PENERAPAN *MULTI PLANAR RECONSTRUCTION SLICING* PADA ARSITEKTUR *VISUAL GEOMETRY GROUP*-16, *UNET*, DAN *INCEPTION* DALAM SEGMENTASI CITRA TIGA DIMENSI PADA HATI

SKRIPSI

Oleh :

Narti

08011282025061

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SRIWIJAYA

2024

LEMBAR PENGESAHAN

PENERAPAN *MULTI PLANAR RECONSTRUCTION SLICING* PADA ARSITEKTUR *VISUAL GEOMETRY GROUP*-16, *UNET*, DAN *INCEPTION* DALAM SEGMENTASI CITRA TIGA DIMENSI HATI

SKRIPSI

Sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Sains Bidang Studi Matematika

Oleh

NARTI
NIM. 08011282025061

Indralaya, Mei 2024

Pembimbing Kedua	Pembimbing Utama
Dr. Anita Desiani, S.Si., M.Kom.	**Dr. Bambang Suprihatin, S.Si., M.Si.**
NIP. 197712112003122002	**NIP. 197101261994121001**

Mengetahui,
Ketua Jurusan Matematika

Dr. Dian Cahyawati S. S.Si., M.Si.
NIP. 197303212000122001

SURAT PERNYATAAN KEASLIAN KARYA ILMIAH

Yang bertanda tangan dibawah ini :

Nama Mahasiswa : Narti
NIM : 08011282025061
Fakutlas/Jurusan : Matematika

Menyatakan bahwa skripsi ini adalah hasil karya saya sendiri dan karya ilmiah ini belum pernah diajukan sebagai penentuan persyaratan untuk memperoleh gelar kesarjanaan strata satu (S1) dari Universitas Sriwijaya maupun perguruan tinggi lain.

Semua informasi yang dimuat dalam skripsi ini berasal dari penulisan lain baik yang dipublikasikan atau tidak, telah diberikan penghargaan dengan mengutip nama sumber penulis secara benar. Semua isi dari skripsi sepenuhnya menjadi tanggung jawab saya sebagai penulis.

Demikianlah surat pernyataan ini saya buat dengan sebenarnya.

Indralaya, 05 Juni 2024

Narti
NIM. 08011282025061

HALAMAN PERSEMBAHAN

Kupersembahkan skripsi ini untuk:

Yang Maha Kuasa Allah Subhanahu Wa Ta'ala,

Kedua orang tuaku tersayang,

Kakak laki-lakiku,

Kakak perempuanku,

Keluarga besarku,

Semua guru dan dosenku,

Sahabat-sahabatku,

Almamaterku.

Moto

"Akhiri ketika sudah bukan ketika lelah"

KATA PENGANTAR

Puji syukur kehadirat Allah Subhanahu wa Ta'ala yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Penerapan *Multi Planar Reconstruction Slicing* pada Arsitektur *Visual Geometry Group*-16, *UNet*, Dan *Inception* dalam Segmentasi Citra Tiga Dimensi pada Hati" sebagai salah satu syarat memperoleh gelar sarjana sains bidang studi Matematika di Fakultas MIPA Universitas Sriwijaya.

Penulis menyadari bahwa proses pembuatan skripsi ini merupakan proses pembelajaran yang sangat berharga serta tak lepas dari kekurangan dan keterbatasan. Dengan segala hormat dan kerendahan hati, penulis mengucapkan terima kasih dan penghargaan kepada:

1. Kedua orang tuaku tercinta, **Abdul Hamid dan Siti Julaiha**, yang tak pernah lelah mendidik, menasehati, membimbing, dan mendukung serta terus mendoakan anaknya. Kakak-kakakku **Junaidi, Abdul Sani, Arianto, Siti Asiah, Siti Khadijah, serta keluarga besar** yang mendoakan serta memberikan semangat dan dukungan kepada penulis. Terima kasih atas segala perjuangan dan pengorbanan hingga detik ini dan sampai kapanpun.
2. Kepada semua keluarga yang berada di kota rantau terkhusus keluarga Ibu **Yenni Yustina**, Bapak **Suradi**, dan Bapak **Robert Pardamaian** yang telah menerima, membantu, dan mendukung penulis selama masa perkuliahan ini.
3. Bapak **Prof. Hermansyah, S.Si., M.Si., Ph.D** selaku Dekan Fakultas MIPA Universitas Sriwijaya.

