TUGAS AKHIR – EB184803

KLASIFIKASI JENIS KERUSAKAN PADA RETINA DARI CITRA *OPTICAL COHERENCE TOMOGRAPHY* 2 DIMENSI BERBASIS *CONVOLUTIONAL NEURAL NETWORK*

Putri Norma Aprilia R
07311640000007

DOSEN PEMBIMBING:
Dr. Tri Arief Sardjono, S.T., M.T.
Nada Fitrieyatul Hikmah, S.T., M. T.

PROGRAM SARJANA
DEPARTEMEN TEKNIK BIOMEDIK
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

TUGAS AKHIR – EB184803

KLASIFIKASI JENIS KERUSAKAN PADA RETINA DARI CITRA *OPTICAL COHERENCE TOMOGRAPHY* 2 DIMENSI BERBASIS *CONVOLUTIONAL NEURAL NETWORK*

Putri Norma Aprilia R
07311640000007

DOSEN PEMBIMBING:
Dr. Tri Arief Sardjono, S.T., M.T.
Nada Fitrieyatul Hikmah, S.T., M. T.

PROGRAM SARJANA
DEPARTEMEN TEKNIK BIOMEDIK
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

FINAL PROJECT – EB184803

CLASSIFICATION OF RETINAL ABNORMALITIES FROM 2D OPTICAL COHERENCE TOMOGRAPHY BASED ON CONVOLUTIONAL NEURAL NETWORK

Putri Norma Aprilia R
07311640000007

SUPERVISOR:
Dr. Tri Arief Sardjono, S.T., M.T.
Nada Fitrieyatul Hikmah, S.T., M. T.

UNDERGRADUATE PROGRAM
BIOMEDICAL ENGINEERING DEPARTMENT
FACULTY OF INTELLIGENT ELECTRICAL AND INFORMATICS TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa sebagian maupun keseluruhan isi tugas akhir saya yang berjudul "**Klasifikasi Jenis Kerusakan pada Retina dari Citra *Optical Coherence Tomography* 2D berbasis *Convolutional Neural Network***" merupakan hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya mandiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Juli 2020

Putri Nonna Aprilia K

NRP. 07311640000007

Tugas Akhir disusun untuk memenuhi salah satu syarat memperoleh gelar
Sarjana Teknik (S.T.)
di
Institut Teknologi Sepuluh Nopember

Oleh
Putri Norma Aprilia Radayanti
NRP. 07311640000007

Tanggal Ujian : **14 Juli 2020**
Periode Wisuda : **September 2020**

Disetujui Oleh :

Dosen Penguji :

1. Dr. Rachmad Setiawan, S.T., M.T.
NIP. 196905291995121001

2. Atar Fuady Babgei, S.T., M.Sc.
NIP. 198911112018121001

3. Muhammad Yazid, B.Eng., M.Eng.
NIP. 198004202015041001

Dosen Pembimbing :

1. Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 197002121995121001

2. Nada Fitrieyatul Hikmah, S.T., M.T.
NIP. 199001072018032001

Kepala Departemen Teknik Biomedik,

Dr. Achmad Arifin, S.T., M.Eng.
NIP. 197103141997021001

KLASIFIKASI JENIS KERUSAKAN PADA RETINA DARI CITRA *OPTICAL COHERENCE TOMOGRAPHY* 2 DIMENSI BERBASIS *CONVOLUTIONAL NEURAL NETWORK*

Nama mahasiswa	:	Putri Norma Aprilia R
NRP	:	07311540000007
Pembimbing	:	Dr. Tri Arief Sardjono, S.T., M.T.
	:	Nada Fitrieyatul Hikmah, S.T., M.T.

ABSTRAK

Salah satu modalitas yang digunakan untuk pemeriksaan kerusakan retina adalah *Optical Coherence* Tomography (OCT). Saat ini, pemeriksaan menggunakan OCT masih bersifat manual dimana setelah dilakukan akuisisi, dokter akan melakukan diagnosa melalui beberapa tahap. Namun pencitraan menggunakan OCT rentan terhadap *speckle noise* dan memiliki kontras gambar yang rendah antar lapisan retina sehingga menyebabkan sulitnya dalam membedakan struktur anatomi retina yang berpengaruh pada penentuan jenis kerusakan retina. Dalam tugas akhir ini diajukan sebuah sistem untuk mengklasifikasi jenis kerusakan retina. Berbeda dengan penelitian sebelumnya, alih-alih menggunakan *transfer learning* untuk mengklasifikasi jenis kerusakan retina, sistem yang diajukan menggunakan rancangan *Convolutional Neural Network* (CNN) yang didesain mandiri dengan konfigurasi yang lebih sederhana untuk melakukan klasifikasi. Sistem yang diajukan juga dilengkapi dengan tahapan pra pengolahan berupa segmentasi area ROI serta reduksi *speckle noise*. Kemudian dilakukan tahapan *balancing data* guna menyeimbangkan persebaran jumlah data masing-masing kelas. Algoritma segmentasi ROI yang telah diimplementasikan memiliki prosentase tingkat kegagalan dalam mensegmentasi area ROI citra sebesar 0.016%. Sedangkan implementasi reduksi *noise* menggunakan bilateral filter dengan parameter *sigma range* sebesar 0.5 dan *sigma spatial* sebesar 20 menghasilkan nilai PSNR yang paling tinggi yaitu sebesar 32.56 dB. Kemudian terdapat 4 prosedur pengujian CNN yang dilakukan. Pengujian kondisi 1 dilakukan dengan kondisi citra input merupakan citra terpraproses serta tidak dilakukannya *balancing data*. Pengujian kondisi 2 dilakukan dengan kondisi citra input merupakan citra terpraproses serta dilakukan *balancing data* menggunakan teknik *weight balancing*. Pengujian kondisi 3 dilakukan dengan kondisi citra input merupakan citra terpraproses serta dilakukan tahapan *balancing data* menggunakan teknik *random undersampling.* Pengujian kondisi 4 dilakukan dengan kondisi citra input merupakan citra yang tidak terpraproses serta dilakukannya tahapan *balancing data* menggunakan teknik *random undersampling*. Hasil terbaik didapatkan pada pengujian kondisi 4 dimana dihasilkan nilai akurasi sebesar 94.2%, rata-rata presisi 94.2%, rata-rata sensitivitas 94.2% dan rata-rata spesifisitas sebesar 98.2%.

Kata kunci: bilateral filter, convolutional neural network, marching squares algorithm, optical coherence tomography, random undersampling

CLASSIFICATION OF RETINAL ABNORMALITIES FROM 2D OPTICAL COHERENCE TOMOGRAPHY BASED ON CONVOLUTIONAL NEURAL NETWORK

Name	:	Putri Norma Aprilia R
Student Identity Number	:	07311540000007
Supervisor	:	Dr. Tri Arief Sardjono, S.T., M.T.
	:	Nada Fitrieyatul Hikmah, S.T., M.T.

ABSTRACT

One of the modalities used to examine retinal abnormality is Optical Coherence Tomography (OCT). The examination using OCT scan is still manual. The opthalmologist will manually identify all the retinal layers, then identify the abnormal layer where is the abnormality occurs, and finally correlate with the pathophysiology of the disease. This process is often complicated, time consuming, and laborious. In other side, imaging using OCT is susceptible to speckle noise that reduce image quality and it is hard to clearly distinguish anatomical structures that contribute to classification process of the retinal abnormality. An automated classification of retinal abnormality is proposed in the research. In contrast to previous studies, instead of using pre-trained model of CNN to classify the types of retinal abnormality, the proposed system use a self-designed CNN model with a simpler configuration for the classification process. The proposed system is also accompanied by an algorithm for segmenting the Region of Intrest (ROI) and reducing speckle noise of the retina. Then, the balancing data process is performed to balance the distribution of dataset. The segmentation algorithm has been implemented with a percentage of failure rate in segmenting the ROI by 0.016%. Whereas the reduction speckle noise using bilateral filter result the best PSNR value by 32.56 dB. The evaluation of CNN is done with four different conditions to know the effect of segmented image and balanced data process. The first condition is preprocessed image and unbalanced data. The second condition is preprocessed image and balanced data using weight balancing method. The third condition is preprocessed image and balanced data using random undersampling method. And the last condition is unpreprocessed image and balanced data using undersampling method. The best evaluation result is obtained in the last condition with the results an accuracy of 94.2%, average precision of 94.2%, average sensitivity of 94.2%, and average specificity of 98.2%.

Keywords: bilateral filter, convolutional neural network, marching squares algorithm, random undersampling

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT yang telah melimpahkan berkah, rahmat, serta hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Klasifikasi Jenis Kerusakan pada Retina dari Citra *Optical Coherence Tomography* 2D berbasis *Convolutional Neural Network*". Penulis ingin menyampaikan terimakasih kepada:

1. Keluarga penulis, khususnya bapak dan ibu yang senantiasa memberikan dukungan baik dukungan material maupun non-material.
2. Dr. Achmad Arifin, S.T., M.Eng selaku Kepala Departemen Teknik Biomedik, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember.
3. Dr. Tri Arief Sardjono, S.T., M.T. dan Ibu Nada Fitrieyatul Hikmah, S.T., M.T. selaku dosen pembimbing yang selalu memberikan motivasi, ilmu dan saran serta bantuan dalam penelitian ini.
4. Prof. Dr. Ir. Mohammad NUH, DEA selaku dosen wali
5. Bapak ibu dosen pengajar Departemen Teknik Biomedik
6. Seluruh teman-teman penulis terutama teman-teman angkatan e-56 dan laboratorium B205.
7. Serta seluruh pihak yang tidak dapat disebutkan satu persatu yang telah membantu penulis hingga Tugas Akhir ini ini bisa diselesaikan dengan semaksimal mungkin.

Kesempurnaan hanya milik Allah SWT, penyusunan tugas akhir ini tentu masih banyak kekurangan. Untuk itu penulis sangat mengharapkan kritik dan saran yang membangun.

Surabaya, Juli 2020

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Retina merupakan membran sensorik yang berfungsi untuk mendeteksi cahaya serta memainkan peran penting dalam persepsi visual. Kerusakan pada retina akan berpengaruh pada kemampuan penglihatan bahkan kebutaan. Terdapat berbagai macam jenis kerusakan retina diantaranya adalah CNV (*Choroidal Neo-Vascularization*), DME (*Diabetic Macular Edema*), dan *drusen*. *Drusen* dan CNV merupakan jenis kerusakan yang terjadi pada penyakit AMD (*Age Related Macular Degeneration*). Menurut WHO, AMD menempati urutan ketiga diantara penyebab kebutaan global setelah Katarak (51%), Glaukoma (8%), dan AMD (5%) [1]. Sedangkan DME merupakan penyebab utama hilangnya penglihatan bagi pasien dengan diabetik retinopati. WHO menyatakan bahwa pada tahun 2014 jumlah orang dewasa dengan penyakit diabetes telah mencapai 422 juta jiwa dimana 6.8% diantaranya menderita DME [2].

Salah satu modalitas yang digunakan untuk pemeriksaan retina adalah OCT (*Optical Coherence* Tomography). OCT merupakan teknik pencitraan *non-invasive* yang menggunakan cahaya dengan koherensi rendah untuk menangkap potongan gambar secara melintang [3]. Hasil citra OCT sendiri merepresentasikan informasi lapisan retina secara detail [4]. Untuk saat ini, pemeriksaan retina menggunakan OCT masih bersifat manual dimana setelah dilakukan akuisisi, dokter akan melakukan diagnosis melalui 3 tahap [5]. Yang pertama dokter akan melakukan identifikasi terhadap semua struktur anatomi retina yang terdapat pada citra. Kemudian dokter melakukan identifikasi terhadap sesuatu yang abnormal dan di lapisan mana keabnormalitasan itu terjadi. Selanjutnya dokter akan mengkorelasikannya dengan *pathophysiology* dari suatu penyakit tertentu.

Disisi lain, pencitraan menggunakan OCT rentan terhadap artefak pergerakan mata dan memiliki *signal-to-noise-ratio* (SNR) yang rendah karena adanya *speckle noise* [6]. Hal tersebut tersebut tentunya dapat menurunkan kualitas citra serta akurasi dalam proses analisa citra. Selain itu, rendahnya kontras gambar antar lapisan retina menyebabkan sulitnya dalam membedakan struktur anatomi retina yang berpengaruh pada penentuan jenis kerusakan retina sehingga segmentasi lapisan retina secara manual dapat memakan waktu dan terlalu terbatas untuk klasifikasi sensitif pada skala piksel [7].

Diagnosis otomatis menggunakan OCT masih berada pada tahap awal karena hanya terdapat publikasi peneltian dibidang akademik dan belum ada pembuatan produk secara komersial [8]. Dalam beberapa tahun terakhir, *deep learning* mengalami kemajuan dalam bidang pencitraan medis. Kermany, dkk mengimplementasikan *transfer learning* menggunakan model *inception-v3* untuk mengklasifikasi 4 jenis kerusakan pada retina dan menghasilkan nilai akurasi

sebesar 96.6% [9]. Islam, dkk melatih ulang 11 model *transfer learning* kemudian memilih model terbaik untuk dioptimasi dan digunakan sebagai fitur ekstraktor. Selanjutnya fitur tersebut dilatih menggunakan ANN dan menghasilkan akurasi sebesar 98.6% [10].

Berbeda dengan penelitian sebelumnya, alih-alih menggunakan *transfer learning* untuk mengklasifikasi jenis kerusakan retina, sistem yang diajukan menggunakan rancangan CNN yang didesain mandiri dengan konfigurasi yang lebih sederhana untuk melakukan klasifikasi. Sistem yang diajukan juga dilengkapi dengan tahapan alogirtma pra pengolahan untuk mensegmentasi area ROI serta mereduksi *speckle noise* pada citra. Penambahan algoritma segmentasi dan dengan penggunaan konstruksi lapisan yang sederhana diharapkan mampu menghasilkan performa klasifikasi yang lebih baik dari penelitian sebelumnya.

1.2 Rumusan Masalah

Terdapat dua rumusan masalah yang akan dibahas dalam tugas akhir ini. Permasalahan yang pertama adalah merancang tahapan pra pengolahan citra untuk memperbaiki kualitas citra. Rancangan pra pengolahan yang digunakan meliputi segmentasi area *Region of Interest* (ROI) serta reduksi *speckle noise.* Permasalahan yang kedua adalah merancang konfigurasi CNN untuk proses klasifikasi jenis kerusakan retina. Selain merancang konfigurasi CNN, juga dilakukan perancangan *balancing data* sebelum dilakukannya proses pelatihan. *Balancing data* akan menyeimbangkan persebaran jumlah data pada masing-masing kelas.

1.3 Batasan Masalah

Terdapat empat batasan masalah dalam tugas akhir ini. Batasan masalah yang pertama adalah citra input merupakan citra hasil modalitas OCT 2D dengan format .jpeg yang digunakan oleh Kermany, dkk [10]. Batasan masalah yang kedua adalah input yang digunakan terdiri dari empat kelas yaitu: CNV, DME, DRUSEN, dan NORMAL. Batasan yang ketiga adalah metode klasifikasi yang digunakan adalah CNN. Batasan masalah yang terakhir adalah output yang dihasilkan berupa hasil klasifikasi beserta nilai prediksi probabilitas dari masing-masing kelas.

1.4 Tujuan dan Manfaat

Terdapat dua tujuan yang menjawab rumusan masalah dalam tugas akhir ini. Tujuan yang pertama adalah mengetahui tahapan pra pengolahan citra yang meliputi segmentasi ROI dan reduksi *speckle noise* pada citra. Tahapan pra pengolahan citra ini akan meningkatkan kualitas citra serta menghilangkan informasi yang tidak diperlukan. Tujuan yang kedua adalah mengetahui rancangan konfigurasi CNN yang akan digunakan. Pada tujuan ini juga akan diketahui bagaimana cara melakukan proses *balacing data.*

Manfaat yang ingin dicapai dalam penelitian ini meliputi aspek teoritis dan aspek praktis. Manfaat dalam aspek teoritis yang diharapkan adalah terciptanya artikel ilmiah untuk memperkaya wawasan dalam bidang biomedik serta dengan

adanya penelitian ini dapat mengembangkan kreatifitas mahasiswa dalam pengembangan teknologi yang lebih efektif dan efisien. Sedangkan manfaat praktis yang dapat dihasilkan dari penelitian ini adalah berfungsi sebagai alat bantu *ophthalmologist* untuk melakukan klasifikasi jenis kerusakan pada retina.

1.5 Kontribusi Penelitian

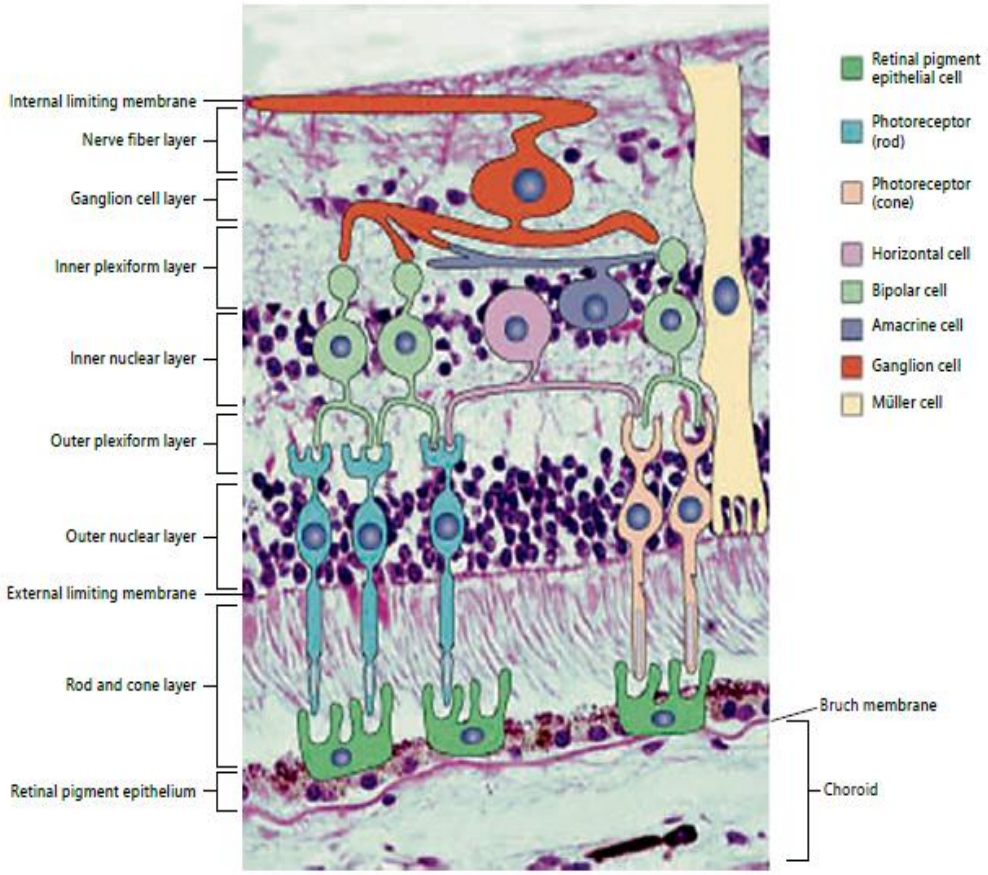
Kontribusi dari penelitian ini dapat dilihat dari dua aspek yaitu aspek ilmiah dan praktis. Pada aspek ilmiah, kontribusinya adalah menjadi suatu pengembangan untuk pendidikan dan penelitian dalam bidang biomedik. Penelitian ini mencoba mengimplementasikan *deep learning* berupa CNN serta dilengkapi dengan algoritma pra pengolahan sebelum citra diumpankan ke tahap klasifikasi.

Sedangkan kontribusi pada aspek praktis adalah proses segmentasi manual yang bergantung pada kontras citra, dan rentannya citra terhadap adanya *speckle noise* yang menyebabkan sulitnya untuk membedakan struktur anatomi dari retina sehingga penentuan jenis kerusakan retina dapat memakan waktu. Dengan adanya algoritma otomatis ini diharapkan dapat menjadi alat bantu bagi tenaga medis dalam penentuan jenis kerusakan pada retina.

BAB II
TINJAUAN PUSTAKA

2.1 Anatomi Retina pada Citra OCT

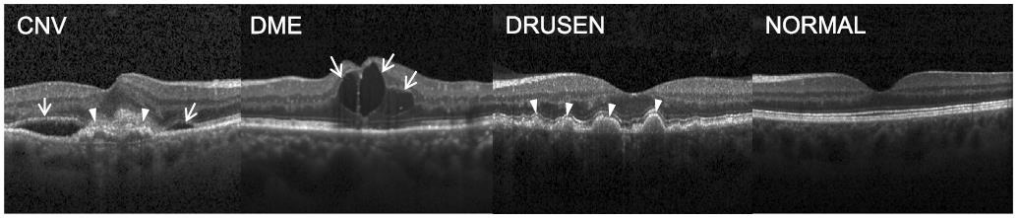
Retina merupakan membran sensorik yang melapisi permukaan bagian dalam bola mata dan berfungsi untuk mendeteksi cahaya serta memainkan peran penting dalam persepsi visual. Retina terdiri dari neuroretina (bagian visual) dan epitel pigmen (bagian non-visual) seperti yang tertera pada Gambar 2.1. Neuroretina terdiri dari sembilan lapisan yaitu: lapisan fotoreseptor/lapisan sel kerucut dan sel batang, membran limitans eksterna, lapisan luar inti fotoreseptor, lapisan luar plexiform, lapisan dalam badan inti, lapisan dalam plexiform, lapisan sel ganglion, lapisan serat saraf, dan membran limitans interna [11]. Selain itu pada neuroretina terdapat sel horizontal dan sel amakrin yang membentuk jalur lateral untuk mengatur sinyal dari sel fotoreseptor. Dari sel fotoreseptor sinyal diteruskan ke sel bipolar melalui lapisan sinapsis luar (*outer plexiform layer*). Kemudian dari sel bipolar informasi diteruskan ke sel ganglion melalui lapisan sinapsis dalam (*inner plexiform layer*). Akson sel ganglion meluas ke daerah posterior, ke diskus



Gambar 2. 1 Struktur Anatomi Retina [11]

optikus, dan keluar dari bola mata sebagai nervus optikus. Sedangkan bagian epitel pigmen atau yang biasa dikenal dengan *retinal pigment epithelium* (RPE) terletak dibawah lapisan sel fotoreseptor. RPE merupakan sel epitel yang mengandung pigmen melanin yang berfungsi untuk menyerap cahaya sehingga dapat mencegah pantulan dan penyebaran cahaya di dalam bola mata. Selain itu RPE juga berfungsi untuk memfasilitasi difusi nutrisi dari koroid dan menghapus sel fotoreseptor yang mati [11]. Di bawah lapisan RPE terdapat membran Bruch's yang memisahkan bagian retina dan koroid. Gambar tersebut diambil secara *cross section* dari subjek laki-laki normal di sebuah rumah sakit yang beradi di USA. Dengan menggunakan modalitas OCT terdapat 18 lapisan retina yang dapat diklasifikasi sebagi berikut [12]:

1. *Posterior cortical vitreous*
2. *Preretinal space:* Ruang yang terbentuk diantara *posterior corctical vitreous* dan *internal limiting membrane.*
3. *Internal limiting membrane (ILM):* Merupakan batas antara retina dan vitreus humor yang dibentuk oleh sel astrosit dan bagian akhir dari sel müller serta berhubungan dengan bagian utama membran/lamina basalis.
   Nerve fiber layer (NFL): NFL terdiri dari akson sel ganglion yang dibentuk oleh perluasan serat saraf optik.
4. *Ganglion cell layer (GCL):* GCL merupakan lapisan sel tunggal yang tebal, kecuali dibagian makula dan mengandung inti dari sel ganglion.
5. *Inner plexiform layer (IPL):* Terdiri dari koneksi sinaptik antara akson sel-sel bipolar dan dendrit sel ganglion.
6. *Inner nuclear layer (INL):* Terdiri dari sel horizontal, sel bipolar, sel amakrin, neuron interpelksiform, sel müller dan beberapa sel ganglion.
7. *Outer plexiform layer (OPL):* Terdiri dari koneksi sinaptik anatar fotoreseptor bipolar dan sel horizontal.
8. *(Inner half) Henle's nerve fibre layer (HL):* Terdiri dari fotoreseptor axon.
   (Outer half) Outer nuclear layer (ONL): Terdiri dari badan fotoreseptor sel batang dan sel kerucut. ONL tertebal berada pada bagian fovea (50 μm).
9. *External limiting membrane (ELM):* Terbentuk dari *zonulae adherens* antara sel müller dan sel fotoreseptor bagian dalam.
10. *Myoid zone (MZ):* Bagian terdalam dari segmen fotoreseptor dalam (IS: *Inner Segment*) yang berisi: retikulum endoplasma kasar dan halus, ribosom, badan golgi, serta mikrotubulus.
11. *Ellipse zone (EZ):* Bagian terluar dari segmen fotoreseptor dalam yang berisi mitokondria.
    Photoreceptor integrity line (PIL) or IS/OS junction: Menghubungkan bagian IS dan OS pada fotoreseptor.
12. *Photoreceptor outer segments (OS):* Merupakan bagian dari segmen fotoreseptor luar (OS: *Outer Segment*).



