn.write(f"Accuracy: {average_accuracy_cnn}\n")</pre>
Model CNN LR.0.0001 Dataset 50:50	<p>▼ CNN => LR 0.0001 EPOCH 100</p> <pre>[] # Define the CNN model model_cnn = Sequential([Conv2D(128, (3, 3), activation='relu', input_shape=(224, 224, 3)), MaxPooling2D((2, 2)), Conv2D(64, (3, 3), activation='relu'), MaxPooling2D((2, 2)), Conv2D(32, (3, 3), activation='relu'), MaxPooling2D((2, 2)), Flatten(), Dense(128, activation='relu', kernel_regularizer=l2(0.01)), Dropout(0.5), Dense(64, activation='relu', kernel_regularizer=l2(0.01)), Dropout(0.5), Dense(2, activation='softmax', kernel_regularizer=l2(0.01))]) # Compile the model model_cnn.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy']) history_cnn = model_cnn.fit(train_batches, validation_data=test_batches, epochs=100) # Simpan model ke dalam file HDF5 os.makedirs('/content/drive/MyDrive/dataset/hasil_50_50/LR 0.0001 EPOCH 100', exist_ok=True) model_cnn.save('/content/drive/MyDrive/dataset/hasil_50_50/LR 0.0001 EPOCH 100/CNN_model.h5')</pre>

```

[ ] # Convert the history to a DataFrame
history_df_cnn = pd.DataFrame(history_cnn.history)

# Define the CSV file path
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_50_50/LR_0.0001_EPOCH_100/History' # Update with your desired folder path
os.makedirs(csv_folder_path_cnn, exist_ok=True) # Ensure the folder exists

# rata-rata accuracy dari semua epoch yang dijalankan
average_accuracy_cnn = history_df_cnn['val_accuracy'].mean()

# Add a column for epoch times
history_df_cnn['epoch_time'] = (pd.to_datetime(datetime.utcnow()) - pd.to_datetime(datetime.utcnow())).total_seconds()

# Generate nana file CSV dengan informasi waktu, jumlah epoch, dan durasi model dijalankan
current_time_cnn = datetime.now().strftime("%Y-%m-%d %H%M%S")
epoch_count_cnn = len(history_cnn.history['loss'])
duration_cnn = history_df_cnn['epoch_time'].sum()

csv_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_CNN_duration{int(duration_cnn)}s_training_history.csv'

#path file CSV disimpan
csv_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_50_50/LR_0.0001_EPOCH_100/History' # Ganti dengan path folder yang diinginkan
csv_filepath_cnn = os.path.join(csv_folder_path_cnn, csv_filename_cnn)

# Save the DataFrame to CSV
history_df_cnn.to_csv(csv_filepath_cnn, index=False)

[ ] import os
import matplotlib.pyplot as plt
import numpy as np

# Define the folder path for saving plots
plot_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_50_50/LR_0.0001_EPOCH_100/Grafik'

# Create the directory if it doesn't exist
os.makedirs(plot_folder_path_cnn, exist_ok=True)

# Create a figure with specified size
plt.figure(figsize=(10, 5))

# Plot Training Loss
plt.subplot(1, 2, 1)
plt.plot(history_df_cnn['loss'], label='Training Loss')
plt.plot(history_df_cnn['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafik Loss')
plt.legend()
plt.grid(True)

# Plot Validation Accuracy
plt.subplot(1, 2, 2)
plt.plot(history_df_cnn['accuracy'], label='Training Accuracy')
plt.plot(history_df_cnn['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Grafik Accuracy')
plt.legend()
plt.grid(True)

# Adjust layout
plt.tight_layout()

# Save the combined plot
combined_plot_filename_cnn = f'{current_time_cnn}_epochs{epoch_count_cnn}_combined_plot.png'
combined_plot_filepath_cnn = os.path.join(plot_folder_path_cnn, combined_plot_filename_cnn)
plt.savefig(combined_plot_filepath_cnn)

# Show the plot
plt.show()

```

```

[ ] test_batches.reset() # Reset the generator to the beginning
y_pred_cnn = model_cnn.predict(test_batches, steps=len(test_batches), verbose=1)
y_pred_classes_cnn = np.argmax(y_pred_cnn, axis=1)
y_true_cnn = test_batches.classes

# Get the accuracy from the history
test_accuracy_cnn = history_cnn.history['val_accuracy']

from sklearn.metrics import f1_score

# Confusion Matrix
conf_matrix_cnn = confusion_matrix(y_true_cnn, y_pred_classes_cnn)
print("Confusion Matrix:")