4. Ibu **Dr. Dian Cahyawati Sukanda, M.Si** selaku Ketua Jurusan Matematika dan Ibu **Des Alwine Zayanti, S.Si., M.Si** selaku Sekretaris Jurusan Matematika FMIPA Universitas Sriwijaya serta dosen penguji yang telah mengarahkan urusan akademik kepada penulis selama proses perkuliahan.
5. Bapak **Dr. Bambang Suprihatinn, S.Si., M.Si.**, selaku dosen pembimbing utama, Ibu **Dr. Anita Desiani, S.Si., M.Kom** selaku dosen pembimbing kedua, dan Ibu **Dra. Ning Eliyati, M.Pd** selaku dosen pembahas dan penguji yang telah bersedia meluangkan waktu, tenaga, pikiran untuk memberikan bimbingan, pengarahan, dan didikan berharga selama proses perkuliahan ini.
6. Ibu **Dr. Fitri Maya Puspita, S.Si., M.SC** selaku dosen pembimbing akademik, **seluruh Dosen, Pak Irwansyah** selaku admin, dan **Ibu Hamidah** selaku pegawai tata usaha Jurusan Matematika FMIPA, dan **seluruh guru** yang telah memberikan bimbingan dan arahan dalam urusan akademik penulis.
7. **M. Owen** yang selalu menemani, menyemangati, serta membantu penulis menyelesaikan skripsi dan perkuliahan ini. **Widya, Adelia, Yulia, Nada, Nabila, Annisa, Thalia, Tarishah, Difa, Fitri, Shindy** selaku sahabat penulis yang telah mendukung, menemani, serta berbagi pikiran dengan penulis dalam berbagai hal termasuk perkuliahan.
8. **Semua sahabat seperjuangan sejak masa putih merah, putih biru, putih abu-abu**, **perkuliahan, Keluarga Matematika 2020, Komputasi 2020, komputasi 2021, Kakak Tingkat Angkatan 2018 dan Kakak Tingkat Angkatan 2019,** serta **Adik Tingkat Angkatan 2021 dan Adik Tingkat**

Angkatan 2022 yang telah memberikan semangat, dukungan, serta berbagi ilmu dan kebaikan selama ini.

Semua pihak yang tidak dapat penulis sebutkan satu per satu. Semoga segala kebaikan yang diberikan mendapatkan balasan terbaik dari Allah.

Semoga skripsi ini dapat menambah pengetahuan dan bermanfaat bagi mahasiswa/mahasiswi Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Sriwijaya dan seluruh pihak yang membutuhkan.

Indralaya, Mei 2024

Penulis

APPLICATION OF MULTI PLANAR RECONSTRUCTION SLICING IN VISUAL ARCHITECTURE GEOMETRY GROUP-16, UNET, AND INCEPTION IN THREE DIMENSIONAL IMAGE LIVER SEGMENTATION

By:

NARTI

08011282025061

ABSTRACT

The UNet architecture has been widely developed for image segmentation. UNet segmentation has been widely applied, one of which is on heart images. Segmentation of liver images is usually carried out on Abdominal Computed Tomography (CT) Scan images. The image from an abdominal CT scan is usually three-dimensional (3D) which focuses on organs including the liver, stomach, small intestine, large intestine, gallbladder, spleen, pancreas and kidneys. Unfortunately, 3D images are very complex and the U-Net architecture requires large amounts of data which can cause overvitting and suboptimal performance. A technique that can overcome the complexity of 3D images can be done using the MPR cutting technique. MPR slicing cuts 3D images into 2D based on the axial, sagittal and coronal axes. A new architectural combination is needed that can improve UNet performance. This research proposes a modification of the VGG-16, Inception, and UNet architecture. The VGG-16, Inception, and UNet architectures modify UNet by replacing the UNet encoder with a VGG-16 encoder to help extract more features. The VGG-16, Inception, and UNet architectures implement Inception on the bridge to help reduce the number of parameters. The application of the VGG-16, Inception, and UNet architecture to liver segmentation resulted in accuracy, sensitivity, specificity, F1-Score, and IoU values achieved above 95%. This shows that the performance of the VGG-16, Inception, and UNet architecture is very good in segmenting liver images. Based on the results of this performance evaluation, it can be concluded that the VGG-16, Inception, and UNet architectures are superior in segmenting liver CT scan images.