Gambar 2. 2 Jenis Kerusakan pada Retina [9]

13. *Interdigitation zone (IZ)*
14. *RPE/Bruch's complex:* Merupakan lapisan yang paling dekat dengan koroid yang terdiri dari zona fagosom dan zona melanosom.
15. *Choriocapillaris:* Lapisan tipis dari reflektifitas moderat didalam koroid.
16. *Sattler's layer:* Lapisan tebal berbentuk bulat atau oval dengan inti hiporeflektif pada bagian tengah koroid.
17. *Haller's layer:* Lapisan tebal berbentuk oval dengan inti hiporeflektif pada bagian luar koroid.
18. *Choroidal-scleral juncture:* Area terluar dari koroid.

2.2 Jenis Kerusakan pada Retina

Beberapa jenis kerusakan pada retina yang juga akan dijadikan input citra pada penelitian yang disusulkan ini digambarkan pada Gambar 2.2.

2.2.1 CNV (*Choroidal Neovascularization*)

CNV merupakan penyakit yang berkaitan dengan *Age-related Macular Degeneration* (AMD). AMD merupakan kondisi medis yang sering terjadi pada usia lanjut (lebih dari 50 tahun) dimana terjadi proses penurunan fungsi pada makula. Makula adalah pusat dari retina yang berfungsi sebagai reseptor sinyal cahaya, yang kemudian akan dihantarkan ke otak. Penurunan fungsi makula dapat mengakibatkan gangguan penglihatan bahkan kebutaan. Proses terjadinya CNV berasal dari sel RPE yang menumpahkan sampah selular ke dalam *extra cellular space.* Material yang ada di *extra cellular space* ini menumpuk bersama dengan protein dan imun kompleks diantara membran dasar RPE dan membran *Bruch*. Hasil akumulasi material ini kemudian disebut dengan *drusen*. *Drusen* dapat menyebabkan pengembangan CNV dengan cara menghalangi difusi oksigen dan nutrisi dari *choriocapillaries* ke RPE dan lapisan terluar retina. CNV terbentuk dari adanya pembuluh darah abnormal yang baru tumbuh dan berkembang biak dari pembuluh neovaskularisasi. CNV dapat dicirikan dengan adanya penebalan fusiformis pada lapisan RPE, hilangnya linearitas, dan proliferasi pembuluh darah baru yang abnormal dimana perubahan tersebut dapat direkam dan diamati menggunakan OCT. CNV dapat terjadi dengan cepat pada individu dengan kelainan pada membran *bruch*. [13]

2.2.2 DME (*Diabetic Macular Edema*)

DME merupakan penyebab utama hilangnya penglihatan bagi pasien dengan diabetik retinopati yang disebabkan adanya akumulasi cairan pada daerah makula. Diabetik biasanya mempengaruhi kedua mata secara simetris dan terjadi ketika kadar gula dalam darah menyebabkan kerusakan pembuluh darah pada retina. Pembuluh ini mulai berdarah atau bocor yang menyebabkan cairan meresap ke dalam retina dan mengganggu fungsinya. Diagnosis, pengobatan, dan tindak lanjut dari DME mengalami perkembangan dalam pencitraan fundus, seperti *optical coherence tomography. Photocoagulation, intravitreal injection,* dan *pars plana vitrectomy surgery* merupakan modalitas pengobatan yang ada saat ini. Patogenesis dari DME belum sepenuhnya didefinisikan karena terdapat proses yang kompleks dengan berbagai faktor yang berkontribusi. Hiperglikemia kronis, hiperkolesterolemia, radikal oksigen bebas, dan protein kinase C terlibat dalam proses patologis. Karakteristik umum dari patogenesis DME adalah peningkatan level dari *vascular endothelial growth factor* (VEGF), yang bertanggung jawab terhadap gangguan *inner blood-retinal barrier* (BRB). Gangguan BRB menyebabkan akumulasi cairan subretinal dan intraretinal, yang pada akhirnya mengubah struktur makula beserta fungsinya. Peningkatan kadar VEGF dapat menyebabkan kebocoran pembuluh darah dan kerusakan BRB, baik secara langsung maupun tidak langsung, melalui berbagai mekanisme. Hipokxia, iskemia, faktor genetik, dan mediator inflamasi dapat berkontribusi pada patogenesis kerusakan BRB dan DME. [14]

2.2.3 *Drusen*

Drusen merupakan hasil akumulasi atau endapan materi ekstraseluler yang berwarna putih atau kuning, berbentuk bulat dan terbentuk diantara membrane *bruch* dan lapisan RPE (*retinal pigment epithelium*). Drusen terbentuk dari lipid dan protein lemak. Keberadaan beberapa drusen yang berukuran kecil merupakan suatu keadaan yang normal seiring dengan bertambahnya usia, dan hampir pada kebanyakan orang di usia lebih dari 40 memiliki '*hard drusen'* [15]. Bagaimanapun, keberadaan drusen yang berukuran lebih besar dengan jumlah yang lebih banyak pada area makula merupakan gejala dini adanya penyakit degenerasi makula atau AMD. Drusen yang terkait dengan degenerasi makula berbeda dengan *optic disc drusen* yang terjadi pada daerah kepala saraf optik. Namun kedua jenis drusen ini dapat diamati oleh oftalmoskopi [16]. '*hard drusen'* dapat menyatu menjadi '*soft drusen'* yang merupakan manifestasi dari degenerasi makula. Sumber protein dan lipid di drusen berasal dari RPE dan koroid. Komposisi protein dalam drusen terdiri dari apolipoprotein dan protein teroksidasi yang kemungkinan bersal dari darah, RPE, dan fotoreseptor [17].

2.3 Pengolahan Citra Digital

2.3.1 Citra digital

Citra digital merupakan suatu representasi atau gambaran dari suatu objek yang dapat diolah oleh komputer. Sebuah citra digital dapat digambarkan oleh sebuah matriks yang terdiri dari M kolom dan N baris, dimana perpotongan antara kolom dan baris dapat disebut sebagai piksel. Piksel memiliki dua parameter yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x, y) adalah $f(x, y)$ yang merupakan besar intensitas atau warna dari piksel pada koordinat tersebut.

2.3.2 Citra biner

Citra biner merupakan citra yang pikselnya hanya memiliki dua nilai, yaitu minimal dan maksimal. Kedua nilai tersebut merupakan representasi warna hitam dan warna putih. Warna hitam diwakili oleh nilai minimal, yaitu nilai 0. Warna putih diwakili oleh nilai maksimal, yaitu 255 (misalkan menggunakan resolusi 8 bit). Dalam pengolahan citra digital, citra biner biasanya digunakan untuk memisahkan antara objek dan latar belakang.

2.3.3 Operasi morfologi

Operasi morfologi merupakan sebuah metode untuk analisa citra yang didasarkan pada teori dasar matematika yaitu teori himpunan, dimana citra diasumsikan tersusun dari himpunan piksel [18]. Jenis-jenis operasi morfologi di antaranya adalah dilasi, erosi, closing, dan opening. Secara berurutan, persamaan yang digunakan untuk masing-masing operasi yaitu:

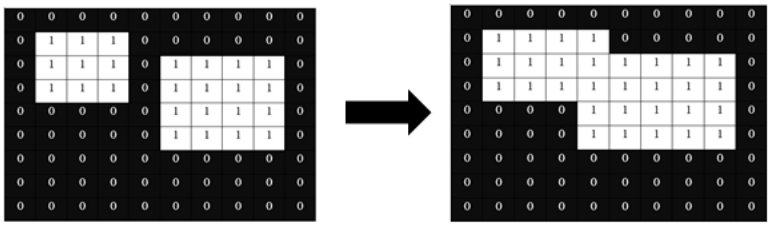
$$A \oplus B \tag{2.1}$$

$$A \ominus B \tag{2.2}$$

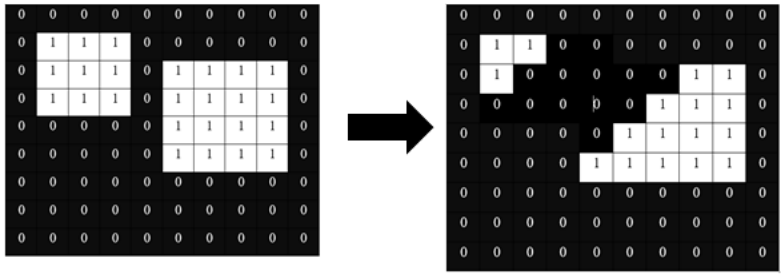
$$A \cdot B = (A \oplus B) \ominus B \tag{2.3}$$

$$A \circ B = (A \ominus B) \oplus B \tag{2.4}$$

dimana A adalah citra asli dan B adalah *structuring element* atau bisa disebut dengan kernel. Kernel merupakan matriks operator yang dapat berbentuk garis, persegi, *disk*, *diamond*, dan lain-lain, dimana elemen dari kernel dapat bernilai 1, 0, serta *don't care*. Nilai *don't care* dilambangkan dengan nilai elemen dikosongkan atau diberi tanda silang. Ilustrasi operasi dasar dilasi dan erosi ditunjukkan pada Gambar 2.3 dan 2.4. Dilasi merupakan teknik untuk memperbesar objek citra biner dengan mmenjadikan titik latar (0) yang bertetangga dengan titik objek (1) menjadi titik objek (1). Sementara Erosi merupakan teknik yang bertujuan untuk memperkecil atau mengikis tepi objek dengan cara menjadikan titik objek (1) yang bertetangga dengan titik latar (0) menjadi titik latar (0). Operasi *closing* merupakan operasi dilasi yang diikuti dengan operasi erosi, sementara operasi *opening* merupakan operasi erosi diikuti dengan operasi dilasi.



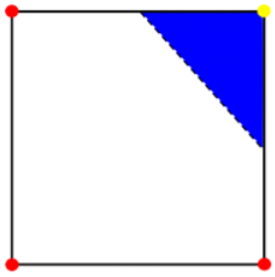
Gambar 2. 3 Ilustrasi Operasi Dilasi



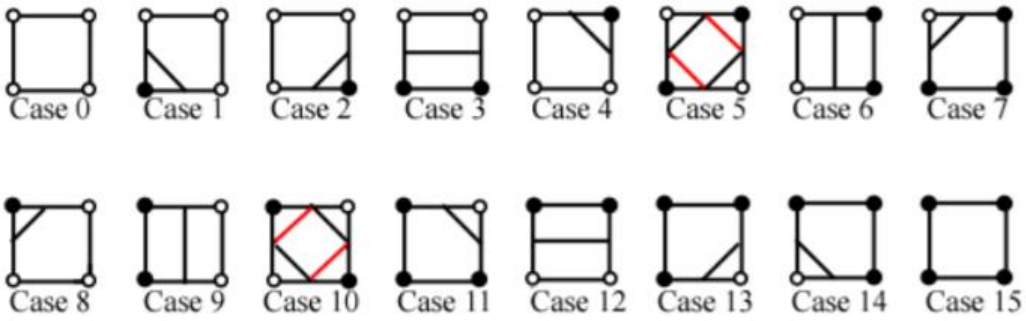
Gambar 2. 4 Ilustrasi Operasi Erosi

2.3.4 *Marching squares algorithm*

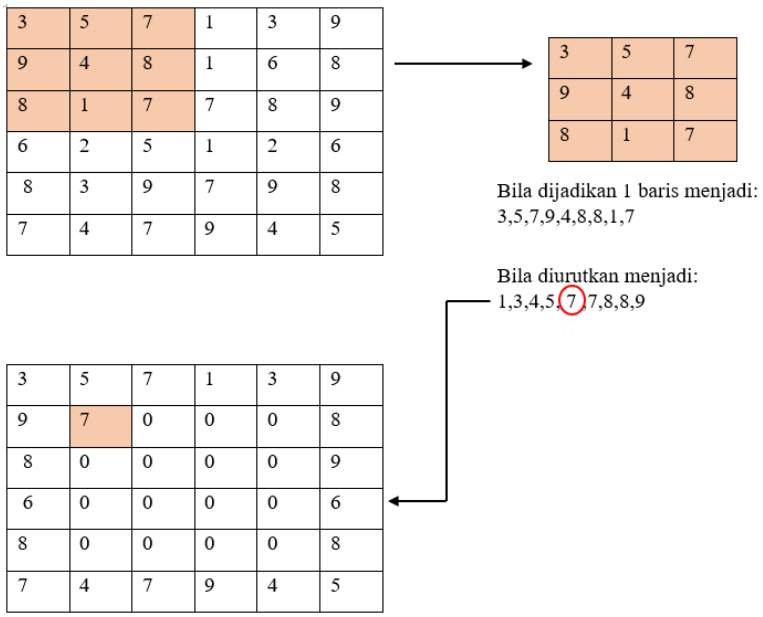
Marching squares algorithm merupakan bagian dari *marching cubes algorithm* yang terbatas pada ruang dua dimensi. Setiap ujung kisi pada *marching squares* terdiri dari 4 titik seperti yang tertera pada Gambar 2.5. Titik merah merupakan bagian luar dari *isoline* sementara titik kuning merupakan bagian dalam dari *isoline*. *Isoline* sendiri merupakan garis-garis yang diikuti oleh level data tunggal atau biasa disebut dengan *isovalue*. Keempat titik pada kisi tersebut yang akan menyediakan sebanyak 2^4=16 kemungkinan konfigurasi kontur seperti yang tertera pada Gambar 2.6. Dari keenam belas kemungkinan tersebut, terdapat 2 konfigurasi kontur yang tergolong ‘ambigu’ yaitu pada konfigurasi nomor 5 dan nomor 10 (pada Gambar 2.6). Keambiguitas ini dapat menyebabkan perbedaan terhadap garis kontur yang dihasilkan. Keambiguitasan tersebut dapat diselesaikan dengan menggunakan nilai data rata-rata pada setiap pusat kisi. [19]



Gambar 2. 5 Kisi pada *marching squares* [19]



Gambar 2. 6 Konfigurasi kontur pada *marching squares* [19]



Gambar 2. 7 Ilustrasi median filter [20]

2.3.5 Median Filter

Median filter merupakan salah satu filter statistik atau filter spasial nonlinear yang responnya didasarkan pada urutan piksel. Pada pengolahan citra digital, median filter digunakan untuk menghilangkan noise pada citra sehingga citra semakin jelas. Biasanya median filter digunakan untuk menghilangkan noise berjenis *salt and pepper*. Median filter biasanya memiliki ukuran kernel ganjil untuk memudahkan pemberian poros tengah sehingga lebih mudah dalam melakukan pengolahan citra.

Pemrosesan median filter ini dilakukan dengan cara mencari nilai tengah dari nilai piksel tetangga yang mempengaruhi piksel tengah. Teknik ini bekerja dengan cara mengisi nilai dari setiap piksel dengan nilai median tetangganya. Proses pemilihan median ini diawali terlebih dahulu dengan mengurutkan nilai-nilai piksel tetangga, baru kemudian dipilih nilai tengahnya. Pengurutan akan menghasilkan

nilai dari yang terkecil sampai nilai yang terbesar sesuai dengan P(1) < P(2) < P(3) < P(n), sedangkan nilai m sesuai dengan rumus $m = \frac{n+1}{2}$ dimana n bernilai ganjil. Ilustrasi penerapan median filter ditunjukkan pada Gambar 2.7 dengan menggunakan ukuran kernel 3 x 3. Dari gambar tersebut, elemen-elemen paling tepi dari matriks hasil median filter sama dengan elemen tepi dari matriks original karena median filter tidak akan dan tidak mungkin memproses bagian tepi tersebut. Pemfilteran diperlakukan untuk semua piksel. Jadi apabila proses filter pertama selesai maka proses berikutnya berlangsung pada piksel yang tepat disebelahnya. Apabila filter telah mencapai kolom terakhir dalam satu baris, maka filter akan turun satu piksel dan dimulai dari sebelah kiri. Proses tersebut akan berlangsung hingga semua nilai pada matriks median filter terisi dengan nilai tengah dari matriks original yang sesuai dengan ukuran kernel filter [20].

2.3.6 Bilateral Filter

Bilateral filter termasuk filter non-linear yang merupakan pengembangan dari gaussian filter. Pada gaussian filter, setiap piksel tunggal pada citra akan dicari piksel-piksel tetangga yang berdekatan yang kemudian dikonvolusikan dengan gaussian window. Gaussian window pada bidang 1 dimensi ditunjukkan pada Gambar 2.10. Pada gaussian window terlihat jelas bahwa intensitas bobot tertinggi terletak pada posisi tengah. Semakin jauh dari posisi tengah, nilai intensitas bobot semakin berkurang. Berdasarkan konsep tersebut, pembobotan rata-rata pada gaussian filter didasarkan pada jarak spasial antar piksel dan bukan pada nilai piksel. Oleh sebab itu meskipun terdapat dua piksel yang berdekatan dengan intensitas berbeda akan di lakukan pembobotan rata-rata secara bersama yang menyebabkan tepi objek dan tekstur citra menjadi kabur.

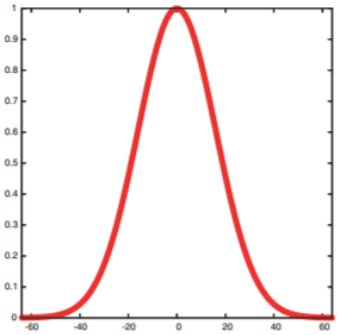
Berbeda dengan gaussian filter, pembobotan rata-rata pada bilateral filter didasarkan pada jarak spasial antar piksel dan intensitas piksel. Sebagai contoh, apabila terdapat dua piksel yang berdekatan dengan intensitas yang berbeda, maka pembobotan rata-rata hanya dilakukan pada salah satu intensitas piksel. Sehingga selain memiliki kemampuan untuk menghaluskan citra, bilateral filter juga memiliki kemampuan untuk mempertahankan tepi objek serta tekstur pada citra. Berdasarkan konsep tersebut, bilateral filter disebut sebagai *edge-preserving filter*. Bilateral filter didefinisikan sebagai

$$I^{filtered}(x) = \frac{1}{W_p} \sum_{x_i \epsilon \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) \tag{2.5}$$

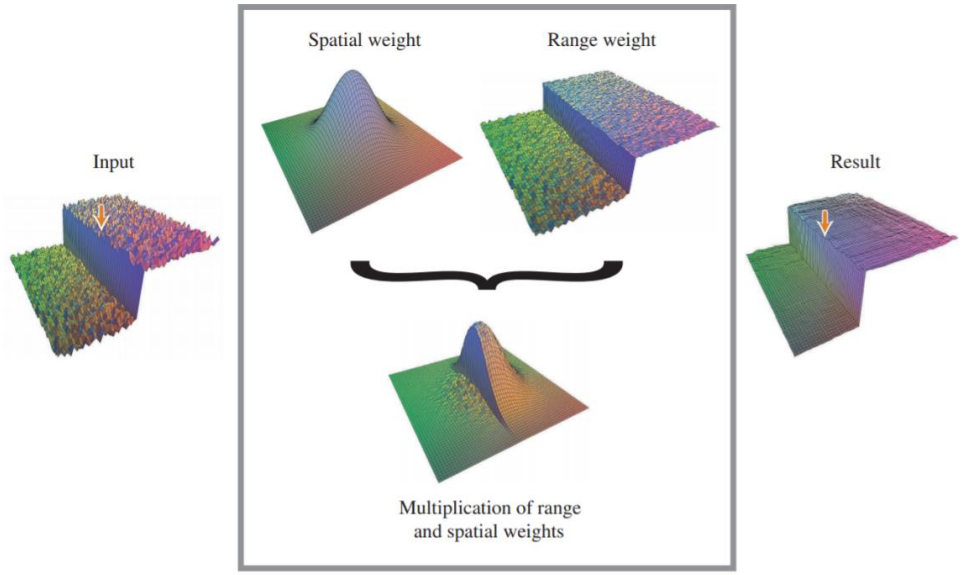
dan bentuk normalisasi W_p didefinisikan sebagai

$$W_p = \sum_{x_i \epsilon \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|) \tag{2.6}$$

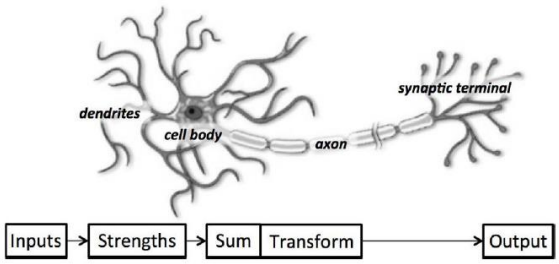
dimana $I^{filtered}$ merupakan hasil citra terfilter, I merupakan citra original, x merupakan koordinat piksel yang akan difilter, Ω merupakan window yang berpusat di x sehingga $x_i \epsilon \Omega$ merupakan piksel lain yang berada di window, f_r merupakan *range kernel*, dan g_s merupakan *spatial kernel*. *Range kernel* melakukan pembobotan berdasarkan perbedaan intensitas sementara *spatial kernel* melakukan pembobotan berdasarkan jarak spasial. *Range kernel* dan *spatial kernel* dapat diasumsikan sebagai fungsi gaussian. Ilustrasi bilateral filter ditunjukkan pada Gambar 2.11. [21]



Gambar 2. 8 Gaussian window 1D [21]



Gambar 2. 9 Ilustrasi Bilateral Filter [21]



Gambar 2. 10 Struktur Neuron pada Otak Manusia [22]

2.4 Neuron

Unit dasar otak manusia adalah neuron. Neuron digunakan untuk menerima informasi dari neuron lain, memproses informasi ini dengan cara yang unik, dan mengirimkan hasilnya ke sel lain. Neuron menerima sinyal masukan melalui dendrit. Lalu sinyal masukan ini diperkuat atau dilemahkan (pembobotan) pada setiap koneksi secara dinamis berdasarkan seberapa sering digunakan. Setelah diberi bobot oleh kekuatan koneksi masing-masing, sinyal masukan dijumlahkan bersama dalam tubuh sel. Jumlah ini kemudian diubah menjadi sinyal baru yang disebarkan sepanjang akson sel dan dikirim ke neuron lain [22]. Dari konsep neuron pada otak manusia inilah, jaringan saraf tiruan dibuat. Struktur neuron otak manusia dapat dilihat pada gambar Gambar 2.13. Neuron atau perceptron pada jaringan saraf tiruan meniru apa yang dilakukan neuron pada otak manusia. Neuron pada jaringan saraf tiruan mengambil beberapa masukan dan melakukan penjumlahan berbobot menggunakan *weight* untuk menghasilkan keluaran [23]. *Weight* pada neuron ditentukan selama proses pelatihan dan didasarkan pada data pelatihan. Struktur neuron pada jaringan saraf tiruan ditunjukkan pada Gambar 2.11.