print(conf_matrix_cnn)

# Classification Report
class_report_cnn = classification_report(y_true_cnn, y_pred_classes_cnn)
print("Classification Report:")
print(class_report_cnn)

# Specificity, Sensitivity, and AUC
tn, fp, fn, tp = conf_matrix_cnn.ravel()
specificity = tn / (tn + fp)
sensitivity = tp / (tp + fn)
auc = roc_auc_score(y_true_cnn, y_pred_cnn[:, 1], multi_class='ovr') # Adjust AUC calculation for multiple classes

print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"AUC: {auc}")

# Calculate F1-score
fiscore = f1_score(y_true_cnn, y_pred_classes_cnn, average='weighted')
print(f"F1-score: {fiscore}")

print(f"Accuracy: {average_accuracy_cnn}")

[ ] import os
import numpy as np

# Define the folder path for saving TXT file
txt_folder_path_cnn = '/content/drive/MyDrive/dataset/hasil_50_50/LR_0.0001_EPOCH_100/Accuracy'

# Create the directory if it doesn't exist
os.makedirs(txt_folder_path_cnn, exist_ok=True)

# Define the filename
txt_filename_cnn = f"{current_time_cnn}_epochs{epoch_count_cnn}_duration{int(duration_cnn)}confusion matrix.txt"
txt_filepath_cnn = os.path.join(txt_folder_path_cnn, txt_filename_cnn)

# Save the information to TXT file
with open(txt_filepath_cnn, 'w') as txt_file_cnn:
    # Write Confusion Matrix
    txt_file_cnn.write("Confusion Matrix:\n")
    txt_file_cnn.write(np.array2string(conf_matrix_cnn, separator=', ') + '\n\n')

    # Write Classification Report
    txt_file_cnn.write("Classification Report:\n")
    txt_file_cnn.write(class_report_cnn + '\n\n')

    # Write Specificity, Sensitivity, AUC, and Accuracy
    txt_file_cnn.write(f"Specificity: {specificity}\n")
    txt_file_cnn.write(f"Sensitivity: {sensitivity}\n")
    txt_file_cnn.write(f"AUC: {auc}\n")
    txt_file_cnn.write(f"F1-score: {fiscore}\n")
    txt_file_cnn.write(f"Accuracy: {average_accuracy_cnn}\n")

```

GUI Tampilan Hasil

```
import numpy as np
import cv2
from tkinter import Tk, Button, Label, filedialog, messagebox, StringVar, Frame
from PIL import Image, ImageTk
from tensorflow.keras.models import load_model

# Fungsi untuk menambahkan saluran alfa ke gambar
def add_alpha_channel(image):
    b, g, r = cv2.split(image)
    alpha = np.ones(b.shape, dtype=b.dtype) * 255 # Membuat saluran alfa penuh (tidak transparan)
    return cv2.merge((b, g, r, alpha))

# Fungsi untuk mengubah gambar menjadi grayscale dan menambahkan saluran alfa
def convert_to_grayscale_with_alpha(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    alpha = np.ones(gray.shape, dtype=gray.dtype) * 255 # Membuat saluran alfa penuh (tidak transparan)
    return cv2.merge((gray, gray, gray, alpha))

# Fungsi untuk memuat gambar dengan pengecekan
def load_image_with_check(path):
    image = cv2.imread(path)
    if image is None:
        messagebox.showerror("Error", f"Gambar pada path {path} tidak berhasil dimuat.")
        return None
    return image

# Fungsi untuk memilih file gambar
def select_file():
    file_path = filedialog.askopenfilename()
    return file_path

def show_image(image, label, frame_width, frame_height):
    image = cv2.resize(image, (frame_width, frame_height))
    image = Image.fromarray(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    image_tk = ImageTk.PhotoImage(image)
    label.config(image=image_tk)
    label.image = image_tk

def load_gambar1():
    global gambar1
    file_path = select_file()
    gambar1 = load_image_with_check(file_path)
    if gambar1 is not None:
        show_image(gambar1, label_gambar1_image, 224, 224)

def load_gambar2():
    global gambar2
    file_path = select_file()
    gambar2 = load_image_with_check(file_path)
    if gambar2 is not None:
        show_image(gambar2, label_gambar2_image, 224, 224)

def process_and_predict():
    global gambar1, gambar2, concatenate

    if gambar1 is None or gambar2 is None:
        messagebox.showerror("Error", "Silakan muat kedua gambar terlebih dahulu.")