Keywords : Segmentation, Liver, UNet , Inception, VGG-16

PENERAPAN *MULTI PLANAR RECONSTRUCTION SLICING* PADA ARSITEKTUR *VISUAL GEOMETRY GROUP*-16, *UNET*, DAN *INCEPTION* DALAM SEGMENTASI CITRA TIGA DIMENSI HATI

Oleh:

NARTI

08011282025061

ABSTRAK

Arsitektur *UNet* banyak dikembangkan untuk segmentasi citra. Segmentasi *UNet* telah banyak diterapkan, salah satunya pada citra hati. Segmentasi pada citra hati biasanya dilakukan pada citra *Computed Tomography* (CT) Scan Abdomen. Citra hasil CT Scan Abdomen biasanya berbentuk tiga dimensi (3D) yang fokus pada organ-organ termasuk hati, lambung, usus halus, usus besar, kandung empedu, limpa, pankreas, dan ginjal. Sayangnya, citra 3D sangat kompleks dan arsitektur *UNet* mmebutuhkan jumlah data yang besar sehingga dapat menyebabkan terjadinya overvitting dan kinerja yang tidak optimal. Teknik yang dapat mengatasi kompleksitas citra 3D dapat dilakukan dengan teknik pemotongan MPR. MPR *slicing* memotong citra 3D menjadi 2D berdasarkan sumbu aksial, sagittal, dan koronal. Diperlukan suatu kombinasi arsitektur baru yang dapat meningkatkan kinerja *UNet*. Penelitian ini mengusulkan modifikasi arsitektur VGG-16, *Inception*, dan *UNet*. Arsitektur VGG-16, *Inception*, dan *UNet* memodifikasi *UNet* dengan mengganti encoder *UNet* ke encoder VGG-16 untuk membantu menggali fitur yang lebih banyak. Arsitektur VGG-16, *Inception*, dan *UNet* menerapkan *Inception* pada bagian bridge untuk membantu mengurangi jumlah parameter. Penerapan arsitektur VGG-16, *Inception*, dan *UNet* pada segmentasi hati menghasilkan nilai akurasi, sensitivitas, spesifisitas, F1-Score, dan IoU yang dicapai diatas 95%. Hal ini menunjukkan kinerja arsitektur VGG-16, *Inception*, dan *UNet* sangat baik dalam melakukan segmentasu citra hati. Berdasarkan hasil evaluasi kinerja ini, dapat disimpulkan bahwa arsitektur VGG-16, *Inception*, dan *UNet* unggul dalam melakukan segmentasi pada citra CT Scan hati.

Kata Kunci : Segmentasi, Hati, *UNet* , *Inception*, VGG-16

DAF TAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Arsitektur *UNet* merupakan arsitektur yang memiliki dua jalur, yang pertama *encoder* dan yang kedua adalah *decoder*. *UNet* terdiri dari banyak saluran fitur yang dapat membantu model dari arsitektur UNet menghasilkan akurasi yang lebih unggul (Du *et al*., 2020). Arsitektur *UNet* banyak dikembangkan untuk segmentasi citra. Arsitektur Segmentasi merupakan proses identifikasi fitur penting dari suatu citra untuk memisahkan fitur yang diambil dengan fitur lainnya (Desiani *et al*., 2022). Segmentasi *UNet* telah banyak diterapkan, salah satunya pada citra hati. Segmentasi pada citra hati biasanya dilakukan pada citra *Computed Tomography* (CT) *Scan Abdomen* (Chandra & Hati, 2022). Citra hasil CT *Scan* Abdomen biasanya berbentuk tiga dimensi (3D) yang fokus pada organ-organ termasuk hati, lambung, usus halus, usus besar, kandung empedu, limpa, pankreas, dan ginjal (Siregar *et al*., 2020).