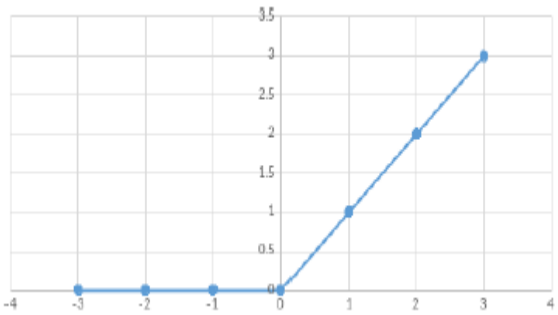
2.5 Fungsi Aktivasi

Untuk menentukan nilai keluaran, setiap unit dari jst menggunakan fungsi aktivasi. Fungsi aktivasi menentukan nilai keluaran dari tiap unit pada satu level aktivasi tertentu berdasarkan pengombinasi linier. Banyak fungsi aktivasi yang dapat digunakan seperti fungsi goniometri dan hiperboliknya, fungsi unit step, impuls, sigmoid dan juga *rectified linear unit (*ReLu) [24].

2.5.1 Rectified Linear Unit (ReLu)

Relu atau *rectified linear unit* menjadi salah satu fungsi aktivasi yang paling banyak digunakan dalam model *deep learning*. Sistem kerja ReLu cukup sederhana, yaitu mengembalikan nilai 0 saat menerima masukan bernilai negatif, tetapi mengembalikan sembarang nilai positif yang masuk tanpa mengubah nilainya. Cara kerja fungsi ReLu ditunjukkan oleh Gambar 2.14. Secara umum, ReLu dapat direpresentasikan menggunakan Persamaan 2.7 [25].

$$f(x) = \max(0, x) \tag{2.7}$$



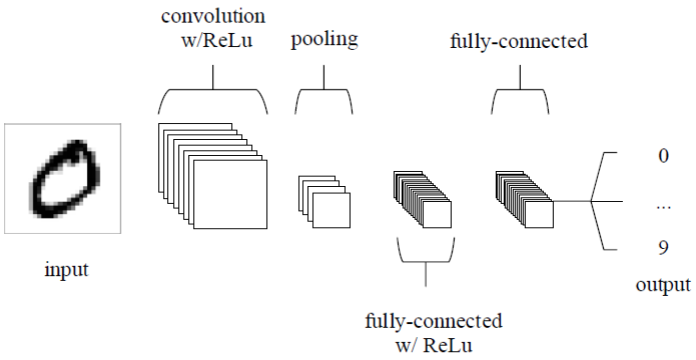
Gambar 2. 11 Fungsi aktivasi ReLu [25]

2.6 Adaptive Moment Estimation

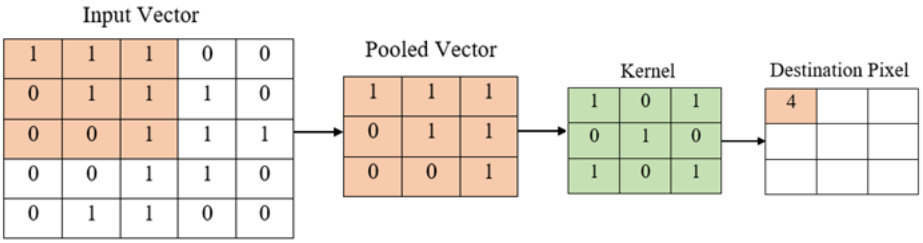
Adaptive Moment Estimation (ADAM) merupakan algoritma yang digunakan untuk optimisasi gradien pada *neural netowork* berdasarkan data latih. Metode inilebih mudah diimplementasikan, efisien secara komputasi, memerlukan kebutuhan memori yang kecil, dan sesuai untuk masalah yang memiliki banyak data maupun parameter [26]. ADAM berbeda dengan algoritma *Stochastic Gradient Descent* (SGD) yang memiliki laju pembelajaran (*learning rate*) yang tidak berubah setiap pembaruan bobot (*weights update*). Algoritma ADAM menghitung rata-rata bergerak eksponensial (*exponential moving average*) dari gradien dan gradien kuadrat. Parameter beta 1 dan beta 2 mengendalikan laju pengurangan (*decay rate*) dari rata-rata bergerak.

2.7 *Convolutional Neural Network*

Convolutional Neural Network biasa disebut dengan CNN merupakan pengembangan dari *feedforward* ANN yang diaplikasikan untuk mengolah citra digital [27]. Komputer mengenali citra digital dalam bentuk *array* dari nilai-nilai pikselnya. Pada citra digital dengan resolusi piksel 128×128, akan terdapat *array* tiga dimensi berukuran $128 \times 128 \times 3$ dimana masing-masing suku merepresentasikan tinggi, lebar, dan jumlah *channel* pada citra. Pada kasus



Gambar 2. 12 Arsitektur CNN [27]

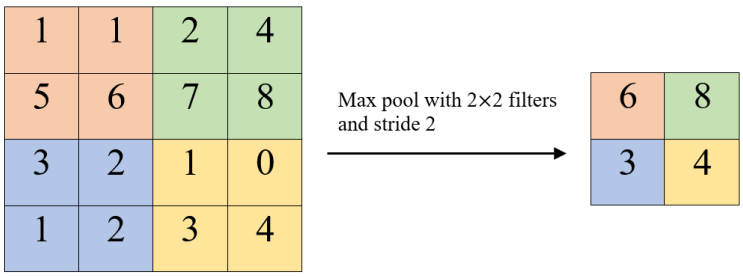


Gambar 2. 13 Operasi Konvolusi [27]

klasifikasi data citra digital, *feedforward* ANN seperti *multilayer perceptron* (MLP) kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik [28]. Konsep CNN mirip dengan MLP, hanya saja pada MLP digunakan input satu dimensi sedangkan pada CNN input berupa data dua dimensi. Pada CNN neuron dari *input layer* akan dipropagasikan ke neuron pada *layer* selanjutnya. Hubungan antar neuron pada dua *layer* memiliki parameter bobot empat dimensi atau yang biasa dikenal dengan *weight* yang merupakan kumpulan kernel konvolusi. Dimensi bobot pada CNN adalah $neuron\ input \times neuron\ output \times tinggi \times lebar$**.** Disetiap data *input* pada *layer* akan dilakukan operasi konvolusi dengan nilai *weight* yang ada. Hasil komputasi kemudian ditransformasi menggunakan operasi *nonlinear* yang disebut dengan fungsi aktivasi [31]. Secara umum, arsitektur CNN terdiri dari 3 *layer* yang dapat dilihat pada Gambar 2.15 yaitu:

1. *Convolutional layer*

Sesuai dengan namanya, *convolutional layer* memegang peranan penting dalam kinerja CNN. Proses yang terjadi pada bagian ini adalah melakukan *encoding* dari sebuah citra menjadi fitur berupa angka yang merepresentasikan citra tersebut menggunakan operasi konvolusi. Konvolusi pada citra digital dilakukan dengan mencuplik piksel pada citra input dengan luasan tertentu pada semua *offset* yang mungkin kemudian melakukan operasi *dot product* dengan sebuah filter yang luasnya sama dengan citra yang dicuplik sehingga menghasilkan sebuah output yang biasa disebut dengan *activation map.* Jumlah *layer* pada filter bergantung pada jumlah *channel* pada citra input dan output dari *convolutional layer* memiliki kedalaman yang sama dengan jumlah filter yang diset. Sebagai contoh apabila citra input memiliki dimensi $(7 \times 7 \times 3)$ dimana masing-masing suku merepresentasikan tinggi, lebar, dan jumlah channel yaitu *red, green, blue* (RGB), maka jumlah *layer* pada filter yang akan dikonvolusikan dengan citra input juga berjumlah tiga buah misal filter $(3 \times 3 \times 3)$. Ukuran tinggi dan lebar filter tidak dapat dipastikan dan bergantung pada implementasinya. Apabila kita mengingkinkan 2 buah filter yaitu $W0\ (3 \times 3 \times 3)$ dan $W1\ (3 \times 3 \times 3)$ maka output dari *convolutional layer* memiliki kedalaman 2 buah sesuai dengan jumlah



Gambar 2. 14 *Maxpooling* [28]

filter yang digunakan Out $(3 \times 3 \times 2)$. Ilustrasi operasi konvolusi pada citra 6×6 piksel dengan filter 3×3 ditunjukkan pada Gambar 2.16.

Convolutional layer juga mampu mengurangi kompleksitas model secara signifikan melalui optimalisasi output dengan menggunakan tiga parameter yaitu *depth*, *stride*, dan *zero padding* [29]. ***Depth*** merupakan parameter dimana kedalaman volume output yang dihasilkan dari *convolutional layer* dapat secara manual diatur melalui jumlah *neuron.* ***Stride*** merupakan parameter yang menentukan jumlah pergeseran filter. Apabila nilai *stride* adalah 2, maka *convolutional filter* akan bergeser sebanyak 2 piksel secara horizontal kemudian vertikal. Semakin kecil nilai *stride* maka semakin detail informasi fitur yang didapat, namun membutuhkan komputasi yang lebih besar dibanding dengan *stride* yang besar. Tetapi dengan penggunaan *stride* yang kecil tidak selalu akan menghasilkan performa yang baik. ***Zero padding*** merupakan parameter yang menentukan jumlah piksel berisi nilai 0 yang akan ditambah disetiap sisi dari input. Dimensi output yang dihasilkan dari *convolutional layer* selalu lebih kecil (kecuali pada saat penggunaan *stride* bernilai 1), dengan adanya *zero padding* dimensi output dapat diatur sedemikian rupa agar tetap sama dengan dimensi input. Sehingga peluang melakukan *convolutional layer* yang dalam/*deep* lebih besar dan menyebabkan lebih banyak fitur yang berhasil diekstrak. Selain itu dengan adanya *zero padding* dapat meningkatkan performa dari model karena *convolutional layer* berfokus pada informasi diantara *zero padding* tersebut. Untuk menghitung dimensi dari *feature map* ditunjukkan pada Persamaan 2.8,

$$output = \frac{(V - R) + 2Z}{S} + 1 \tag{2.8}$$

dimana *V* adalah volume input (panjang/tinggi), R adalah *receptive field size* (panjang/tinggi filter), *Z* adalah *Zero padding, S* adalah *Stride.*

2. *Pooling layer*

Pooling layer bertujuan untuk mereduksi dimensionalitas dari *feauture map* (*downsampling*) sehingga mengurangi kompleksitas perhitungan model. Dalam pengolahan citra digital, *pooling* berguna untuk menurunkan variansi fitur. Jenis *pooling layer* yang sering digunakan dalam CNN adalah *max pooling.* Cara kerja *max pooling* adalah membagi matriks dari hasil output *convolutional layer* menjadi

beberapa bagian. Kemudian dari setiap bagian diambil nilai piksel maksimal untuk menyusun matriks yang telah direduksi. Sebagai contoh pada Gambar 2.17 apabila input citra berukuran 4×4 dan digunakan kernel berukuran 2×2 dan *stride* sebesar 2 maka hasil *activation map* berhasil direduksi sebanyak 25%. Apabila *stride* dan dimensi kernel tidak sesuai maka akan terjadi *overlapping pooling. Overlapping pooling* dapat menurunkan performansi model.

3. *Fully connected layer*

Fully connected layer akan melakukan tugas yang sama seperti MLP. *Layer* ini berfungsi untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasi secara linear. Penggunaan ReLu dapat diaplikasikan pada *layer* ini untuk meningkatkan performansi. Proses *training* pada CNN untuk mendapatkan fitur serta *update weight* dari setiap kelas adalah dengan menggunakan *backpropagation.*

2.8 Uji Performansi

Pengujian performansi dilakukan untuk menguji hasil rancangan secara kuantitatif. Pengujian rancangan algoritma pra pengolahan berupa reduksi *noise* dilakukan menggunakan *Peak Signal to Noise Ratio* (PSNR), sementara pengujian untuk rancangan CNN dilakukan menggunakan *confusion matrix*.

2.8.1 Peak Signal to Noise Ratio (PSNR)

Pengujian menggunakan PSNR dilakukan untuk menentukan kualitas citra setelah dilakukan operasi pengurangan derau dimana citra hasil akan dibandingkan dengan citra asli. Semakin besar nilai PSNR maka semakin baik algoritma yang ditetapkan dalam mengurangi derau. PSNR diukur dalam satuan desibel (dB). Untuk menentukan PSNR, terlebih dahulu harus ditentukan nilai rata-rata kuadrat dari error (MSE- *Mean Square Error*). Perhitungan MSE tertera pada Persamaan 2.9

$$MSE = \frac{1}{mn}\sum_{i}^{j}\sum_{j}^{n}\|I(i,j) - K(i,j)\|^2 \tag{2.9}$$

dimana m dan n merupakan panjang dan lebar citra dalam satuan piksel, i, j merupakan koordinat masing-masing piksel, I merupakan nilai citra hasil, dan K merupakan nilai citra asli.

Sementara nilai PSNR dihitung dari kuadrat nilai maksimum sinyal dibagi dengan MSE. Persamaan 2.10 merupakan perhitungan PSNR dalam satuan dB

$$PSNR = 20\ \log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{2.10}$$

dimana MAX merupakan nilai maksimum piksel dan MSE merupakan nilai MSE.

2.8.2 Confusion Matrix

Ketika membangun sebuah model klasifikasi, pertanyaan yang muncul adalah bagaimana mengetahui seberapa baik model tersebut. Mengevaluasi model klasifikasi dilakukan dengan mencaritahu seberapa baik hasil prediksi dari model tersebut. Penjelasan variabel ada pada *confusion matrix* pada Tabel 2.1.

Tabel 2. 1 *Confusion matrix*

		True Values	
		True	False
Prediction	True	TP	FP
	False	FN	TN

Keterangan:

1. *True Positive* (TP) pada kelas x dan dikenali oleh keluaran program sebagai kelas x.
2. *True Negative* (TN) pada kelas bukan x, pada keluaran program tidak dikenali sebagai kelas x.
3. *False Negative* (FN) pada kelas x, pada keluaran program tidak dikenali sebagai kelas x.
4. *False Positive* (FP) pada kelas bukan x, pada keluaran program dikenali sebagai kelas x.

Masing-masing indikator dapat dituliskan dalam persamaan matematika seperti pada Persamaan 2.11 hingga 2.13 [30].

$$akurasi = \frac{TP + TN}{TP + TN + FN + FP} \tag{2.11}$$

$$presisi = \frac{TP}{TP + FP} \tag{2.12}$$

$$sensitivitas = \frac{TP}{TP + FN} \tag{2.13}$$

$$spesifitas = \frac{TN}{TN + FP} \tag{2.14}$$

2.9 Python

Python adalah bahasa pemograman yang popular. *Python* sering dimanfaatkan dalam pengembangan web, perangkat lunak, penelitian, dan *system scripting*. Python dapat digunakan untuk menangani data besar dan melakukan operasi matematika yang kompleks. Python bekerja di berbagai *platform* seperti Windows, Mac, Linux, Raspberry Pi, dan lain-lain. Python dirancang untuk mudah dibaca, yaitu memiliki sintaks yang sederhana dan menggunakan bahasa inggris. [31]

2.10 Scikit-image

Scikit-image atau *skimage* merupakan pustaka pemrosesan gambar dan *computer vision* berbasis *open source* untuk bahasa python. *Skimage* mencakup algoritma untuk segmentasi, transformasi geometrik, manipulasi ruang warna, analisis, pemfilteran, morfologi, deteksi fitur, dan lain sebagainya. [32]

2.11 Tensorflow

Tensorflow merupakan pustaka perangkat lunak sumber terbuka yang dirilis pada 2015 oleh Google untuk memudahkan pengembang merancang, membuat, dan melatih model *deep learning*. Meskipun *Tensorflow* hanya satu dari beberapa opsi yang tersedia untuk pengembang, *Tensorflow* memiliki fitur yang lengkap dan mudah untuk digunakan. *Tensorflow* adalah pustaka Python yang memungkinkan pengguna untuk merepresentasikan perhitungan sebagai grafik aliran data. *Node* dalam grafik ini merepresentasikan operasi matematika, sedangkan *edge* mewakili data yang dikomunikasikan dari satu *node* ke *node* lainnya. Data dalam *Tensorflow* direpresentasikan sebagai tensor, yang merupakan array multidimensi (mewakili vektor dengan tensor 1D, matriks dengan tensor 2D, dll.). *Tensorflow* juga menyertakan Tensorboard, alat untuk visualisasi data. [33]

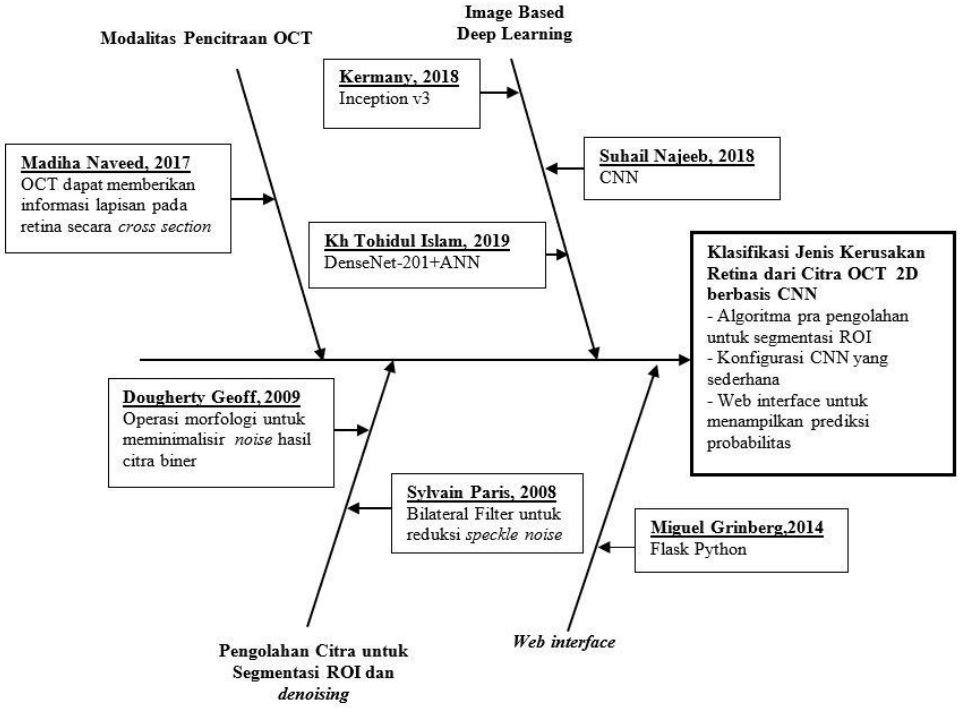
2.12 Keras

Keras merupakan pustaka perangkat lunak sumber terbuka yang berisi fungsi-fungsi pemrograman terkait tentang jaringan syaraf tiruan dan turunannya. *Keras* dikembangkan oleh *François Chollet* dan versi pertamanya dirilis pada Maret 2015. *Keras* adalah API yang memudahkan dalam perancangan dan pemrograman jaringan syaraf tiruan. *Keras* tidak dapat berjalan sendiri dan secara *default* menggunakan *TensorFlow* sebagai *backend*-nya. [34]

2.13 Rasional

Secara keseluruhan rasional dari tugas akhir ini dapat dilihat dalam diagram *fishbone* pada Gambar 2.18. Ide yang diusulkan merupakan suatu sistem untuk mengklasifikasi jenis kerusakan pada retina. Modalitas pencitraan yang dipilih adalah OCT karena modalitas tersebut *non-invasive* serta mampu memberikan informasi lapisan retina secara *cross secion* [4]. Sistem yang diusulkan dilengkapi dengan algoritma pra pengolahan untuk mensegmentasi area ROI dan reduksi *speckle noise*. Teknik dasar pengolahan citra yang digunakan berupa operasi biner, operasi morfologi [18], dan algoritma *marching square* [19] untuk mendeteksi kontur citra, serta bilateral filter untuk reduksi *speckle noise* [21]. Setelah tahap pra pengolahan, maka dilakukan pelatihan pada citra menggunakan model CNN yang telah dirancang sebelumnya. Berbeda dengan penelitian pada [9],[10], dan [30], model CNN yang dirancang pada tugas akhir ini memiliki konfigurasi yang lebih sederhana. Setelah didapatkan hasil pelatihan yang baik, maka model akan disimpan yang kemudian digunakan untuk proses pembuatan antar muka berbasis web menggunakan *Flask Python* [35]. Antar muka yang dibuat akan menampilkan

prediksi probabilitas dari citra yang diinputkan. Pada penelitian ini juga dilakukan pelatihan CNN pada beberapa kondisi untuk melihat pengaruh ada tidaknya proses segmentasi dan *balancing data*.