        return

    # Mengubah ukuran gambar menjadi 224x224
    gambar1_resized = cv2.resize(gambar1, (224, 224))
    gambar2_resized = cv2.resize(gambar2, (224, 224))

    # Mengonversi gambar menjadi grayscale dan menambahkan saluran alfa
    gambar1_gray_with_alpha = convert_to_grayscale_with_alpha(gambar1_resized)
    gambar2_gray_with_alpha = convert_to_grayscale_with_alpha(gambar2_resized)

    # Menentukan lebar area yang akan ditimpak
    width_to_overlay = 224

    # Memisahkan gambar pertama menjadi dua bagian
    gambar1_left_with_alpha = gambar1_gray_with_alpha[:, :width_to_overlay]
    gambar1_right_with_alpha = gambar1_gray_with_alpha[:, width_to_overlay:]

    # Memisahkan gambar kedua menjadi bagian yang akan menerima
    gambar2_overlay_with_alpha = gambar2_gray_with_alpha[:, :width_to_overlay]

    # Mengatur transparansi pada bagian yang akan digabungkan
    alpha_gambar1 = 0.7 # 70% transparansi
    alpha_gambar2 = 0.3 # 30% transparansi

    # Melakukan blending pada bagian yang akan digabungkan
    blended_overlay = cv2.addWeighted(gambar1_left_with_alpha, alpha_gambar1, gambar2_overlay_with_alpha, alpha_gambar2, 0)

    # Menggabungkan dengan menempatkan bagian kiri gambar pertama dengan hasil blending
    concatenate = np.concatenate((blended_overlay, gambar1_right_with_alpha), axis=1)

    # Preprocessing gambar untuk model (sesuai dengan preprocessing saat melatih model)
    concatenate_preprocessed = concatenate[:, :, :3] / 255.0 # Normalisasi gambar

    # Memuat model
    model = load_model('hasil_80_LR_0.00001_EPOCH_100/CHN_model.h5')

    # Melakukan prediksi
    predictions = model.predict(np.expand_dims(concatenate_preprocessed, axis=0))

    # Menentukan klasifikasi berdasarkan prediksi
    class_names = ['Retinopati Diabetik', 'Normal']
    predicted_class = class_names[np.argmax(predictions)]
```

```

# Menampilkan hasil penggabungan
show_image(concatenate[:, :, :3], label_concatenate_image, 124, 224)

# Menampilkan hasil prediksi
result_text.set("Klasifikasi: " + predicted_class)

def reset():
    global gambar1, gambar2, concatenate
    gambar1 = None
    gambar2 = None
    concatenate = None
    label_gambar1.image.config(image="", text="")
    label_gambar2.image.config(image="", text="")
    label_concatenate.image.config(image="", text="")
    result_text.set("Klasifikasi: ")

    # Inisialisasi variabel global
    gambar1 = None
    gambar2 = None
    concatenate = None

    # Inisialisasi GUI
    root = Tk()
    root.title("Klasifikasi Early Fusion Retinopati Diabetik")

    Frame_top = Frame(root)
    Frame_top.pack(side="top", ipady=10)

    title_label = Label(Frame_top, text="KLASIFIKASI PENYAKIT RETINOPATI DIABETIK MENGGUNAKAN CNN DENGAN MULTIMODAL DATA (FUSION PADA DATASET CITRA RETINA)", font=("Arial", 10), justify="center")
    title_label.pack()

    Frame_gambar = Frame(root)
    Frame_gambar.pack(ipady=10)

    frame_gambar1 = Frame(Frame_gambar, bd=2, relief="solid", width=250, height=300)
    frame_gambar1.pack(side="left", padx=10, pady=10)
    frame_gambar1.pack_propagate(False)
    btn_load_gambar1 = Button(frame_gambar1, text="Mut Gambar Fokus", command=load_gambar1)
    btn_load_gambar1.pack()
    label_gambar1_image = Label(frame_gambar1)
    label_gambar1_image.pack(pady=10)

    frame_gambar2 = Frame(Frame_gambar, bd=2, relief="solid", width=250, height=300)
    frame_gambar2.pack(side="left", padx=10, pady=10)
    frame_gambar2.pack_propagate(False)
    btn_load_gambar2 = Button(frame_gambar2, text="Mut Gambar OCT", command=load_gambar2)
    btn_load_gambar2.pack()
    label_gambar2_image = Label(frame_gambar2)
    label_gambar2_image.pack(pady=10)

    frame_concatenate = Frame(Frame_gambar, bd=2, relief="solid", width=250, height=300)
    frame_concatenate.pack(side="left", padx=10, pady=10)
    frame_concatenate.pack_propagate(False)
    btn_process_predict = Button(frame_concatenate, text="Proses dan Prediksi", command=process_and_predict)
    btn_process_predict.pack()
    label_concatenate_image = Label(frame_concatenate)
    label_concatenate_image.pack(pady=10)

    result_text = StringVar()
    result_label = Label(root, textvariable=result_text, width=30, font=("Arial", 14, "bold"))
    result_label.pack(ipady=10)

    reset_button = Button(root, text="Reset gambar", command=reset)
    reset_button.pack(ipady=5)

root.mainloop()

```