Citra 3D lebih kompleks dibandingkan citra 2D karena citra 3D tidak hanya memiliki panjang dan lebar namun juga memiliki kedalaman. Penerapan arsitektur *UNet* pada segmentasi citra hati telah dilakukan oleh beberapa peneliti. Naraloka *et al*., (2022) telah menerapkan arsitektur *UNet* pada segmentasi citra hati dengan nilai akurasi dan spesifisitas yang sangat baik diatas 95%. Sayangnya sensitivitas yang dihasilkan hanya sebesar 89,84%. Zhou *et al*., (2018) menerapkan arsitektur *UNet* pada segmentasi hati memberikan nilai akurasi hanya sebesar 76,58%.

Kushnure & Talbar, (2021) menerapkan arsitektur *UNet* pada segmentasi hati menghasilkan *IoU* hanya sebesar 91%. Ketiga penelitian tersebut menggunakan citra yang telah berbentuk 2D tanpa menjelaskan teknik pengubahan citra 3D menjadi citra 2D.

Pengubahan citra 3D ke citra 2D harus memperhatikan setiap sudut pandang yang diambil dari 3D agar tidak ada informasi dari citra yang terlewatkan. Selain itu, data citra 3D CT Scan Abdomen hati masih terbatas. Sedangkan *UNet* sendiri dikenal sebagai arsitektur yang memiliki lapisan yang mendalam sehingga memerlukan jumlah data yang besar. Citra 3D lebih kompleks dibandingkan citra 2D dikarenakan citra 3D tidak hanya memiliki panjang dan lebar seperti citra 2D namun juga memiliki kedalaman. Salah satu teknik yang dapat dilakukan untuk menyederhanakan kompleksitas pada citra 3D adalah dengan mengubahnya menjadi 2D.

Salah satu solusi untuk melakukan pengubahan citra 3D menjadi citra 2D adalah dengan menerapkan teknik *slicing*. Teknik *slicing* adalah teknik pemotongan citra 3D dalam berbagai sumbu sehingga hasilnya berupa citra 2D (Yu *et al*., 2019). Pemotongan citra 3D menjadi citra 2D ini mengakibatkan jumlah data menjadi bertambah dan kompleksitas menjadi menurun. Salah satu teknik *slicing* yang diterapkan pada segmentasi citra medis adalah *multi planar reconstruction* (MPR) *slicing.*

Teknik MPR *slicing* merupakan teknik yang diterapkan untuk memotong citra 3D menjadi citra 2D berdasarkan tiga sumbu dari *dataset* yaitu aksial, sagital, dan koronal (Yu *et al*., 2019). Pemotongan berbagai bidang pada citra 3D berdasarkan

tiga sumbu tersebut dapat memberikan gambaran dari berbagai sudut pandang yang berbeda sehingga informasi yang ada dapat terwakili secara keseluruhan. Selain itu, hasil pemotongan ini dapat meningkatkan jumlah data dan mengurangi kompleksitas pada *UNet*. Selain memerlukan jumlah data yang banyak, *UNet* dikenal sebagai arsitektur yang mendalam dan memiliki kompleksitas yang cukup tinggi pada bagian *encoder* dan *decoder*. *Encoder* dan *decoder UNet* biasanya menggunakan *skip connections*. *Skip connections* menjalankan informasi dengan melewati satu atau beberapa lapisan yang ada (Zaeemzadeh & Member, 2021). Penerapan *skip connections* yang melewatkan lapisan-lapisan yang ada pada arsitektur *UNet* dapat menyebabkan struktur dari *UNet* menjadi lebih dangkal sehingga proses penangkapan informasi dan ekstraksi fitur menjadi lebih sulit (Wu *et al*., 2020).