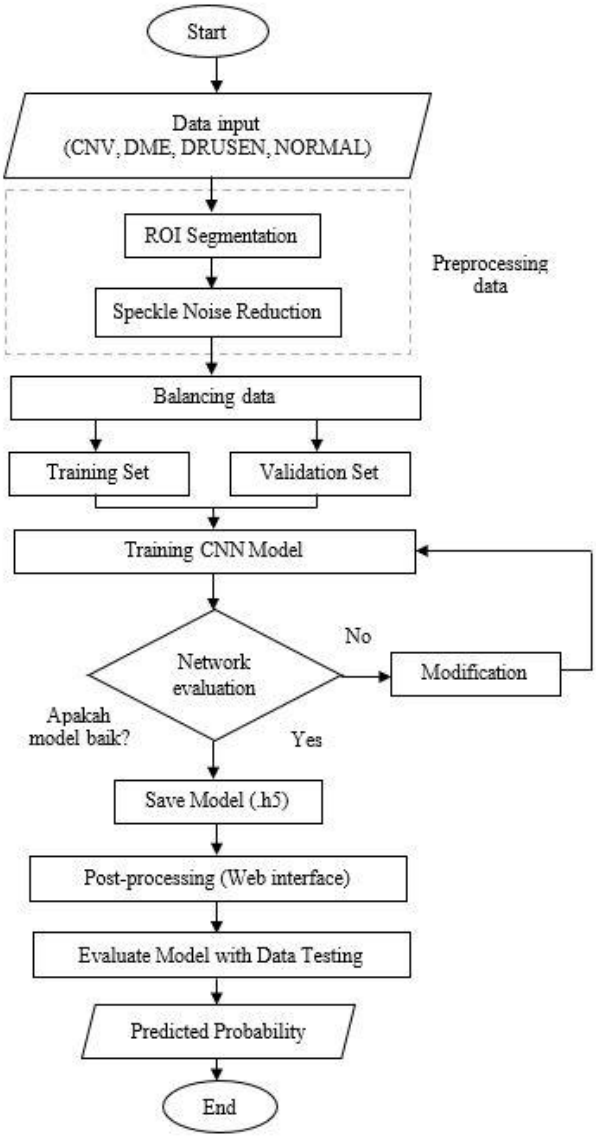
Gambar 2. 15 Diagram *Fishbone* Penelitian

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan tentang metodologi mengenai perancangan sistem yang akan dikerjakan. Perancangan tersebut meliputi persiapan dataset, perancangan pra pengolahan, perancangan *balancing data*, perancangan CNN, serta perancangan uji coba model. Semua sistem tersebut akan dibahas dalam bagian bab ini.

3.1 Desain Sistem

Desain rancangan sistem secara keseluruhan dijelaskan melalui diagram alir pada Gambar 3.1. Alur pengerjaan dimulai dari pemilihan dataset. Dataset yang



Gambar 3. 1 Diagram alir sistem

telah dipilih terdiri dari 4 kelas yaitu: CNV, DME, DRUSEN, dan NORMAL. Selanjutnya dilakukan proses pra pengolahan citra berupa segmentasi citra dan reduksi *speckle noise.* Tahapan pra pengolahan dijelaskan lebih detail pada bagian 3.3. Dataset yang diperoleh pada penelitian sebelumnya terdiri dari dua bagian yaitu data untuk *training* dan data untuk *testing*. Berbeda dengan data *testing*, pada data *training* persebaran jumlah data pada setiap kelas tidak seimbang, oleh sebab itu dibutuhkan tahapan *balancing data* untuk menyeimbangkan persebaran data tiap kelas. Data *training* yang sudah melewati proses *balancing data* kemudian dibagi menjadi dua bagian yaitu *training-set* dan *validation-set.* Rancangan jaringan CNN yang dijelaskan pada bagian 3.5 diaplikasikan dengan data *training*-set agar komputer dapat belajar mengenali citra input. Setelah pembelajaran selesai dilakukan pengujian terhadap data *validation-set*. Parameter pengujian yang digunakan berupa akurasi, presisi, sensitivitas, dan spesifisitas. Apabila hasil pengujian diperoleh hasil yang kurang baik (kurang dari 80%), maka dilakukan modifikasi ulang pada model CNN berupa pemilihan hyperparameter. Sementara apabila diperoleh hasil yang baik, maka model CNN tersebut akan digunakan untuk pengujian pada data *testing*. Proses pengujian pada data *testing* dilakukan menggunakan *web interface*. Output pengujian berupa hasil klasifikasi beserta nilai probabilitas dari masing-masing kelas yang ditampilkan dalam bentuk bar grafik.

3.2 Dataset

Pada penelitian ini data yang digunakan merupakan data 2 dimensi yang didapatkan dari *open source* dataset hasil modalitas OCT. Dataset ini merupakan dataset yang sama dengan penelitian dari Kermany dkk [10] yang diperoleh dari *Shiley Eye Institute of the University of California San Diego, the California Retinal Research Foundation, Medical Center Opthalmology Associates, the Sanghai First People's Hospital, dan Beijing Tongren Eye Center* dalam kurun waktu 1 Juli 2013 hingga 1 Maret 2017. Tidak ada keriteria khusus yang ditetapkan peneliti sebelumnya baik berupa umur, jenis kelamin, maupun ras. Dataset yang diperoleh terdiri dari dua bagian yaitu data training sebanyak 84.276 dan data testing sebanyak 968. Dari data training tersebut kemudian dibagi menjadi dua bagian yaitu *training-set* dan *validation-set* dengan rasio 4:1. Distribusi dataset yang akan digunakan tertera pada Tabel 3.1.

Tabel 3. 1 Persebaran Jumlah Dataset tiap Kelas

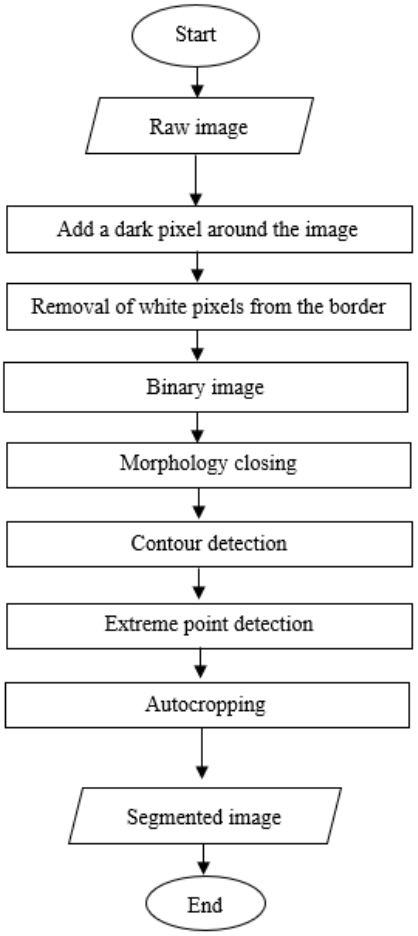
Input Kelas	**Data Training**		**Data Testing**
	Training-Set	**Validation-Set**	
CNV	29.764	7.441	242
DME	9.079	2.269	242
DRUSEN	6.893	1.723	242
NORMAL	21.052	5.263	242

3.3 Perancangan Pra Pengolahan Citra

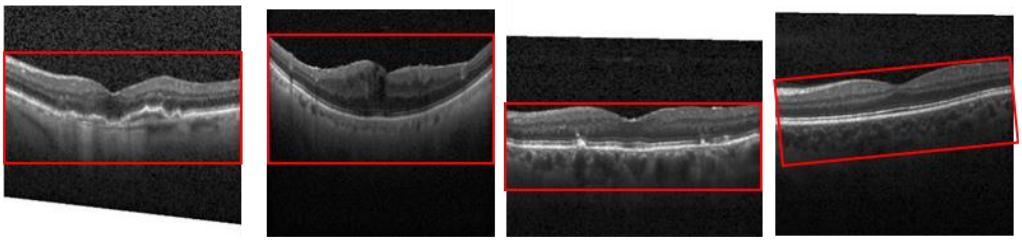
3.3.1 Perancangan Segmentasi ROI

Diagram alir segementasi ROI pada citra dijelaskan pada Gambar 3.2. Seperti yang terlihat pada Gambar 3.3, untuk setiap jenis penyakit pada retina, terdapat area spesifik yang biasa disebut dengan ROI (*Region of Interest*), dimana penekanan dapat diberikan untuk mendeteksi ketidaknormalan retina. Beberapa teknik dasar pra pengolahan citra akan diterapkan untuk mensegmentasi area ROI tersebut. Pertama, setiap citra akan diberi tambahan border berupa piksel berwarna hitam disekeliling citra. Hal ini bertujuan agar ROI menempati posisi tengah pada saat operasi *binary thresholding*. Apabila tidak ditempatkan pada posisi tengah, maka algoritma deteksi kontur yang akan digunakan tidak dapat mendeteksi keseluruhan kontur citra.

Pada beberapa dataset, terdapat area dengan piksel berwarna putih disekitar tepian citra. Karena pada step berikutnya digunakan operasi *binary thresholding*, maka batas putih ini akan menganggu proses segmentasi. Oleh karena itu, batas



Gambar 3. 2 Diagram Alir Segmentasi Citra



Gambar 3. 3 *Region of Interest* (a) CNV (b) DME (c) DRUSEN (d) NORMAL

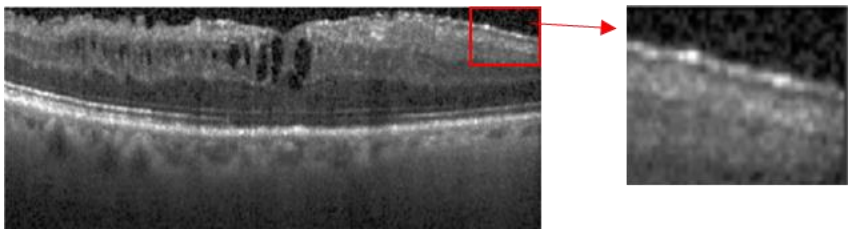
putih tersebut akan dihilangkan dengan cara mengganti bagian piksel putih tersebut dengan piksel berwarna hitam.

Selanjutnya dilakukan operasi *binary thresholding* dengan cara menetapkan suatu nilai ambang batas. Operasi ini akan menghasilkan citra biner dengan *dark pixel* menempati area ROI dan *white pixel* menempati area *background*.

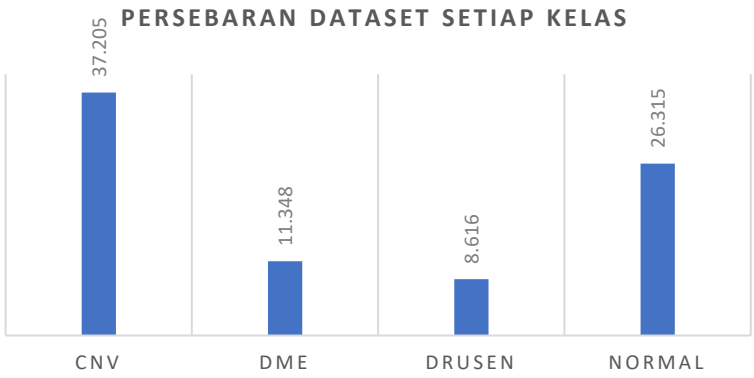
Operasi *binary thresholding* pada langkah sebelumnya meninggalkan *noise* pada gambar yang akan diminimalisir menggunakan operasi morfologi. Operasi morfologi yang digunakan adalah operasi *closing* yang didefinisikan seperti pada Persamaan 2.3, dimana A adalah citra input dan B adalah *structuring element.* Operasi *closing* sendiri dilaksanakan dengan melakukan operasi dilasi terlebih dahulu dan kemudian diikuti dengan operasi erosi. Operasi morfologi ini mampu meminimalisir *noise* dan menghasilkan citra biner yang bersih.

Tahapan selanjutnya adalah menggunakan metode *marching squares* dengan bantuan pustaka skimage untuk mendeteksi kontur. Setiap blok piksel berukuran 2×2 dalam citra biner akan membentuk sel kontur, sehingga seluruh bagian citra biner direpresentasikan sebagai sebuah kisi yang terdiri dari beberapa sel kontur. Masing-masing sel kontur terdiri dari 4-bit nilai piksel yang akan dikomposisikan sebagai indeks biner. Indeks biner diperoleh dengan cara membaca nilai bit dari posisi kiri atas kemudian digeser secara *clockwise* dan berakhir pada posisi kiri bawah. Sehingga nilai bit piksel pada kiri atas merupakan MSB (*most significant bit*) sementara nilai bit piksel pada posisi kiri bawah merupakan LSB (*least significant bit*). Indeks 4-bit yang dihasilkan dapat memiliki 16 nilai yang mungkin dalam kisaran 0-15. Nilai tersebut kemudian digunakan untuk mendapatkan kontur dengan cara mengakses *table contour lines* yang sudah dibuat sebelumnya. Setelah berhasil mendapatkan kontur, maka tahapan selanjutnya adalah mencari area kontur terbesar.

Kontur yang didapat pada tahap sebelumnya kemudian digunakan sebagai acuan untuk menentukan koordinat minimum dan maksimum sepanjang sumbu vertikal dan horizontal. Koordinat tersebut kemudian digunakan untuk proses *autocropping* area ROI.



Gambar 3. 5 *Speckle noise* pada Citra



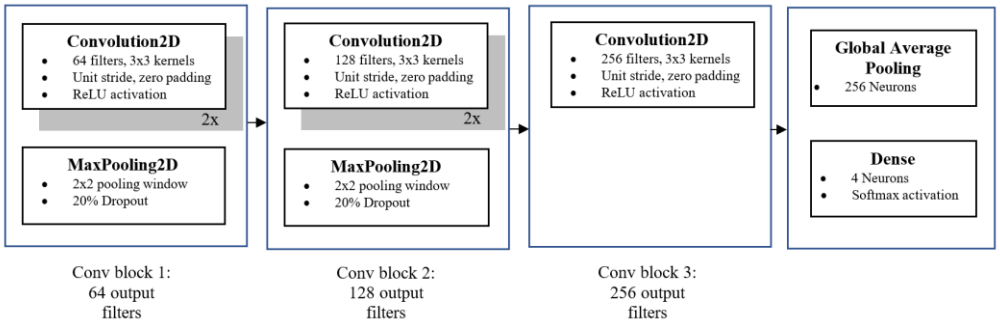
Gambar 3. 4 Grafik persebaran dataset setiap kelas

3.3.2 Perancangan *Image Enhancement*

Rancangan *image enhancement* atau perbaikan citra dilakukan untuk mereduksi *speckle noise*. Seperti yang dijelaskan sebelumnya, data mentah pada hasil citra OCT masih mengandung *speckle noise*. *Speckle noise* pada citra OCT terlihat sebagai granular atau butiran-butiran hitam putih seperti yang ditunjukkan pada Gambar 3.4. Rancangan reduksi *speckle noise* dilakukan menggunakan median filter dan bilateral filter. Kedua filter tersebut dipilih karena selain memiliki kemampuan untuk menghaluskan citra juga memiliki kemampuan untuk mempertahankan dan menajanmkan tekstur dan tepi citra.

3.4 Perancangan *Balancing Data*

Seperti yang tertera dalam grafik pada Gambar 3.5, bahwa persebaran data training antar kelas tidak seimbang atau tidak sama rata yang akan berakibat pada terjadinya misklasifikasi. Oleh sebab itu dilakukan tahapan *balancing data* dengan dua metode yang berbeda yaitu *weight balancing* dan *random undersampling*. *Weight balancing* dilakukan dengan cara memberikan nilai bobot yang berbeda pada setiap kelas dimana kelas minoritas akan diberikan bobot yang lebih besar dibanding dengan kelas mayoritas. Sementara *random undersampling* dilakukan dengan cara memilih data secara random untuk di *copy* ke folder yang baru sehingga jumlah data masing-masing kelas akan mengikuti jumlah data pada kelas terendah. Kedua metode tersebut kemudian akan dibandingkan untuk melihat metode mana yang memiliki hasil yang lebih baik.



Gambar 3. 6 Rancangan CNN

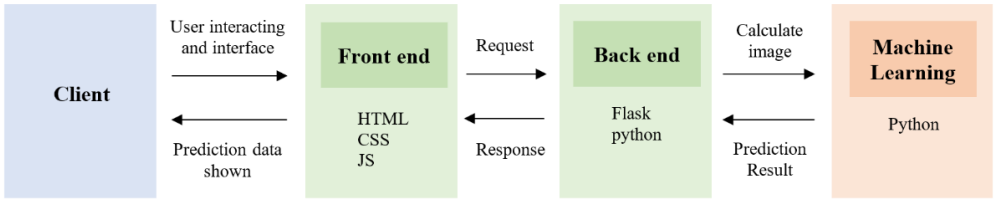
3.5 Perancangan Arsitektur CNN

Klasifikasi jenis kerusakan retina menggunakan jaringan syaraf konvolusi atau *Convolutional Neural Network* (CNN). Dalam klasifikasi ini, jaringan syaraf konvolusi berfungsi untuk melakukan klasifikasi terhadap 4 jenis kerusakan retina. Algoritma jaringan syaraf konvolusi dipilih karena tidak perlu mendefinisikan fitur citra yang akan dijadikan masukan sistem. Jaringan syaraf konvolusi akan menentukan sendiri fitur yang akan digunakan ketika proses pelatihan berlangsung.

Rancangan model CNN yang digunakan dijelaskan pada Gambar 3.6. Rancangan terdiri dari 2 bagian yaitu lapisan untuk ekstraksi fitur dan lapisan untuk klasifikasi. Pada bagian ekstraksi fitur terdiri dari 3 blok bagian. Pada blok pertama dan kedua digunakan lapisan konvolusi ganda diikuti dengan lapisan *max pooling* berukuran 2 × 2, serta *dropout* sebesar 20% untuk mencegah *overfitting*. Sementara pada blok ketiga hanya terdiri dari lapisan konvolusi. Masing masing lapisan konvolusi pada setiap blok terdiri dari operasi konvolusi 2 dimensi menggunakan filter dengan ukuran 3 × 3 piksel serta ReLU sebagai fungsi aktivasi. ReLU akan meloloskan input positif dan mengganti nilai negatif menjadi 0. Kemudian unit dengan nilai 0 akan dinonaktifkan. Pada lapisan konvolusi juga digunakan parameter unit stride yaitu jumlah pergeseran filter sebesar 1 dan diterapkan pula *zero padding*. Stride bernilai 1 memiliki makna bahwa maka convolutional filter akan bergeser sebanyak 1 piksel secara horizontal lalu vertikal. Semakin kecil stride maka semakin detail informasi yang didapatkan dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan stride yang besar. Jumlah filter pada masing-masing blok lapisan konvolusi secara berturut-turut sebesar 64, 128, dan 256 filter. Sehingga pada *hidden layer* ketiga dihasilkan 256 fitur map 2D yang kemudian oleh lapisan *global average pooling* dikonversi menjadi bentuk vektor tunggal 1D berupa 256 node atau neuron. Selanjutnya fungsi aktivasi *softmax* diaplikasikan untuk menghasilkan probabilitas prediksi masing-masing kelas.

3.6 Perancangan Uji Coba Model

Uji coba model dilakukan untuk menguji data *testing* dengan cara membangun sebuah *web interface* dengan memanfaatkan model CNN yang sudah disimpan sebelumnya. *Web interface* ini juga sebagai langkah *post-processing*, yang mana akan menampilkan prediksi probabilitas dari sebuah citra input. Secara keseluruhan, skema perancangan *web interface* ditunjukkan pada Gambar 3.6. *Client* dalam hal ini browser akan mengirimkan sebuah citra retina yang diminta oleh user melalui *front-end*. *Front-end* dibangun menggunakan HTML (*HyperText Markup Language*), CSS (*Cascading* Style Sheets), dan *JavaScript*. *Front-end* kemudian melakukan proses *request* kepada *back-end* untuk melakukan prediksi terhadap citra tersebut. Pada *back-end*, digunakan Flask sebagai web framework. Proses prediksi dilakukan menggunakan model *machine learning* yang sudah disimpan sebelumnya. *Machine learning* tersebut kemudian memberikan hasil prediksi berupa probabilitas dalam rentang nilai 0 hingga 1 pada masing-masing kelas kepada *back-end*. Setelah *back-end* mendapatkan data prediksi tersebut, *back-end* akan mengirim ke *front-end* dalam bentuk *response*. *Front-end* kemudian memproses data yang sudah didapat menjadi sebuah tampilan yang kita lihat di browser.



Gambar 3. 7 Rancangan uji coba model

BAB IV
IMPLEMENTASI SISTEM

4.1 Lingkungan Implementasi

Spesifikasi perangkat lunak yang digunakan untuk implementasi sistem klasifikasi jenis kerusakan retina adalah sebagai berikut:

- Bahasa pemrograman Python
- Pustaka skimage
- Pustaka Keras dan Tensorflow

Implementasi sistem klasifikasi jenis kerusakan retina menggunakan bahasa pemrograman Python. Pustaka skimage merupakan pustaka yang digunakan untuk pemrosesan citra. Pustaka keras, tensorflow, dan scikit-learn digunakan untuk proses klasifikasi jenis kerusakan retina. Implementasi perangkat lunak di atas dijalankan pada sistem operasi Windows 10 64 bit, dengan spesifikasi perangkat keras yang digunakan sebagai berikut:

- Processor Intel® Core™ i7-7500U (6M Cache, up to 3.5 GHz)
- Memory RAM 8 GB
- Kartu grafis NVIDIA® GeForce® GT940MX untuk pra pengolahan

4.2 Implementasi Pengolahan Citra

4.2.1 Implementasi Segmentasi ROI

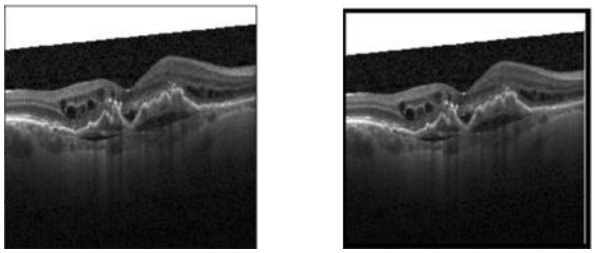
Proses segmentasi ROI dilakukan dengan mengimplementasikan diagram alir yang terdapat pada subbab 3.3. Diagram alir tersebut terdiri dari beberapa tahapan yang akan dijelaskan pada subbab 4.2.1 hingga 4.2.6.

4.2.1.1 Penambahan border hitam disekeliling citra

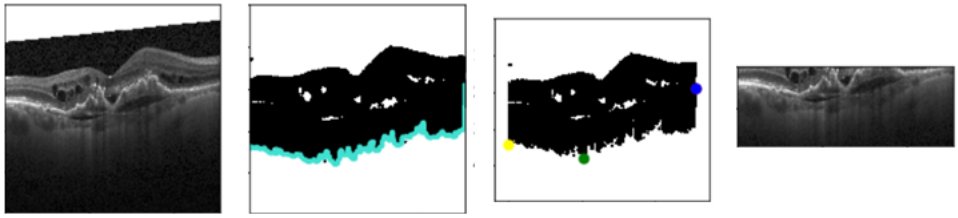
Implementasi penambahan border berwarna hitam disekeliling citra dalam bahasa python adalah sebagai berikut,

```
(row, col) = img.shape
img2 = np.zeros((row+20, col+20))
img2[10:row+10, 10:col+10] = img
```

penambahan border dilakukan dengan cara membuat array baru berisi piksel dengan nilai 0 berukuran $(p + 20, l + 20)$ dimana p dan l merupakan ukuran citra original. Kemudian citra original akan ditambahkan dengan array baru sehingga menghasilkan border hitam pada citra original sejauh 10 piksel mengelilingi citra. Hasil implementasi ditunjukkan pada Gambar 4.1. Penambahan border bertujuan agar ROI menempati posisi tengah pada saat operasi biner. Apabila tidak ditempatkan pada posisi tengah, maka algoritma deteksi kontur yang akan



Gambar 4. 2 Implementasi penambahan border hitam disekeliling citra



Gambar 4. 1 Segementasi yang gagal karena tidak ada penambahan border hitam disekeliling citra

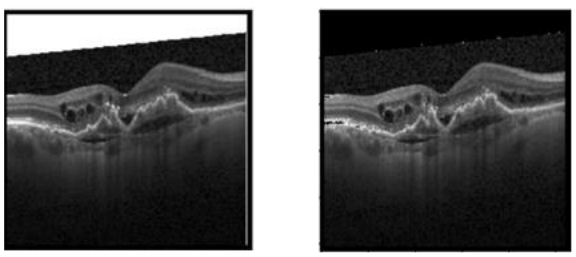
digunakan tidak dapat mendeteksi keseluruhan kontur citra seperti yang ditunjukkan pada Gambar 4.2.

4.2.1.2 Penghapusan border putih pada tepian citra

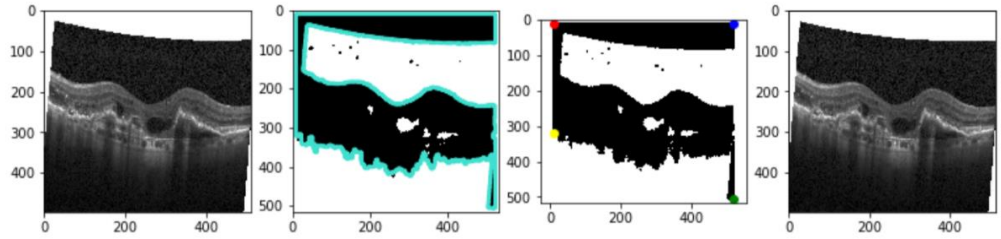
Implementasi tahapan untuk menghilangkan border putih disekitar tepian citra dalam bahasa python adalah sebagai berikut,

```
(row2, col2) = img2.shape
img3 = np.array(img2, copy=True)
for r in range(row2):
    for c in range(col2):
        px = img2[r][c]
        if (px >= 235):
            img3[r][c] = 0
```

implementasi dilakukan dengan cara menetapkan sebuah threshold sebesar 235 dimana nilai piksel yang melebihi threshold akan diubah menjadi 0. Hasil implementasi ditunjukkan pada Gambar 4.3. Apabila tidak dilakukan penghapusan, maka border putih ini akan menganggu proses segmentasi citra seperti yang ditunjukkan pada Gambar 4.4.



Gambar 4. 3 Implementasi penghapusan border putih ditepian citra



Gambar 4. 4 Segementasi yang gagal karena tidak adanya penghapusan border putih ditepi citra

4.2.1.3 Operasi Biner

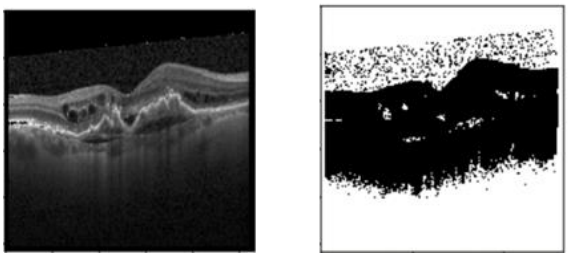
Implementasi operasi biner dalam bahasa python adalah sebagai berikut,

```
(row3, col3) = img2.shape
tresh = np.array(img3, copy=True)
for r in range(row3):
    for c in range(col3):
        px = img3[r][c]
        if (px < 35):
            tresh[r][c] = 255
        else:
            tresh[r][c] = 0
```

operasi biner dilakukan dengan cara menetapkan suatu nilai threshold yaitu sebesar 35. Setiap nilai piksel pada *source image* akan dibandingkan dengan nilai tersebut, apabila nilai piksel lebih dari nilai threshold maka output piksel menjadi 0 dan sebaliknya apabila nilai piksel kurang dari nilai threshold maka output piksel menjadi nilai piksel maksimal yaitu 255. Hasil implementasi operasi biner ditunjukkan pada Gambar 4.5 dimana operasi ini menghasilkan citra biner dengan *dark pixel* menempati area ROI dan *white pixel* menempati area *background*.

4.2.1.4 Operasi Morfologi

Operasi biner pada langkah sebelumnya meninggalkan *noise* pada gambar yang akan diminimalisir menggunakan operasi morfologi. Operasi morfologi yang digunakan adalah operasi *closing* yang didefinisikan seperti pada Persamaan 2.3, dimana A adalah citra input dan B adalah *structuring element.* Implementasi operasi morfologi dalam bahasa python adalah sebagai berikut,



Gambar 4. 5 Implementasi Operasi Biner

```
1  def disk(radius, dtype=np.uint8):
2      L = np.arange(-radius, radius + 1)
3      X, Y = np.meshgrid(L, L)
4      np.array((X ** 2 + Y ** 2) <= radius ** 2, dtype=np.uint8)
5  structuring_element=disk(3)
6  gal = morphology.closing(tresh, structuring_element)
```

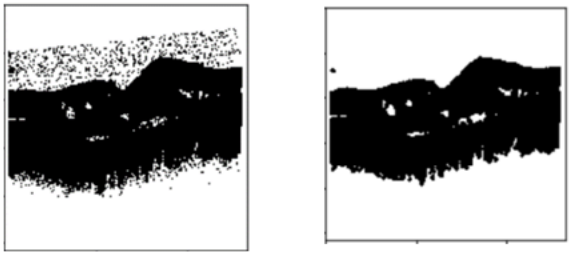
proses diawali dengan mendefinisikan fungsi untuk membuat matriks *structuring element. Structuring element* yang dibuat berbentuk disk dengan ukuran 3x3. Kemudian dilanjutkan dengan operasi *closing* pada baris ke-6. Operasi *closing* sendiri dilaksanakan dengan melakukan operasi dilasi terlebih dahulu dan kemudian diikuti dengan operasi erosi. Operasi morfologi ini mampu meminimalisir *noise* dan menghasilkan citra biner yang bersih seperti yang ditunjukkan pada Gambar 4.6.

4.2.1.5 Deteksi Kontur

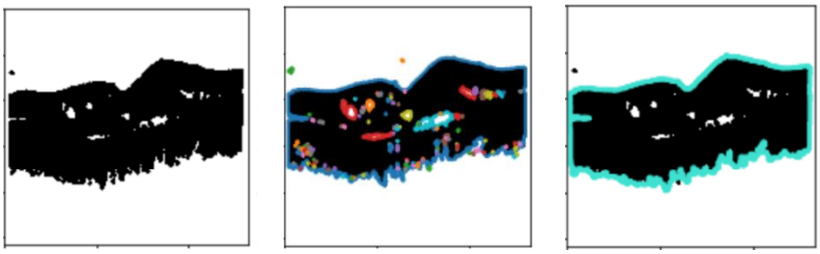
Setelah didapatkan hasil citra biner yang bersih, tahapan selanjutnya adalah mendeteksi kontur pada citra tersebut. Implementasi pendeteksian kontur dalam bahasa python adalah sebagai berikut,

```
1  contours = np.array(measure.find_contours(gal,0))
2  maxcontours = sorted(contours, key=lambda gal: len(gal))[-1]
```

pada fungsi *measure.find_contours* digunakan dua parameter yaitu *input array* yang merupakan citra hasil operasi morfologi dan sebuah nilai level isovalue. Nilai level pada fungsi ini sebenarnya digunakan untuk membuat citra menjadi biner. Namun pada tugas akhir ini, nilai tersebut diset sebesar 0 dan dianggap sebagai *don't care* karena pada tahapan sebelumnya sudah dilakukan operasi biner. Operasi biner dilakukan secara terpisah karena dibutuhkan metode lain yaitu operasi morfologi untuk membersihkan *noise* pada hasil citra biner. Kode pada baris ke-1 menghasilkan sebanyak 132 array kontur dengan ukuran masing-masing array sebesar (n,2) dimana n merupakan banyaknya baris dan 2 merupakan banyaknya kolom. Jumlah array kontur yang terdeteksi ini tentunya berbeda setiap citra. Gambar 4.7 b merupakan hasil keseluruhan kontur yang dideteksi. Tahapan selanjutnya kemudian mencari area kontur terbesar. Tahapan ini dilakukan dengan cara mengurutkan setiap array kontur dari yang terkecil hingga terbesar



Gambar 4. 6 Implementasi operasi morfologi



a b c

Gambar 4. 7 Implementasi deteksi kontur (a) citra hasil morfologi (b) kontur-kontur yang terdeteksi (c) area kontur terbesar

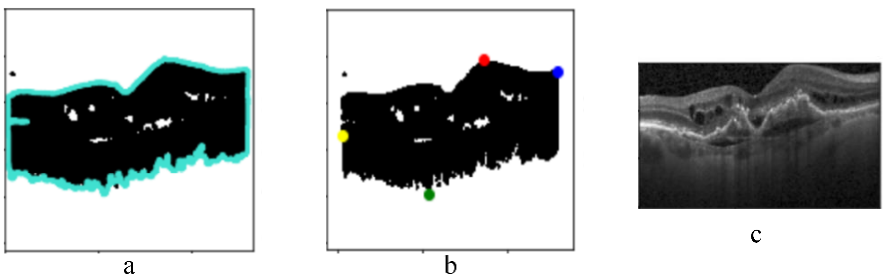
menggunakan fungsi sorted dimana array terbesar yaitu array dengan jumlah n baris terbanyak akan terletak di urutan paling terakhir. Setelah terurutkan maka parameter [-1] pada baris kode ke-2 akan mengambil nilai pada array terakhir dan menetapkannya sebagai area kontur terbesar. Hasil implementasi deteksi kontur ditunjukkan pada Gambar 4.7 c.

4.2.1.6 Autocropping

Kontur yang didapat pada tahap sebelumnya kemudian digunakan sebagai acuan untuk menentukan *extreme point. Extreme point* merupakan koordinat minimum dan maksimum sepanjang sumbu vertikal dan horizontal. Koordinat tersebut kemudian digunakan untuk proses *autocropping* area ROI. Implementasi *autocropping* dalam bahasa python adalah sebagai berikut

```
xmin = maxcontours[:,0].argmin() #top
xmax = maxcontours[:,0].argmax() #buttom
ymin = maxcontours[:,1].argmin() #left
ymax = maxcontours[:,1].argmax() #right
xmin_ = maxcontours[xmin,0].astype('uint16')
xmax_ = maxcontours[xmax,0].astype('uint16')
ymin_ = maxcontours[ymin,1].astype('uint16')
ymax_ = maxcontours[ymax,1].astype('uint16')
crop = img2[xmin_:xmax_, ymin_:ymax_]
```

Hasil implememementasi *autocropping* ditunjukkan pada Gambar 4.8.



a b c

Gambar 4. 8 Implementasi *autocropping* (a) kontur terbesar (b) deteksi *extreme point* (c) *cropped image*

4.2.2 Implementasi *Image Enhancement*

Implementasi perbaikan citra dilakukan untuk mereduksi *speckle noise* melalui proses *blurring.* Proses *blurring* akan menghaluskan citra secara seragam termasuk tepi atau ujung-ujung objek pada citra. Namun pada tugas akhir ini diinginkan agar tetap mempertahankan tepi dan tekstur dari citra. Oleh sebab itu pengujian reduksi *speckle noise* dilakukan menggunakan median filter (*classic edge-preserving filter*) dan bilateral filter (*edge-preserving filter*). Implementasi perbaikan citra pada bahasa *python* adalah sebagai berikut,

```
1  def median_filter(data, kernel_size):
2      temp = []
3      indexer = kernel_size // 2
4      data_final = []
5      data_final = numpy.zeros((len(data),len(data[0])))
6      for I in range(len(data)):
7          for j in range(len(data[0])):
8              for z in range(kernel_size):
9                  if I + z - indexer < 0 or I + z - indexer >
10                 len(data) - 1:
11                     for c in range(kernel_size):
12                         temp.append(0)
13                 else:
14                     if j + z - indexer < 0 or j + indexer >
15                     len(data[0]) - 1:
16                         temp.append(0)
17                     else:
18                         for k in range(kernel_size):
19                             temp.append(data[I + z -
20                             indexer][j + k - indexer])
21
22             temp.sort()
23             data_final[i][j] = temp[len(temp) // 2]
24             temp = []
25     return data_final
26 med = median_filter(image, 5)
27 bilateral = denoise_bilateral(image, sigma_color=20,
               sigma_spatial=0.8)
```

kode pada baris ke-1 hingga 26 merupakan implementasi dari median filter. Pemrosesan median filter diawali terlebih dahulu dengan mengurutkan nilai-nilai piksel yang ada pada kernel. Pengurutan akan menghasilkan nilai dari yang terkecil hingga terbesar yang kemudian dicari nilai tengahnya. Sementara kode pada baris ke-27 merupakan implementasi bilateral filter.

4.3 Implementasi *Balancing Data*

Implementasi *balancing data* yang dilakukan menggunakan teknik *undersampling* dijelaskan sebagai berikut

```
1  dir = "~/RawData/CNV"
2  balanced_dir = "~/DataBalanced/CNV"
3  files = [file for file in os.listdir(dir) if
   os.path.isfile(os.path.join(dir, file))]
4  rndnums     =     list(random.sample(range(0,     len(files)),
   len(files)))
5  train_count  = np.round(8615)
6  train_file_index = rndnums[0:int(train_count)+1]
7  train_file_name = [files[i] for I in train_file_index]
8  for x in (train_file_name):
         file = x
         shutil.copyfile(os.path.join(dir,file),
         os.path.join(balanced_dir, file))
```

Out[3]:

	category	n_train
0	CNV	37205
1	DME	11348
2	DRUSEN	8616
3	NORMAL	26315

Out[3]:

	category	n_train
0	CNV	8616
1	DME	8616
2	DRUSEN	8616
3	NORMAL	8616

Gambar 4. 10 Kondisi data sebelum dan sesudah dilakukan *balancing data*

```
Found 27572 images belonging to 4 classes.
Found 6892 images belonging to 4 classes.
{0: 'CNV', 1: 'DME', 2: 'DRUSEN', 3: 'NORMAL'}
```

Gambar 4. 9 Implementasi pemanggilan dan pembagian dataset

Proses *balancing* dilakukan pada masing-masing kelas secara terpisah. Baris ke-1 merupakan direktori awal, sementara baris ke-2 merupakan direktori untuk menyimpan hasil proses *balancing.* Baris ke-3 merupakan proses untuk menyimpan semua citra dalam folder kedalam sebuah array bernama 'files'. Baris ke-4 merupakan kode untuk merandom semua data pada array 'files'. Kemudian data pada urutan ke-0 hingga ke-8615 (jumlah kelas pada data terendah) akan dimasukkan kedalam array baru yang dijelaskan melalui kode pada baris ke-5 hingga ke-7. Data yang ada pada array baru akan di *copy* ke dalam sebuah folder baru yang ditunjukkan melalui kode pada baris ke-8. Hal tersebut dilakukan berulang pada masing-masing kelas sehingga semua kelas memiliki total jumlah data yang sama. Gambar 4.10 merupakan perbandingan kondisi data sebelum dan sesudah dilakukan *balancing*.

4.4 Implementasi Pelatihan dan Pengujian CNN

4.4.1 Implementasi Pemanggilan dan Pembagian Dataset

Implementasi pemanggilan dan pembagian dataset menjadi dataset untuk *training* dan dataset untuk validasi dalam bahasa python dijelaskan sebagai berikut

```
1  train_dir = “/home/riset/putri/train/”
2  imageSize=224
3  batch_size = 16
4  train_datagen =
   ImageDataGenerator(rescale=1./255,validation_split=0.2)
5  train_generator = train_datagen.flow_from_directory(
           train_dir,
           target_size=(imageSize, imageSize),
           batch_size=batch_size,
           subset=’training’,
           shuffle = True,
           class_mode=’categorical’)
6  validation_generator = train_datagen.flow_from_directory(
           train_dir,
           target_size=(imageSize, imageSize),
           batch_size=batch_size,
           subset=’validation’,
           shuffle=True,
           class_mode=’categorical’)
7  class_labels = validation_generator.class_indices
8  class_labels = {v: k for k, v in class_labels.items()}
9  print(class_labels)
```

Pada baris ke-1, variabel train_dir merupakan alamat dari dataset yang telah diberi label untuk masing-masing kelas. Baris ke-2 dan ke-3 masing-masing menunjukkan variabel yang menyimpan nilai ukuran citra dan *batch size*. *Batch size* sebesar 16 memiliki arti bahwa dalam satu kali iterasi terdapat 16 citra yang di latih. Baris ke-4 merupakan fungsi untuk menormalisasi keseluruhan citra serta membagi citra kedalam data validasi sebesar 20%. Normalisasi citra dilakukan dengan membagi nilai piksel dengan nilai piksel maksimal yaitu 255. Sehingga citra awal yang nilai pikselnya dalam rentang 0-255 menjadi 0-1. Proses ini bertujuan untuk meringankan komputasi pada saat pelatihan. Baris ke-5 dan ke-6 merupakan cuplikan kode untuk memuat data dalam direktori. Proses pemuatan data dilakukan menggunakan data generator dari pustaka Keras, karena apabila dilakukan pembacaan manual maka sistem akan mengalami *out of memory.* Kode pada baris ke-7 hingga ke-8 merupakan kode untuk menampilkan pelabelan masing-masing kelas. Output dari implementasi ditunjukkan pada Gambar 4.11 dimana sebanyak 27572 data digolongkan sebagai data training sementara 6892 data sebagai data validasi.

4.4.2 Implementasi Rancangan Model CNN

Implementasi model CNN yang dirancang pada subbab 3.5 dijelaskan dalam bahasa python sebagai berikut

```
1  num_classes = 4
2  def cnn():
```

```
    model = Sequential([
    Conv2D(64, (3, 3), activation="relu",padding = 'same',
    input_shape = (imageSize, imageSize, 3)),
    BatchNormalization(),
    Conv2D(64, (3, 3), activation="relu", padding = "same"),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.2),
    Conv2D(128, (3, 3), activation="relu",padding="same"),
    BatchNormalization(),
    Conv2D(128, (3, 3), activation="relu", padding="same"),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.2),
    Conv2D(256, (3, 3), activation="relu", padding="same"),
    BatchNormalization(),
    GlobalAveragePooling2D(),
    Dense(num_classes,activation="softmax"),
    ])
    return model
3   model = cnn()
4   model.compile(loss='categorical_crossentropy',
5  optimizer=Adam(lr=0.001,beta_1=0.9,beta_2=0.999),
                    metrics=['accuracy'])
```

Gambar 4.11 merupakan hasil ringkasan dari model beserta jumlah variabel keluaran seperti *weight* dan *bias* pada masing-masing lapisan. Dapat dilihat dari Gambar 4.12 bahwa output dari lapisan konvolusi pasti memiliki dimensi $h \times w$ yang sama dengan lapisan sebelumnya. Hal tersebut terjadi karena adanya *zero-padding* yang diterapkan pada setiap lapisan konvolusi. *Zero padding* merupakan parameter yang menentukan jumlah piksel berisi nilai 0 yang akan ditambah disetiap sisi dari input. Dengan adanya *zero padding* dimensi output dapat diatur sedemikian rupa agar tetap sama dengan dimensi input. Sehingga peluang melakukan *convolutional layer* yang dalam lebih besar dan menyebabkan lebih banyak fitur yang berhasil diekstrak. Selain itu dengan adanya *zero padding* dapat meningkatkan performa dari model karena *convolutional layer* berfokus pada informasi diantara *zero padding* tersebut. Selain itu dari Gambar 4.11 menunjukkan bahwa hasil output dimensi citra $(h \times w)$ dari lapisan *pooling* selalu dua kali lebih kecil. Hal tersebut terjadi karena lapisan *pooling* sendiri berfungsi untuk mereduksi dimensionalitas dari citra sesuai dengan ukuran *window* yang digunakan (dalam tugas akhir ini *window* di set sebesar 2×2) sehingga kompleksitas perhitungan model dapat diminimalisir. Output dari *global average pooling* (GAP) berupa vektor tunggal karena GAP sendiri mereduksi dimensi dari $h \times w \times d$ menjadi $1 \times 1 \times d$ dengan cara mengambil nilai rata-rata semua nilai hw. Berdasarkan output pada Gambar 4.11 didapatkan parameter total sebesar 558.916, parameter yang dilatih sebesar 557.636 serta parameter yang tidak dilatih sebesar 1.280. Hasil

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 224, 224, 64)      1792
_________________________________________________________________
batch_normalization_1 (Batch (None, 224, 224, 64)      256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 224, 224, 64)      36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 224, 224, 64)      256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 112, 112, 64)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 112, 112, 64)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 112, 112, 128)     73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 112, 112, 128)     512
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 112, 112, 128)     147584
_________________________________________________________________
batch_normalization_4 (Batch (None, 112, 112, 128)     512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 56, 56, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 56, 56, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 56, 56, 256)       295168
_________________________________________________________________
batch_normalization_5 (Batch (None, 56, 56, 256)       1024
_________________________________________________________________
global_average_pooling2d_1 ( (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028
=================================================================
Total params: 558,916
Trainable params: 557,636
Non-trainable params: 1,280
```

Gambar 4. 11 Hasil Ringkasan Model CNN

ini kemudian dioptimasi menggunakan optimasi Adam dengan tingkat belajar atau learning rate sebesar 0.001, beta1 sebesar 0.9 serta beta2 sebesar 0,999.

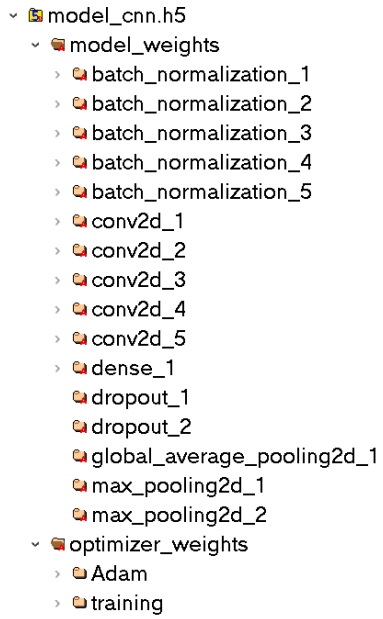
4.4.3 Implementasi Pelatihan Data Latih

Implementasi pelatihan data latih dijelaskan dalam bahasa python sebagai berikut

```
1  epochs=40
2  checkpoint = ModelCheckpoint(“model_cnn.h5”,
                                monitor=”val_acc”,
                                mode=”max”,
                                save_best_only = True,
                                verbose=1)
3  earlystop = EarlyStopping(monitor = ‘val_acc’,
                             patience = 10,
                             verbose = 1,
```

```
                              restore_best_weights = True)
4   callbacks_list = [earlystop, checkpoint]
5   history = model.fit_generator(
            train_generator,
            steps_per_epoch = train_generator.samples //
            batch_size,
            epochs = epochs,
            callbacks =
            callbacks_list+[MetricsCheckpoint('logs')],
            validation_data = validation_generator,
            validation_steps = validation_generator.samples //
    batch_size)
6   classes = list(class_labels.values())
7   y_pred = classifier.predict_generator(validation_generator,
    validation_generator.samples // batch_size+1,verbose=1)
8   y_pred_label = np.argmax(y_pred, axis=1)
9   confusion_mtx =
    confusion_matrix(validation_generator.classes, y_pred_label)
10  plot_confusion_matrix(confusion_mtx, classes = classes )
11  plt.show()
```

Besar epoch yang digunakan ditunjukkan dalam baris kode ke -1 yaitu sebesar 40 epoch. Pada implementasi data latih ini digunakan *callback* berupa *checkpoint* dan *earlystop* yang ditunjukkan pada baris kode ke-2 hingga ke-4. *Checkpoint* pada baris kode ke-2 merupakan *callback* yang berfungsi untuk menyimpan model di setiap akhir dari epoch hanya ketika nilai yang dimonitor (dalam hal ini nilai akurasi data validasi) mengalami peningkatan. Model akan disimpan ke dalam file berekstensi h5/hdf5 (*hierarchial data format*). Hasil model yang disimpan ditunjukkan pada Gambar 4.12. Sementara *earlystop* pada baris ke- 3 merupakan



Gambar 4. 12 Model CNN yang disimpan dalam file .h5

callback untuk mengakhiri proses pelatihan apabila nilai yang dimonitor (dalam hal ini nilai akurasi dari data validasi) tidak mengalami peningkatan lebih dari 10 epoch. Data latih kemudian di *training* menggunakan fungsi model.fit_generator pada baris kode ke- 5 dengan memasukkan parameter-parameter yang telah diset sebelumnya.

4.4.4 Implementasi Pengujian Data Validasi

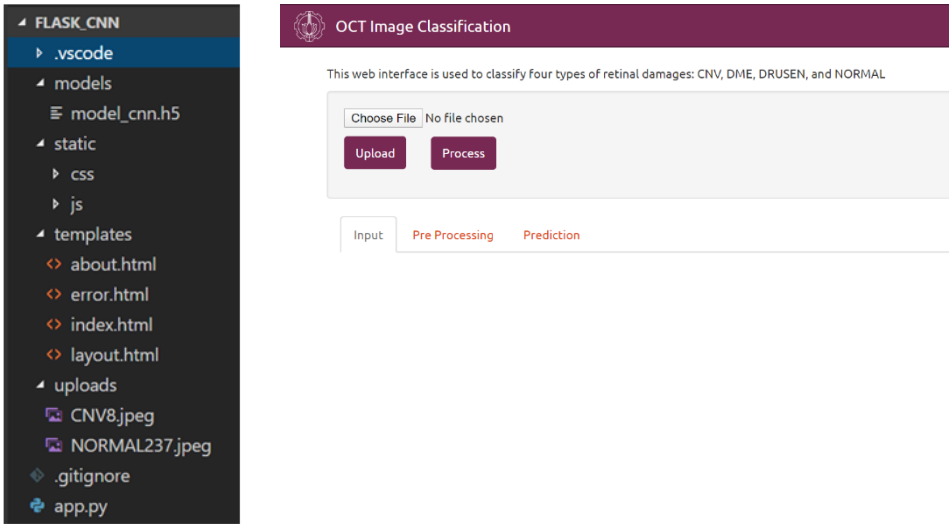
Pengujian data validasi pada Bab 5 dilakukan dengan menghitung nilai presisi, sensitivitas, dan akurasi. Implementasi pengujian data validasi dijelaskan dalam bahasa python sebagai berikut

```
1 classifier = load_model”/home/riset/putri/model_cnn.h5”)
2 score=classifier.evaluate_generator(validation_generator,step
  s=validation_generator.samples // batch_size, verbose=1)
3 print(‘\nTest                    result:                    %.3f
  Loss: %.3f’ %(score[1]*100,score[0]))
4 class_labels = validation_generator.class_indices
5 class_labels = {v: k for k, v in class_labels.items()}
```

Baris ke-1 menunjukkan pemuatan model yang telah disimpan pada tahap sebelumnya. Model kemudian disimpan dalam variabel classifier. Data validasi kemudian dievaluasi menggunakan fungsi classifier.evaluate_generator yang ditunjukkan pada baris ke-2 dan ke-3. Baris ke-4 hingga ke-6 merupakan kode untuk mendapatkan label kelas. Baris ke-7 hingga ke-13 merupakan kode untuk mendapatkan *confusion matrix* pada pengujian data validasi. Variabel y_pred menyimpan hasil prediksi probababilitas masing-masing kelas (dalam rentang 0 hingga 1) sementara y_pred_label merupakan variabel untuk menyimpan hasil kelas terprediksi.

4.5 Implementasi Uji Coba Model

Gambar 4.13(a) merupakan struktur file yang dibuat dalam implementasi uji coba model berupa *web interface*. Folder models merupakan folder untuk menyimpan model *machine learning* yang akan digunakan untuk memprediksi citra input. Folder static berisi semua aset statis seperti file css dan javascript. Css digunakan untuk mengatur tampilan elemen yang tertulis dalam bahasa markup dalam hal ini html. Sementara javascript dalam implementasi ini digunakan untuk memvisualisasikan hasil prediksi probabilitas kedalam sebuah bar grafik. Folder templates berisi semua file html yang akan direnderer sementara file app.py berfungsi sebagai main *back-end*. Halaman web kemudian dijalan secara lokal dimana tampilan awal web dapat dilihat pada Gambar 4.13(b). Pada halaman home terdapat 2 button dimana masing-masing button berfungsi untuk mengakses citra dari komputer serta melakukan proses pra pengolahan dan prediksi. Pada implementasi ini dibatasi hanya file citra yang berkekstensi .jpeg dan .jpg yang dapat di proses.



(a) (b)

Gambar 4. 13 Implementasi uji coba model (a) struktur file (b) tampilan *web interface*

BAB V
PENGUJIAN DAN ANALISA

Bab ini membahas mengenai pengujian dan analisa terhadap sistem yang telah diimplementasikan. Pengujian yang dilakukan berupa pengujian pra pengolahan yang meliputi segmentasi ROI dan perbaikan citra, pengujian tahapan *balancing data*, pengujian klasifikasi jenis kerusakan retina menggunakan CNN, serta uji coba model. Pengujian sistem dilakukan dengan mengikuti batasan-batasan masalah yang telah dirumuskan pada tugas akhir ini.

5.1 Pengujian Pra Pengolahan Citra

Pengujian pra pengolahan citra yang meliputi segmentasi ROI dan perbaikan citra dilakukan baik secara kualitatif maupun kuantitatif. Pengujian dan analisa terhadap bab inilah yang akan menjawab poin pertama pada rumusan masalah yaitu mengenai peningkatan kualitas citra.

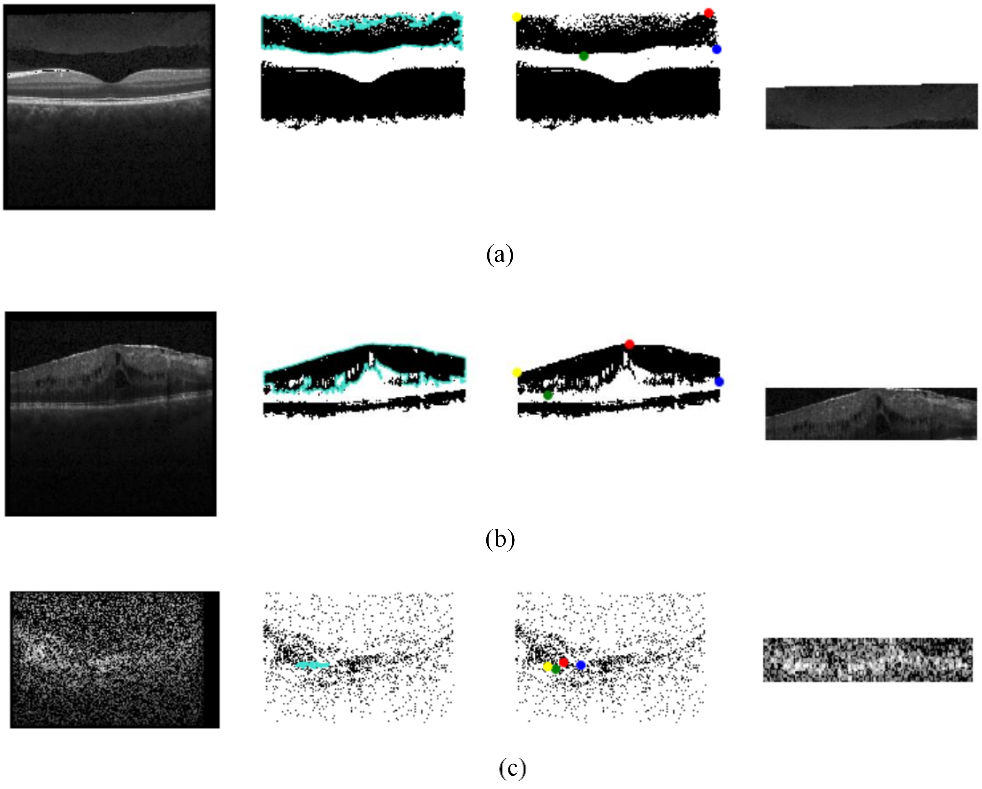
5.1.1 Pengujian Segmentasi ROI

Pengujian ini dilakukan untuk menguji tingkat kegagalan dari algoritma segmentasi ROI yang telah diimplementasikan. Penentuan citra yang gagal segmentasi dilakukan secara manual menggunakan inspeksi visual dengan memperhatikan struktur anatomi retina yang sudah dijelaskan pada bagian subbab 2.1. Hasil pengujian ditunjukkan pada Tabel 5.1. Dapat dilihat dari tabel tersebut, secara keseluruhan tingkat kegagalan memiliki persentase sebesar 0.016%, sehingga dapat dikatakan bahwa algoritma segmentasi yang telah dirancang dapat mensegmentasi area ROI pada citra dengan baik. Kegagalan segmentasi yang terjadi dapat disebabkan oleh beberapa hal, yaitu:

1. Terdapat area pada *background* yang memiliki nilai piksel hampir sama dengan area ROI, sehingga apabila dilakukan operasi biner maka terdapat dua area kontur berwarna hitam yang memiliki ukuran cukup besar. Sehingga algoritma deteksi kontur dapat salah mendeteksi kontur seperti yang ditunjukkan pada Gambar 5.1 a
2. Kondisi citra yang terlalu gelap mengakibatkan area ROI pada hasil operasi biner mengalami penyusutan berlebih seperti yang ditunjukkan pada Gambar 5.1 b. Sehingga apabila dilakukan deteksi kontur, maka area ROI tidak dapat terdeteksi dengan baik. Kondisi citra yang terlalu gelap ini membutuhkan parameter nilai *threshold* pada operasi biner yang berbeda yaitu sebesar 15
3. Terdapat *speckle noise* berlebih pada citra seperti yang ditunjukkan pada Gambar 5.1 c. *Speckle noise* sendiri merupakan gangguan granular yang sceara inheren dapat menyebabkan penurunan kualitas pada citra

Tabel 5. 1 Tingkat kegagalan dari algoritma segmentasi

Kategori	Total Citra	Jumlah Citra yang Gagal Segmentasi	Tingkat kegagalan (%)
CNV	37447	419	0.01
DME	11590	360	0.03
DRUSEN	8858	83	0.009
NORMAL	26557	484	0.018
TOTAL	84452	1346	0.016



Gambar 5. 1 Contoh citra yang gagal segmentasi (a) disebabkan karena terdapat area pada *background* yang memiliki nilai piksel hampir sama dengan area ROI (b) kondisi citra terlalu gelap (c) terlalu banyak speckle noise pada citra

5.1.2 Pengujian Perbaikan Citra

Tahapan perbaikan citra dilakukan untuk mereduksi *speckle noise* pada citra. Sesuai dengan rancangan dan implementasi sistem yang telah dijelaskan pada bab sebelumnya, perbaikan citra akan diuji cobakan menggunakan filter median dan bilateral filter. Pengujian dan analisa dilakukan secara kuantitatif dan kualitatif dengan merubah parameter pada setiap jenis filter yang kemudian akan dianalisa melalui nilai MSE dan PSNR. Hasil pengujian secara kuantitatif tertera pada Tabel 5.2 dan 5.3. Perhitungan MSE pada tabel memberikan nilai error kuadrat rata-rata antara citra original dengan citra terfilter. Sementara perhitungan PSNR

memberikan perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau atau *noise* yang berpengaruh pada citra tersebut.

Pengujian filter median pada Tabel 5.2 dilakukan pada kernel berukuran ganjil guna memudahkan pemberian poros tengah sehingga lebih mudah dalam melakukan pengolahan citra. Dari tabel tersebut dapat diamati bahwa nilai PSNR tertinggi terdapat pada kernel berukuran 3×3. Selain itu, semakin besar ukuran kernel yang digunakan pada filter median, semakin rendah nilai PSNR yang dihasilkan. Nilai PSNR yang rendah menunjukkan bahwa hasil citra terfilter mengalami banyak perubahan sehingga citra hasil terfilter sangat berbeda dengan citra original. Perubahan yang dimaksud disini adalah citra mengalami pengkaburan seperti yang ditunjukkan pada Gambar 5.2. Hasil pengkaburan ini mengakibatkan batas tepian antara area ROI dengan backgorund, batas antar lapisan retina, serta tekstur citra menjadi tidak jelas. Hasil ini tentunya akan berpengaruh pada proses klasifikasi. Sehingga dilakukan pengujian lebih lanjut menggunakan bilateral filter.

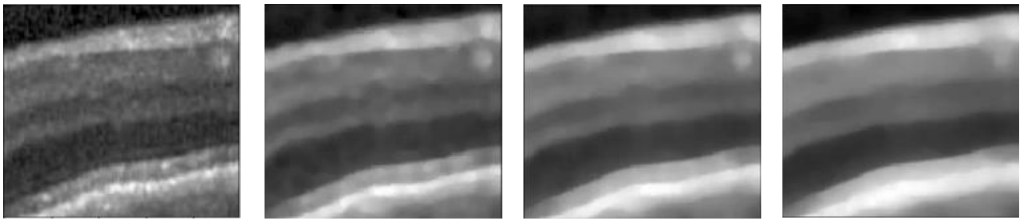
Pengujian menggunakan bilateral filter dilakukan dengan ukuran sigma yang berbeda beda. Berdasarkan Tabel 5.3 semakin besar ukuran sigma yang digunakan semakin rendah nilai PSNR yang dihasilkan. Sama seperti median filter, nilai PSNR yang rendah menunjukkan bahwa hasil citra terfilter mengalami banyak perubahan dibandingkan dengan citra original. Gambar 5.3 menunjukkan perbandingan hasil citra terfilter pada bilateral filter dengan parameter sigma yang berbeda-beda. Dari gambar terlihat bahwa citra dengan nilai sigma range sebesar 0.5 dan sigma spatial sebesar 20 selain mampu untuk menghilangkan *noise* yang ada pada citra juga mampu mempertahankan batas tepi serta tekstur citra. Pengujian pada citra tersebut juga menghasilkan nilai PSNR tertinggi dimana nilai tersebut mengindikasikan bahwa semakin baik penapisan *noise* atau derau yang dihasilkan.

Tabel 5. 2 Pengujian Filter Median pada Citra CNV 57

Kernel	MSE	PSNR (dB)
3×3	0.0019	27.21
5×5	0.0026	25.78
7×7	0.0030	25.15

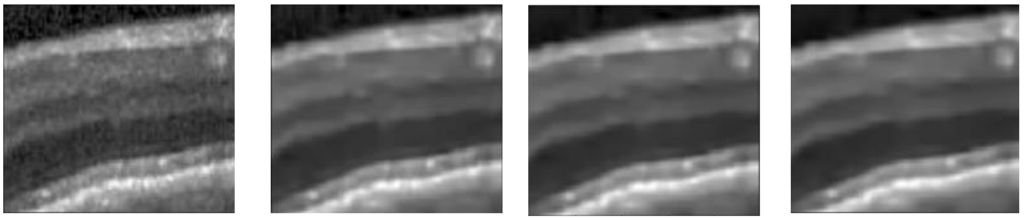
Tabel 5. 3 Pengujian Bilateral filter pada Citra CNV 57

σ_r	σ_s	MSE	PSNR (dB)
0.5	20	0.00055	32.56
0.6	25	0.00077	31.14
0.8	30	0.00090	30.46



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

Gambar 5. 3 Hasil filter median pada zoom citra CNV 57 (a) original (b) kernel 3x3 (c) kernel 5x5 (d) kernel 7x7



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

Gambar 5. 2 Hasil bilateral filter pada zoom citra CNV 57 (a) original (b) σ_r = 0.5 $\sigma_r = 20$ (c) σ_r = 0.6 $\sigma_r = 25$ (d) σ_r = 0.8 $\sigma_r = 30$

5.2 Pengujian CNN

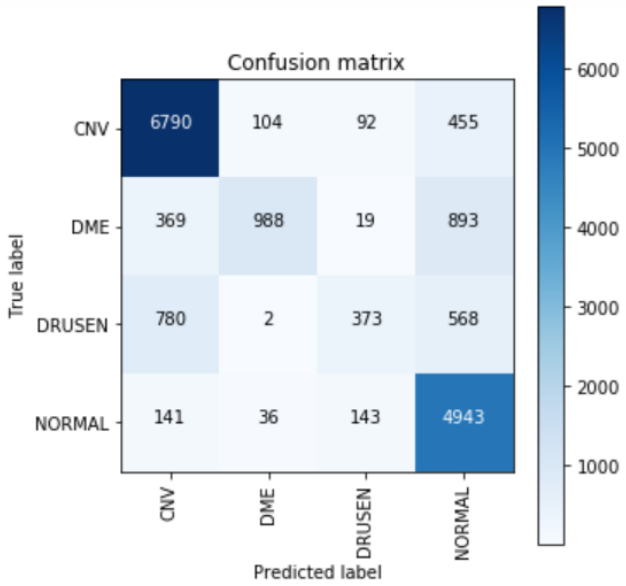
Pengujian CNN dilakukan dengan cara menghitung nilai akurasi, presisi, sensitivitas, dan spesifisitas pada data validasi. Pengujian CNN terdiri dari empat kondisi berbeda yang dilakukan untuk mengetahui pengaruh ada tidaknya tahapan *pre-processing* dan *balancing* data.

5.2.1 Perhitungan Uji Performansi pada Kondisi 1

Perhitungan uji performansi pada kondisi 1 dilakukan dengan kondisi citra yang digunakan merupakan citra hasil pra pengolahan serta tidak dilakukannya tahapan *balancing data* pada data latih dan data validasi. Data validasi yang digunakan pada kondisi 1 berjumlah sebanyak 16.696 citra. Perhitungan nilai akurasi, presisi, sensitivitas, dan spesifisitas dilakukan dengan memasukkan nilai *True* Positive (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN) yang ada pada *confusion matrix*. *Confusion matrix* pada Gambar 5.4 juga memberikan informasi mengenai perbandingan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi sebenarnya. Setiap kolom dari *confusion matrix* merepresentasikan *instance* dari kelas prediksi sementara setiap baris dari *confusion matrix* mewakili *instance* dari kelas aktual.

5.2.1.1 Akurasi

Akurasi merupakan persentase dari total data yang dinilai benar yaitu data yang terletak pada nilai diagonal dalam *confusion matrix* dibanding dengan total keseluruhan data. Perhitungan akurasi pada kondisi 1 dijelaskan sebagai berikut:



Gambar 5. 4 *Confusion matrix* pengujian kondisi 1

$$Akurasi = \frac{TP + TN}{TP + TN + FN + FP} = \frac{13094}{16696} = 0.784$$

5.2.1.2 Presisi

Presisi didapat dengan memasukkan nilai TP dan FP yang ada pada *confusion matrix* 5.4 kedalam Persamaan 2.12. Berikut merupakan perhitungan presisi pada kondisi 1

a) CNV

$$Presisi = \frac{TP}{TP + FP} = \frac{6790}{6790 + (369 + 780 + 141)} = 0.84$$

b) DME

$$Presisi = \frac{988}{988 + (104 + 2 + 36)} = 0.874$$

c) DRUSEN

$$Presisi = \frac{373}{373 + (92 + 19 + 143)} = 0.595$$

d) NORMAL

$$Presisi = \frac{4943}{4943 + (455 + 893 + 568)} = 0.721$$

5.2.1.3 Sensitivitas

Sensitivitas didapat dengan memasukkan nilai TP dan FN yang ada pada *confusion matrix* 5.4 kedalam Persamaan 2.13. Berikut merupakan perhitungan sensitivitas pada kondisi 1

a) CNV

$$Sensitivitas = \frac{TP}{TP + FN} = \frac{9906}{9906 + (104 + 92 + 455)} = 0.913$$

b) DME

$$Sensitivitas = \frac{988}{988 + (104 + 2 + 36)} = 0.435$$

c) DRUSEN

$$Sensitivitas = \frac{373}{373 + (92 + 19 + 143)} = 0.216$$

d) NORMAL

$$Sensitivitas = \frac{4943}{4943 + (455 + 893 + 568)} = 0.939$$

5.2.1.4 Spesifisitas

Spesifisitas didapat dengan memasukkan nilai TP dan FN yang ada pada *confusion matrix* 5.4 kedalam Persamaan 2.14. Berikut merupakan perhitungan spesifisitas pada kondisi 1

a) CNV

$$Spesifisitas = \frac{TN}{TN + FP} = \frac{6304}{6304 + (369 + 780 + 141)} = 0.83$$

b) DME

$$Spesifisitas = \frac{12106}{12106 + (104 + 2 + 36)} = 0.988$$

c) DRUSEN

$$Spesifisitas = \frac{12721}{12721 + (92 + 19 + 143)} = 0.98$$

d) NORMAL

$$Spesifisitas = \frac{8151}{8151 + (455 + 893 + 568)} = 0.81$$

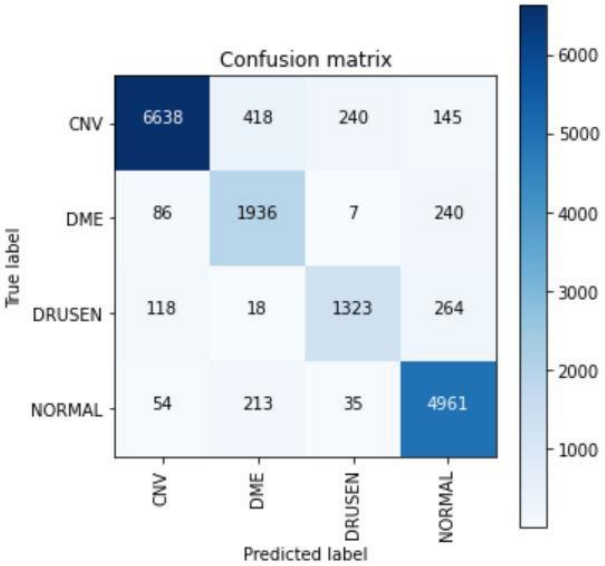
5.2.2 Perhitungan Uji Performansi pada Kondisi 2

Perhitungan uji performansi pada kondisi 2 dilakukan dengan citra input merupakan citra hasil pra pengolahan serta dilakukannya tahapan *balancing data* menggunakan teknik *weight balancing*. Data validasi yang digunakan pada kondisi 2 berjumlah 6.892 citra. *Confusion matrix* pada Gambar 5.5 digunakan untuk menentukan nilai akurasi, presisi, sensitivitas, dan spesifisitas sebagai berikut:

5.2.2.1 Akurasi

Perhitungan akurasi pada kondisi 3 dilakukan sebagai berikut:

$$Akurasi = \frac{TP + TN}{TP + TN + FN + FP} = \frac{14858}{16696} = 0.889$$



Gambar 5. 5 *Confusion matrix* pengujian kondisi 2

5.2.2.2 Presisi

Presisi didapat dengan memasukkan nilai TP dan FP yang ada pada *confusion matrix* 5.5 kedalam Persamaan 2.12. Berikut merupakan perhitungan presisi pada kondisi 3

a) CNV

$$Presisi = \frac{TP}{TP + FP} = \frac{6638}{6638 + (86 + 118 + 54)} = 0.963$$

b) DME

$$Presisi = \frac{1936}{1936 + (418 + 18 + 213)} = 0.749$$

c) DRUSEN

$$Presisi = \frac{1323}{1323 + (240 + 7 + 35)} = 0.824$$

d) NORMAL

$$Presisi = \frac{4961}{4961 + (145 + 240 + 264)} = 0.884$$

5.2.2.3 Sensitivitas

Sensitivitas didapat dengan memasukkan nilai TP dan FN yang ada pada *confusion matrix* 5.5 kedalam Persamaan 2.13. Berikut merupakan perhitungan sensitivitas pada kondisi 3

a) CNV

$$Sensitivitas = \frac{TP}{TP + FN} = \frac{6638}{6638 + (418 + 240 + 145)} = 0.892$$

b) DME

$$Sensitivitas = \frac{1936}{1936 + (86 + 7 + 240)} = 0.853$$

c) DRUSEN

$$Sensitivitas = \frac{1323}{1323 + (118 + 18 + 264)} = 0.768$$

d) NORMAL

$$Sensitivitas = \frac{4961}{4961 + (54 + 213 + 35)} = 0.943$$

5.2.1.4 Spesifisitas

Spesifisitas didapat dengan memasukkan nilai TP dan FN yang ada pada *confusion matrix* 5.5 kedalam Persamaan 2.14. Berikut merupakan perhitungan spesifisitas pada kondisi 3

a) CNV

$$Spesifisitas = \frac{TN}{TN + FP} = \frac{10058}{10058 + (86 + 118 + 54)} = 0.975$$

b) DME

$$Spesifisitas = \frac{14760}{14760 + (418 + 18 + 213)} = 0.958$$

c) DRUSEN

$$Spesifisitas = \frac{15373}{15373 + (240 + 7 + 35)} = 0.982$$

d) NORMAL

$$Spesifisitas = \frac{11735}{11735 + (145 + 240 + 264)} = 0.948$$

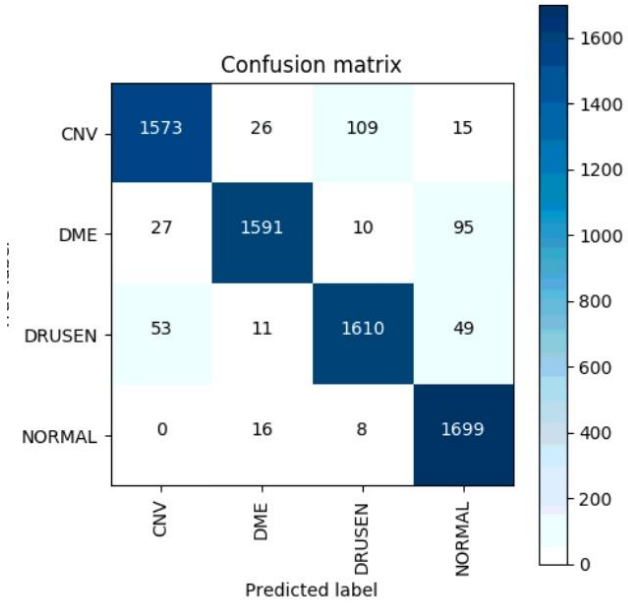
5.2.3 Perhitungan Uji Performansi pada Kondisi 3

Perhitungan uji performansi pada kondisi 3 dilakukan dengan citra input merupakan citra hasil pra pengolahan serta dilakukannya tahapan *balancing data* menggunakan teknik *random undersampling*. Data validasi yang digunakan pada kondisi 3 berjumlah 6.892 citra. *Confusion matrix* pada Gambar 5.6 digunakan untuk menentukan nilai akurasi, presisi, sensitivitas, dan spesifisitas sebagai berikut:

5.2.2.1 Akurasi

Perhitungan akurasi pada kondisi 3 dilakukan sebagai berikut:

$$Akurasi = \frac{TP + TN}{TP + TN + FN + FP} = \frac{6473}{6892} = 0.939$$



Gambar 5. 6 *Confusion matrix* pengujian kondisi 3

5.2.2.2 Presisi

Presisi didapat dengan memasukkan nilai TP dan FP yang ada pada *confusion matrix* 5.5 kedalam Persamaan 2.12. Berikut merupakan perhitungan presisi pada kondisi 3

a) CNV

$$Presisi = \frac{TP}{TP + FP} = \frac{1573}{1573 + (27 + 53 + 0)} = 0.952$$

b) DME

$$Presisi = \frac{1591}{1591 + (26 + 11 + 16)} = 0.968$$

c) DRUSEN

$$Presisi = \frac{1610}{1610 + (109 + 10 + 8)} = 0.927$$

d) NORMAL

$$Presisi = \frac{1699}{1699 + (15 + 95 + 49)} = 0.914$$

5.2.2.3 Sensitivitas

Sensitivitas didapat dengan memasukkan nilai TP dan FN yang ada pada *confusion matrix* 5.5 kedalam Persamaan 2.13. Berikut merupakan perhitungan sensitivitas pada kondisi 3

a) CNV

$$Sensitivitas = \frac{TP}{TP + FN} = \frac{1573}{1573 + (26 + 109 + 15)} = 0.913$$

b) DME

$$Sensitivitas = \frac{1591}{1591 + (27 + 10 + 95)} = 0.923$$

c) DRUSEN

$$Sensitivitas = \frac{1610}{1610 + (53 + 11 + 49)} = 0.934$$

d) NORMAL

$$Sensitivitas = \frac{1699}{1699 + (0 + 16 + 8)} = 0.986$$

5.2.1.4 Spesifisitas

Spesifisitas didapat dengan memasukkan nilai TP dan FN yang ada pada *confusion matrix* 5.5 kedalam Persamaan 2.14. Berikut merupakan perhitungan spesifisitas pada kondisi 3

a) CNV

$$Spesifisitas = \frac{TN}{TN + FP} = \frac{4900}{4900 + (27 + 53 + 0)} = 0.985$$

b) DME

$$Spesifisitas = \frac{4882}{4882 + (26 + 11 + 16)} = 0.99$$

c) DRUSEN

$$Spesifisitas = \frac{4863}{4863 + (109 + 10 + 8)} = 0.977$$

d) NORMAL

$$Spesifisitas = \frac{4774}{4774 + (15 + 95 + 49)} = 0.97$$

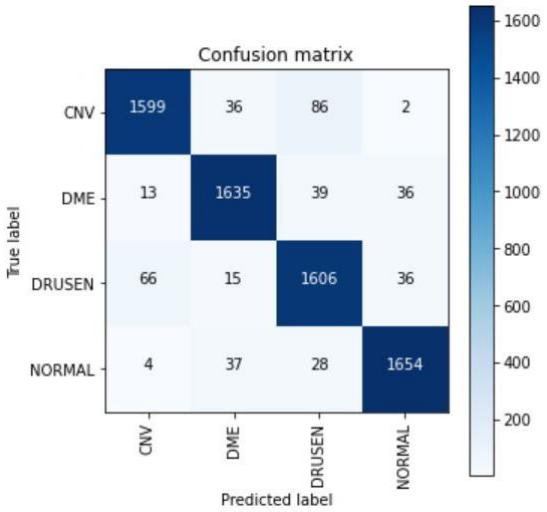
5.2.3 Perhitungan Uji Performansi pada Kondisi 4

Perhitungan uji performansi pada kondisi 4 dilakukan dengan kondisi citra input yang digunakan merupakan citra yang tidak melalui proses pra pengolahan serta dilakukan proses *balancing data* menggunakan teknik *random undersampling*. Data validasi yang digunakan pada kondisi 4 berjumlah sebanyak 6.892 citra. *Confusion matrix* pada Gambar 5.7 digunakan untuk menentukan nilai akurasi, presisi, sensitivitas, dan spesifisitas sebagai berikut:

5.2.3.1 Akurasi

Perhitungan akurasi pada kondisi 4 dilakukan sebagai berikut:

$$Akurasi = \frac{TP + TN}{TP + TN + FN + FP} = \frac{6494}{6892} = 0.942$$



Gambar 5. 7 *Confusion matrix* pengujian kondisi 4

5.2.3.2 Presisi

Presisi didapat dengan memasukkan nilai TP dan FP yang ada pada *confusion matrix* 5.4 kedalam Persamaan 2.12. Berikut merupakan perhitungan presisi pada kondisi 4

a) CNV

$$Presisi = \frac{TP}{TP + FP} = \frac{1599}{1599 + (13 + 66 + 4)} = 0.951$$

b) DME

$$Presisi = \frac{1635}{1635 + (36 + 15 + 37)} = 0.949$$

c) DRUSEN

$$Presisi = \frac{1606}{1606 + (86 + 39 + 28)} = 0.913$$

d) NORMAL

$$Presisi = \frac{1654}{1654 + (2 + 36 + 36)} = 0.957$$

5.2.3.3 Sensitivitas

Sensitivitas didapat dengan memasukkan nilai TP dan FN yang ada pada *confusion matrix* 5.6 kedalam Persamaan 2.13. Berikut merupakan perhitungan sensitivitas pada kondisi 4

a) CNV

$$Sensitivitas = \frac{TP}{TP + FN} = \frac{1599}{1599 + (36 + 86 + 2)} = 0.928$$

b) DME

$$Sensitivitas = \frac{1635}{1635 + (13 + 39 + 36)} = 0.949$$

c) DRUSEN

$$Sensitivitas = \frac{1606}{1606 + (66 + 15 + 36)} = 0.932$$

d) NORMAL

$$Sensitivitas = \frac{1654}{1654 + (4 + 37 + 28)} = 0.96$$

5.2.1.4 Spesifisitas

Spesifisitas didapat dengan memasukkan nilai TP dan FN yang ada pada *confusion matrix* 5.6 kedalam Persamaan 2.18. Berikut merupakan perhitungan spesifisitas pada kondisi 4

a) CNV

$$Spesifisitas = \frac{TN}{TN + FP} = \frac{4895}{4895 + (13 + 66 + 4)} = 0.985$$

b) DME

$$Spesifisitas = \frac{4859}{4859 + (36 + 15 + 37)} = 0.984$$

c) DRUSEN

$$Spesifisitas = \frac{4888}{4888 + (86 + 39 + 28)} = 0.972$$

d) NORMAL

$$Spesifisitas = \frac{4840}{4840 + (2 + 36 + 36)} = 0.986$$

5.2.4 Analisa Hasil Masing-Masing Kondisi

Tabel 5.4 menunjukkan perbandingan hasil pada masing-masing kondisi. Seperti yang telah dijelaskan sebelumnya, pengujian dengan tiga kondisi yang berbeda dilakukan untuk mengetahui pengaruh tahapan *balancing data*, *preprocessing data,* serta menentukan metode *balancing data* mana yang lebih baik.

5.2.4.1 Pengaruh tahapan *balancing data*

Pengaruh tahapan *balancing data* didapat dengan membandingkan hasil perhitungan uji performansi pada kondisi 1 dan 3. Hasil perhitungan presisi pada kondisi 1 menunjukkan bahwa kelas DRUSEN memiliki nilai presisi yang paling rendah yaitu sebesar 0.595. Nilai presisi yang rendah mengindikasikan bahwa terdapat banyaknya kasus *false positive*. Dalam kasus DRUSEN, *false positive* terjadi apabila terdapat citra bukan DRUSEN tetapi dikenal sebagai DRUSEN. Sementara itu, hasil perhitungan presisi pada kondisi 3 menunjukkan bahwa dengan melakukan tahapan *balancing data,* dapat mempengaruhi nilai presisi pada masing-masing kelas. Peningkatan nilai presisi yang cukup signifikan terjadi pada kelas DRUSEN yang mana sebelumnya kelas ini merupakan kelas paling minoritas. *Balancing data* membuat jumlah data pada masing-masing kelas menjadi seimbang

sehingga terjadi penurunan terhadap kasus *false positive* yang berdampak pada peningkatan terhadap nilai presisi.

Kemudian apabila dilihat dari perhitungan sensitivitas, hasil perhitungan sensitivitas pada kondisi 1 menunjukkan bahwa kelas DME dan DRUSEN memiliki nilai sensitivitas yang rendah yaitu sebesar 0.435 dan 0.216. Nilai sensitivitas yang rendah mengindikasikan banyaknya kasus *false negative*. Misalnya dalam kasus DRUSEN, *false negative* terjadi apabila terdapat citra DRUSEN tetapi tidak dikenal sebagai DRUSEN. Rendahnya nilai sensitivitas pada kelas DME dan DRUSEN dapat terjadi karena data negatif (data yang bukan DME dan DRUSEN) lebih besar daripada data positif. Hal ini sejalan dengan posisi kelas DRUSEN dan DME sebagai kelas minoritas. Sementara itu, hasil perhitungan pada kondisi 3 menunjukkan bahwa dengan melakukan tahapan *balancing data*, dapat meningkatkan nilai sensitivitas pada masing-masing kelas. Peningkatan nilai sensitivitas yang cukup signifikan terjadi pada kelas DME dan DRUSEN yang mana sebelumnya kelas ini merupakan kelas minoritas. Sama seperti hasil presisi, *balancing data* membuat jumlah data pada masing-masing kelas menjadi seimbang sehingga terjadi penurunan terhadap kasus *false negative* yang berdampak pada peningkatan terhadap nilai sensitivitas.

Perhitungan akurasi pada kondisi 1 menghasilkan nilai sebesar 0.784 sementara pada kondisi 3 menghasilkan nilai sebesar 0.939. Dari perhitungan tersebut menunjukkan bahwa proses *balancing data* juga dapat meningkatkan hasil akurasi.

Tabel 5. 4 Perbandingan hasil masing-masing kondisi

Cond.	Preprocessing Data	Balancing Data	Performance Evaluation	CNV	DME	DRUSEN	NORMAL	Average
1	✓	-	Precision	0.84	0.874	0.595	0.721	0.758
			Sensitivity	0.913	0.435	0.216	0.939	0.626
			Specificity	0.83	0.988	0.98	0.81	0.902
			Accuracy	0.784				
2	✓	✓ (*weight balancing*)	Precision	0.963	0.749	0.824	0.884	0.855
			Sensitivity	0.892	0.853	0.768	0.943	0.864
			Specificity	0.975	0.958	0.982	0.948	0.966
			Accuracy	0.889				
3	✓	✓ (*random undersampling*)	Precision	0.952	0.968	0.927	0.914	0.94
			Sensitivity	0.913	0.923	0.934	0.986	0.939
			Specificity	0.985	0.99	0.977	0.97	0.981
			Accuracy	0.939				
4	-	✓ (*random undersampling*)	Precision	0.951	0.949	0.913	0.957	0.942
			Sensitivity	0.928	0.949	0.932	0.96	0.942
			Specificity	0.985	0.984	0.972	0.986	0.982
			Accuracy	0.942				

5.2.4.2 Pengaruh tahapan *preprocessing data*

Pengaruh tahapan *preprocessing data* didapat dengan membandingkan hasil perhitungan uji performansi pada kondisi 3 dan 4. Seperti yang telah dijelaskan sebelumnya, tahapan *preprocessing data* dilakukan melalui 2 tahap. Tahap pertama adalah melakukan segmentasi area ROI dengan tujuan untuk menghilangkan fitur-fitur berupa *background* hitam yang tidak dibutuhkan selama proses pelatihan. Sementara tahap yang kedua berupa perbaikan citra yang berfungsi untuk mereduksi *speckle noise* pada citra. Sehingga dengan adanya proses pra pengolahan, dapat meningkatkan hasil klasifikasi. Namun berdasarkan Tabel 5.4, pengujian menggunakan data mentah memiliki nilai akurasi, rata-rata spesifisitas dan rata-rata sensitivitas lebih tinggi dibanding dengan citra yang melalui proses pra pengolahan. Hal tersebut dapat disebabkan karena pada hasil citra terpraproses masih terdapatnya citra yang mengalami kegagalan segmentasi.

5.2.4.3 Perbandingan Metode *Balancing Data*

Untuk mengatasi masalah ketidakseimbangan data, digunakan 2 metode yang berbeda yaitu *weight balancing* dan *random undersampling* dimana hasil uji performansi tertera pada Tabel 5.4. Dari tabel tersebut dapat dilihat bahwa penyeimbangan data baik menggunakan metode *weight balancing* atau *random undersampling* dapat menghasilkan nilai akurasi, presisi, sensitivitas, dan spesifisitas yang lebih baik dibandingkan dengan pengujian pada kondisi 1. Selanjutnya apabila dibandingkan hasil pada pengujian kondisi 2 dan 3, metode *random undersampling* menghasilkan hasil yang lebih baik daripada metode *weight balancing*. Hal tersebut dapat disebabkan karena pada *random undersampling* masing-masing bobot pada setiap kelas sudah pasti sama rata yaitu sebesar 1.

5.3 Analisa Perbandingan dengan Penelitian Sebelumnya

Subbab ini membahas mengenai analisa perbandingan dari metode yang diusulkan dengan metode-metode yang sudah ada pada penelitian sebelumnya. Hasil perbandingan dapat dilihat pada Tabel 5.5. Dapat dilihat dari tabel tersebut, apabila dibandingkan dengan penelitian dari Kermany dkk [9], hasil penelitian yang dirancang memiliki nilai akurasi dan sensitivitas yang lebih rendah. Dan apabila dibandingkan dengan penelitian dari Islam dkk [10], hasil penelitian yang dirancang juga memiliki nilai akurasi, presisi, dan spesifisitas yang lebih rendah. Hal tersebut dapat disebabkan oleh beberapa hal. Yang pertama adalah karena pada penelitian [9] dan [10] menggunakan model *transfer learning*. *Transfer learning* sendiri merupakan suatu teknik atau metode yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan permasalahan lain yang serupa dengan cara menggunakannya sebagai *starting point* kemudian memodifikasi dan mengupdate parameternya sehingga sesuai dengan dataset yang baru. Penyebab yang kedua adalah penggunaan lapisan ekstraksi fitur yang semakin dalam pada penelitian [9] dan [10] yaitu sebanyak 48 dan 201 lapisan, menyebabkan semakin banyak fitur-fitur yang dapat diekstrak sehingga semakin detail informasi yang didapatkan dari sebuah citra input. Penyebab yang ketiga adalah arsitektur

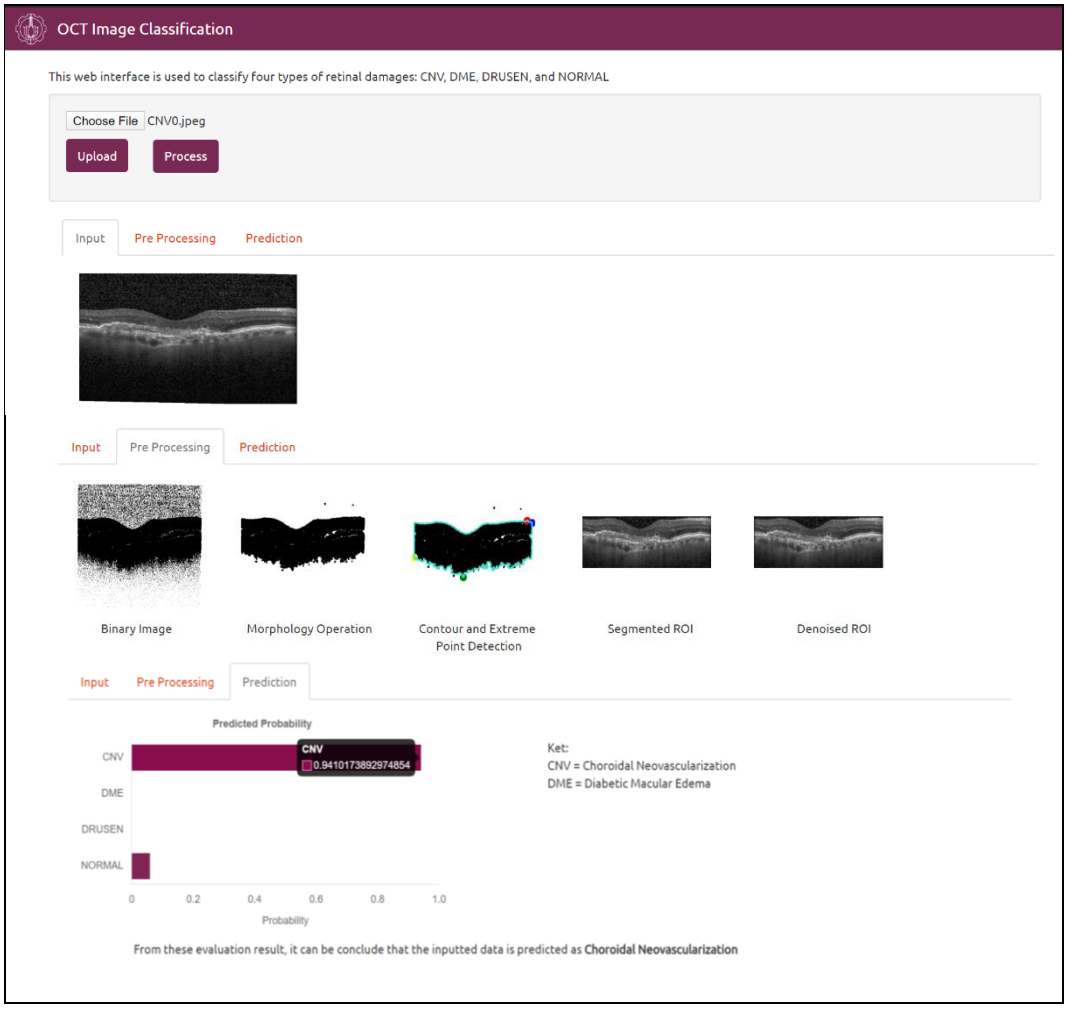
Tabel 5. 5 Perbandingan dengan Penelitian Sebelumnya

	Metode			Akurasi	Presisi	SN	SP
	Pre processing	Network	Data Pengujian				
Kermany, dkk [9]	-	48layer Inception v3 architecture	Data Validasi	96.6%	-	97.8%	97.4%
Islam, dkk [10]	ROI Segmentation	201layer DenseNet architecture	Data Testing	98.6%	98.6%	-	99.5%
Najeeb, dkk [30]	ROI Segmentation	2 Hidden layer CNN	Data Validasi	85%	95.1%	95.6%	-
Proposed	ROI Segmentation, Speckle Noise Reduction	7 Hidden layer CNN	Data Validasi	93.9%	94%	93.9%	98.1%

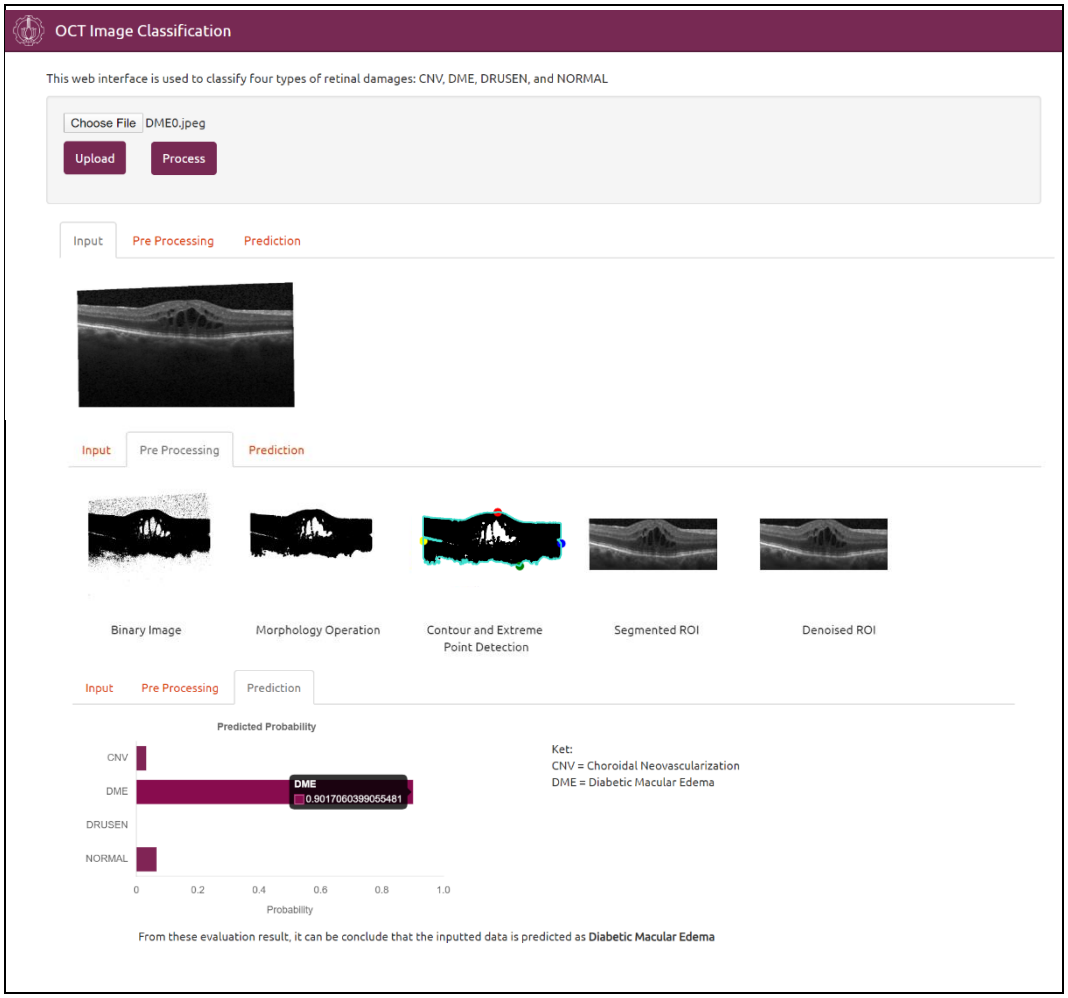
Inception pada penelitian [9] sudah mengalami beberapa perkembangan dan pembaruan dari mulai versi 1 hingga versi 4. Penyebab yang keempat adalah perbedaan perlakuan terhadap dataset dimana metode segmentasi ROI yang diimplementasikan masih menemui kegagalan segmentasi ROI pada citra. Sehingga kedepannya perlu dilakukan penelitian lebih lanjut mengenai metode lain untuk melakukan segmentasi ROI atau metode lain untuk meminimalisir kegagalan segmentasi. Penyebab yang kelima adalah perbedaan data pengujian antara penelitian yang dirancang dengan penelitian [10] dimana hal tersebut dapat menyebabkan perbedaan terhadap hasil uji performansi. Selanjutnya apabila dibandingkan dengan penelitian dari Najeeb dkk [30], penelitian yang dirancang memiliki nilai akurasi yang lebih tinggi namun memiliki nilai sensitivitas dan spesifisitas yang lebih rendah.

5.4 Uji Coba Sistem Secara Keseluruhan

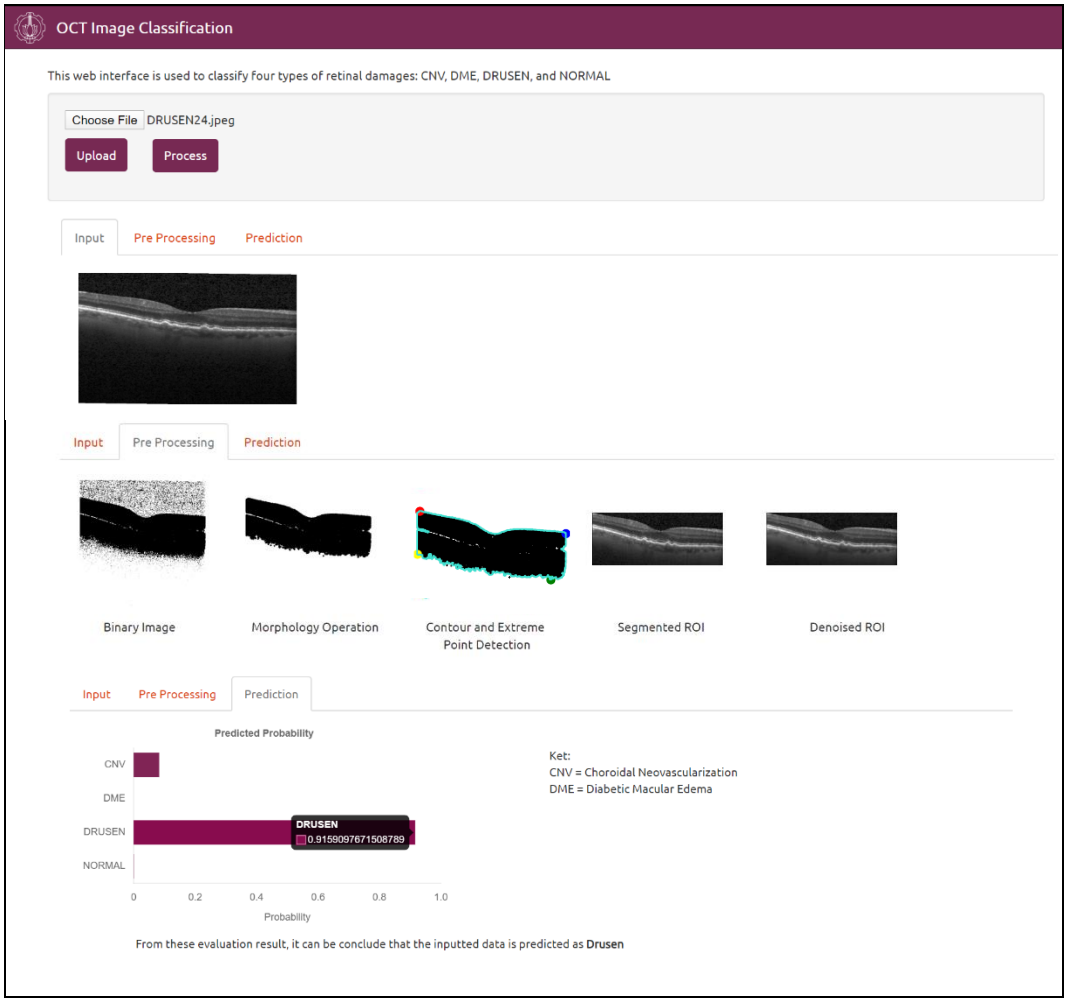
Uji coba sistem secara keseluruhan dilakukan dengan menjalankan *interface* berbasis web yang telah dibuat. Pada *interface* tersebut ditunjukkan mengenai tahapan pra pengolahan data beserta hasil prediksi probabilitas dari masing-masing kelas. Uji coba dilakukan de ngan cara memilih citra secara random untuk masing-masing kelas. Gambar 5.8 hingga 5.11 masing-masing menunjukkan hasil uji coba sistem pada citra CNV 0, DME 0, DRUSEN 24, dan NORMAL 227 dimana sistem dapat memprediksi kelas dengan prosentase probabilitas pada kelas masing-masing sebesar 94.1%, 90.17%, 91.6% dan 91.16%.



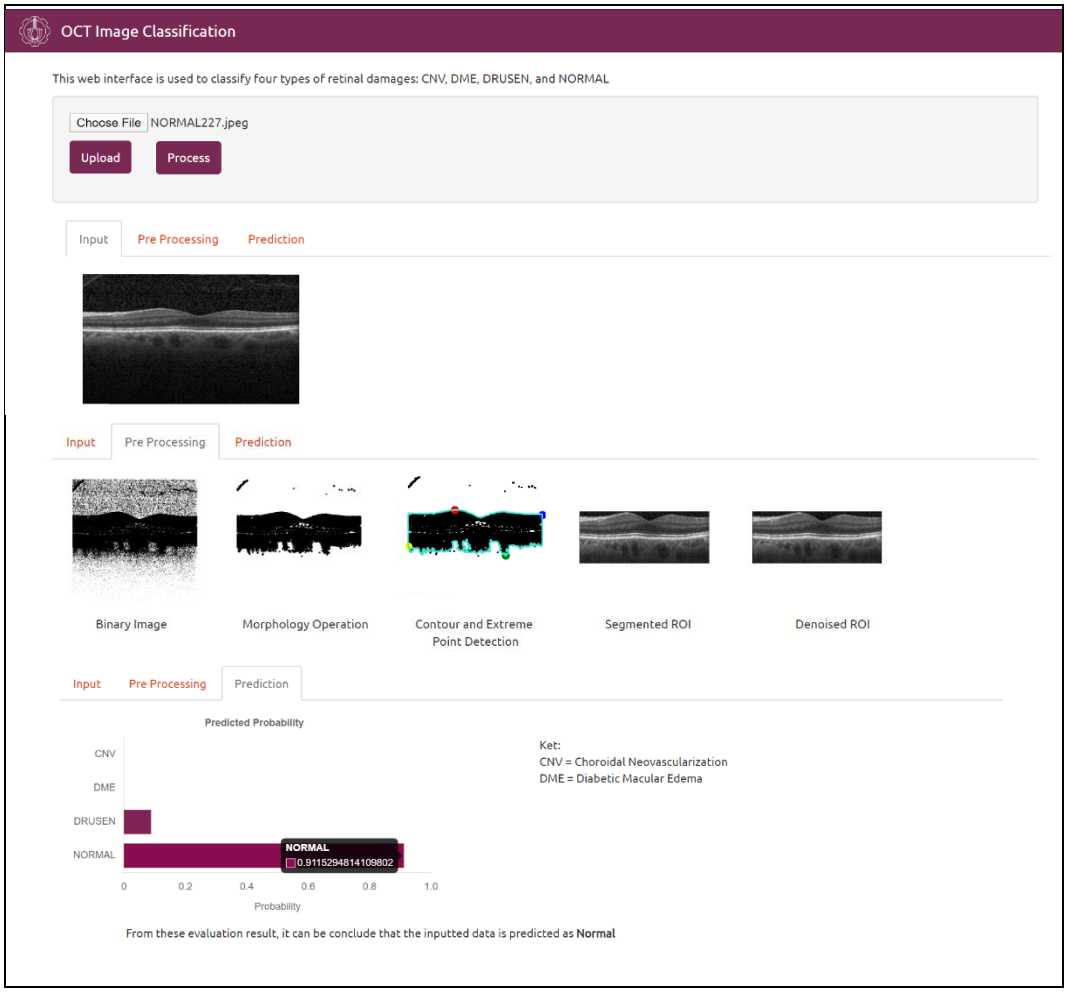
Gambar 5. 8 Uji coba pada citra CNV 0



Gambar 5. 9 Uji Coba pada citra DME 0



Gambar 5. 10 Uji Coba pada citra DRUSEN 24



Gambar 5. 11 Uji Coba pada citra NORMAL 227

BAB VI
PENUTUP

Bab ini membahas mengenai kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil pengujian yang sudah dilakukan dapat ditarik beberapa kesimpulan sebagai berikut:

1. Algoritma segmentasi ROI yang telah diimplementasikan memiliki prosentase tingkat kegagalan dalam mensegmentasi area ROI citra sebesar 0.016%. Kegagalan segmentasi yang terjadi dapat disebabkan oleh beberapa hal yaitu terdapat area pada *background* yang memiliki nilai piksel hampir sama dengan area ROI, kondisi citra yang terlalu gelap, serta terdapat *speckle noise* berlebih pada citra.
2. Implementasi reduksi noise didapatkan hasil yang baik pada jenis filter bilateral filter dengan parameter sigma range dan spatial masing-masing sebesar 0.5 dan 20 dimana didapatkan nilai MSE sebesar 0.00055 dan PSNR sebesar 32.56 Db
3. Klasifikasi jenis kerusakan retina menggunakan CNN pada kondisi 1 menghasilkan nilai akurasi sebesar 78.4%, rata-rata presisi 75.8%, rata-rata sensitivitas 62.6%, rata-rata spesifisitas 90.2%.
4. Klasifikasi jenis kerusakan retina menggunakan CNN pada kondisi 2 menghasilkan nilai akurasi sebesar 88.9%, rata-rata presisi 85.5%, rata-rata sensitivitas 86.4%, rata-rata spesifisitas 96.6%.
5. Klasifikasi jenis kerusakan retina menggunakan CNN pada kondisi 3 menghasilkan nilai akurasi sebesar 93.9%, rata-rata presisi 94%, rata-rata sensitivitas 93.9%, rata-rata spesifisitas 98.1%.
6. Klasifikasi jenis kerusakan retina menggunakan CNN pada kondisi 4 menghasilkan nilai akurasi sebesar 94.2%, rata-rata presisi 94.2%, rata-rata sensitivitas 94.2%, rata-rata spesifisitas 98.2%.
7. Tahapan *balancing data* yang dirancang, membuat jumlah data pada masing-masing kelas menjadi seimbang sehingga terjadi penurunan terhadap kasus *false negative* dan *false positive* yang berdampak pada peningkatan terhadap nilai akurasi, presisi, sensitivitas dan spesifisitas.
8. Pengujian dengan citra mentah menghasilkan hasil yang lebih baik daripada citra terpraproses. Hal tersebut dapat disebabkan karena masih terdapatnya citra yang mengalami kegagalan segmentasi pada citra terpraproses.
9. Pada tahapan *balancing data*, metode *random undersampling* menghasilkan hasil yang lebih baik daripada metode *weight balancing*.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem klasifikasi citra OCT 2D, yaitu:

1. Perlu dilakukan uji coba terhadap data primer
2. Perlu dilakukan uji coba terhadap metode lain untuk melakukan segmentasi ROI atau metode lain untuk meminimalisir kegagalan segmentasi
3. Perlu dilakukan tuning terhadap *hyperparameter* CNN seperti perbedaan penggunaan *learning rate,* banyaknya lapisan CNN, batch size, epoch, model optimasi dan lain sebagainya.

6.3 Ucapan Terima Kasih

Pada sub bab ini, penulis secara khusus menyampaikan ungkapan terima kasih kepada Daniel S. Kermany dkk [9] yang berkolaborasi dengan *Shiley Eye Institute of the University of California San Diego, the California Retinal Research Foundation, Medical Center Opthalmology Associates, the Sanghai First People's Hospital, dan Beijing Tongren Eye Center,* yang telah menghasilkan dataset citra *Optical Coherence Tomography* 2D sehingga penelitian pada tugas akhir ini dapat terlaksana.

DAFTAR PUSTAKA

[1] G. Parkash dan T. Iwata (Editor). 2017. *Advances in Vision Research, Volume 1*. Diakses 2 Nov 2019, dari Google Book.

[2] (2016) Global Report on Diabetes, 2016, World Health Organization. [Online].

[3] Katkar, Rujuta & Tadinada, Aditya & Amaechi, Bennett & Fried, Daniel. (2018). Optical Coherence Tomography. Dental clinics of North America. 62. 421-434. 10.1016/j.cden.2018.03.004.

[4] M. Naveed, A. Ramzan and M. U. Akram, "Clinical and technical perspective of glaucoma detection using OCT and fundus images: A review," 2017 IEEE of International Conference on Next Generation Computing Applications (NextComp), Mauritius, 2017, pp. 157-162.

[5] J. G Lee, R. B. Rosen "Learning to Read Retinal OCT", [Online]. Available:https://www.ophthalmologymanagement.com/issues/2015/july-2015/learning-to-read-retinal-oct. [Diakses: 2 Nov 2019]

[6] H. Bogunović et al., "RETOUCH: The Retinal OCT Fluid Detection and Segmentation Benchmark and Challenge," in IEEE Transactions on Medical Imaging, vol. 38, no. 8, pp. 1858-1874, Aug. 2019.

[7] Y. Cha and J. Han, "High-Accuracy Retinal Layer Segmentation for Optical Coherence Tomography Using Tracking Kernels Based on Gaussian Mixture Model," in IEEE Journal of Selected Topics in Quantum Electronics, vol. 20, no. 2, pp. 32-41, March-April 2014, Art no. 6801010.

[8] M. Awais, H. Müller, T. B. Tang and F. Meriaudeau, "Classification of SD-OCT images using a Deep learning approach," 2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), Kuching, 2017, pp. 489-492.

[9] D. S. Kermany, et al, "Identifying medical diagnoses and treatable diseases by image-based deep learning," Cell, vol. 172, no. 5, pp. 1122–1131.e9, Feb 2018.

[10] K. T. Islam, S. Wijewickrema and S. O'Leary, "Identifying Diabetic Retinopathy from OCT Images using Deep Transfer Learning with Artificial Neural Networks," 2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS), Cordoba, Spain, 2019, pp. 281-286, doi: 10.1109/CBMS.2019.00066.

[11] Gupta, Mrinali & Herzlich, Alexandra & Sauer, Theodor & Chan, Chi-Chao. (2015). Retinal Anatomy and Pathology. Developments in ophthalmology. 55. 7-17. 10.1159/000431128.

[12] Staurenghi, Giovanni; Sadda, Srinivas; Chakravarthy, Usha; Spaide, Richard F. (2014). "Proposed Lexicon for Anatomic Landmarks in

Normal Posterior Segment Spectral-Domain Optical Coherence Tomography". Ophthalmology. 121 (8): 1572–1578. Doi:10.1016/j.ophtha.2014.02.023. PMID 24755005.

[13] R. Prashanth, S. V. Paranjape, S. Ghosh, P. K. Dutta and J. Chatterjee, "Characterization of changes in Retinal Pigment Epithelium layer in Choroidal Neovascularization through analysis of Optical Coherence Tomographs," 2010 IEEE Students Technology Symposium (TechSym), Kharagpur, 2010, pp. 39-43.

[14] Gundogan, F. C., Yolcu, U., Akay, F., Ilhan, A., Ozge, G., & Uzun, S. (2016). Diabetic Macular Edema. Pakistan journal of medical sciences, 32(2), 505–510. Doi:10.12669/pjms.322.8496

[15] Silvestri G, Williams MA, McAuley C, Oakes K, Sillery E, Henderson DC, Ferguson S, Silvestri V, Muldrew KA (2012). "Drusen prevalence and pigmentary changes in Caucasians aged 18-54 years". Eye (Lond). 26 (10): 1357–62.

[16] Davis PL, Jay WM (December 2003). "Optic nerve head drusen". Semin Ophthalmol. 18 (4): 222–42

[17] Lengyel I, Flinn JM, Peto T, et al. (April 2007). "High concentration of zinc in sub-retinal pigment epithelial deposits". Exp. Eye Res. 84 (4): 772–80

[18] Dougherty, Geoff. (2009). Digital Image Processing for Medical Applications. Unites States of America: Cambridge University Press

[19] Rassovsky, George. 2014. *Cubical Marching Squares Implementation* [Thesis]. Inggris (EN): Bournemouth University

[20] Putra, Darma. (2010). Pengolahan Citra Digital. Yogyakarta: Penerbit ANDI.

[21] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand. 2008. *Bilateral Filtering: Theory and Applications.* Computer Graphics and Vision, Vol. 4, No.1, pp. 1-73. Doi:10.1561/0600000020

[22] N. Buduma and N. Locascio, Fundamentals of Deep Learning: Designing Next-generation Machine Intelligence Algorithms, O'Reilly Media, 2017, p. 283.

[23] M. A. Ponti, L. S. F. Ribeiro, T. S. Nazare, T. Bui and J. Collomosse, "Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask," in 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2017.

[24] Purnomo, M. H. & Kurniawan, A. 2006. Supervised Neural Networks dan Aplikasinya.Yogyakarta: Graha Ilmu.

[25] Agarap, A.F., "Deep Learning using Rectified Linear Units (ReLU)," 2018 arXiv:1803.08375 [cs, stat]

[26] D.P Kingma, J. Ba, “Adam: A Method for Stochastic Optimization,” 2014 arXiv:1412.6980 [cs. LG]

[27] A. Karpathy dan J. Johnson, C231n Convolutional Neural Networks for Visual Recognition, Stanford University, [Online]. Available: http://cs231n.github.io/. [Diakses 2 November 2019]

[28] I. W. Suartika, A. Y. Wijaya dan R. Soelaiman, Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101, Jurnal Teknik ITS, vol. 5, no. 1, pp. 65-69, 2016

[29] K. O’Shea, R. Nash, “An Introduction to Convolutional Neural Networks,” 2015 arXiv: 1511.08458 [cs.NE]

[30] S. Najeeb, N. Sharmile, M. S. Khan, I. Sahin, M. T. Islam and M. I. Hassan Bhuiyan, “Classification of Retinal Diseases from OCT scans using Convolutional Neural Networks,” 2018 10th International Conference on Electrical and Computer Engineering (ICECE), Dhaka, Bangladesh, 2018, pp. 465-468. Doi: 10.1109/ICECE.2018.8636699

[31] “About Python.” [Online]. Available: https://www.python.org/about/. [Diakses: 11-Mei-2020].

[32] “Scikit-image.” [Online]. Available: https://scikit-image.org/. [Diakses: 11 Mei 2020].

[33] "Tensorflow." [Online]. Available: https://www.tensorflow.org/. [Diakses: 11 Mei 2020].

[34] “Keras Documentation.” [Online]. Available: https://keras.io/. [Diakses: 11 Mei 2020].

[35] Grinberg, Miguel. 2014. *Flask Web Development.* Sebastopol:O’Reilly Media Inc.

BIOGRAFI PENULIS

Putri Norma Aprilia R, lahir pada 10 April 1998 di Kota Sidoarjo, Provinsi Jawa Timur. Pada tahun 2010 penulis menyelesaikan pendidikan di SD Negeri Cemeng Kalang. Tahun 2013 penulis lulus dari SMP Negeri 4 Sidoarjo serta pada tahun 2015 lulus dari SMA Negeri 3 Sidoarjo. Penulis diterima pada Program Studi S-1 Departemen Teknik Biomedik Fakultas Teknologi Elektro dan Informatika Cerdas ITS pada tahun 2016. Selama masa perkuliahan, penulis aktif dalam unit kegiatan mahasiswa robotika. Penulis tergabung dalam tim robot seni tari yang bernama VI-ROSE. Selama tergabung dalam tim, penulis berhasil meraih beberapa prestasi dalam kompetisi robot nasional dan internasional. Penulis dapat dihubungi melalui e-mail: putri.norma16@mhs.bme.its.ac.id.

BIOMEDICAL ENGINEERING ITS
2020