Arsitektur CNN yang memiliki lapisan yang mendalam tanpa melibatkan *skip connections* adalah *Visual Geometry Group*-16 (VGG-16). VGG-16 memiliki lapisan yang lebih mendalam dibandingkan *UNet* sehingga VGG-16 dapat mengekstrak fitur-fitur penting lebih baik (Mei *et al*., 2021). VGG-16 menggunakan kernel yang relatif kecil yaitu 3x3 sehingga dapat menangkap fitur lebih banyak. VGG-16 telah banyak dikombinasikan dengan *UNet* untuk segmentasi hati. Ibrahim *et al*., (2022) menerapkan VGG-16 pada segmentasi hati sayangnya nilai sensitivitas yang dihasilkan hanya sebesar 58%. Elnakib *et al*., (2020) menerapkan VGG-16 pada segmentasi hati dengan menerapkan pemotongan hanya pada sumbu aksial. Sayangnya, akurasi yang diperoleh hanya sebesar 81%.

Meskipun arsitektur yang memiliki lapisan yang mendalam dapat menggali fitur dengan detail tetapi arsitektur dengan lapisan yang mendalam juga memiliki kelemahan yaitu dapat menyebabkan jumlah parameter meningkat. Jumlah parameter yang terlalu banyak dapat menimbulkan resiko *overfitting* (Khan *et al*., 2021). *Overfitting* merupakan kondisi dimana model hanya dapat mempelajari data latih tetapi tidak bisa mempelajari data baru (Abrar *et al*., 2023). Arsitektur yang dikenal memiliki parameter yang lebih kecil adalah *Inception.* Arsitektur *Inception* memanfaatkan ukuran kernel berbeda-beda yang dapat membantu mengurangi jumlah parameter yang didapatkan dari model (J. Wang *et al*., 2021). *Inception* telah diterapkan pada beberapa penelitian dalam segmentasi hati. Z. Zhang *et al*., (2023) melakukan penelitian dengan menggabungkan arsitektur *Inception* dan *UNet* namun sensitivitas yang dihasilkan hanya sebesar 81%. Selain itu, Özcan *et al*., (2023) melakukan penelitian dengan menerapkan *Inception* pada arsitektur *UNet* bagian bridge menghasilkan nilai akurasi dan *IoU* diatas 95% namun tidak mengukur kinerja lainnya serta tidak menjelaskan sumbu pemotongan yang digunakan.

Penelitian ini menggunakan dua tahapan yaitu menerapkan teknik MPR *slicing* dan segmentasi hati. Teknik *slicing* akan diterapkan pada tahap *preprocessing*. Pada tahap MPR *slicing* akan dilakukan pemotongan pada tiga sumbu yaitu aksial, koronal, dan sagital. Hal tersebut dilakukan agar informasi dari masing-masing sumbu dapat terwakili. Pada tahap segmentasi akan dilakukan modifikasi arsitektur *UNet* dengan mengganti bagian *encoder UNet* dengan arsitektur VGG-16 dan bagian bridge dengan *Inception*. Kombinasi arsitektur yang diusulkan ini dievaluasi

dengan kinerja yang dapat diukur berdasarkan nilai akurasi, sensitivitas, spesifisitas, *IoU*, dan *F1-Score* untuk melihat seberapa akuratnya arsitektur yang diusulkan pada segmentasi hati.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini yaitu bagaimana hasil penerapan teknik MPR *slicing* dan modifikasi arsitektur VGG-*16, Inception,* dan *UNet* dalam segmentasi hati pada citra hasil *CT Scan* dengan melakukan pengukuran evaluasi kinerja arsitektur berdasarkan nilai akurasi, sensitivitas, spesifisitas, *IoU,* dan *F1-Score.*

1.3 Pembatasan Masalah

Penelitian ini memiliki batasan masalah sebagai berikut :

1. Penelitian ini terbatas pada data citra 2D hasil MPR *slicing.*
2. Penelitiaan ini mengukur evaluasi kinerja model berdasarkan nilai akurasi, sensitivitas, spesifisitas, *IoU*, dan *F1-Score.*

1.4 Tujuan

Penelitian ini bertujuan untuk menerapkan teknik MPR *slicing* dan modifikasi arsitektur VGG-*16, UNet,* dan *Inception* dalam memperoleh hasil segmentasi hati pada citra *CT Scan* dengan mengukur hasil kinerja model berdasarkan akurasi, sensitivitas, spesifisitas*, IoU,* dan *F1-Score.*

1.5 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut :

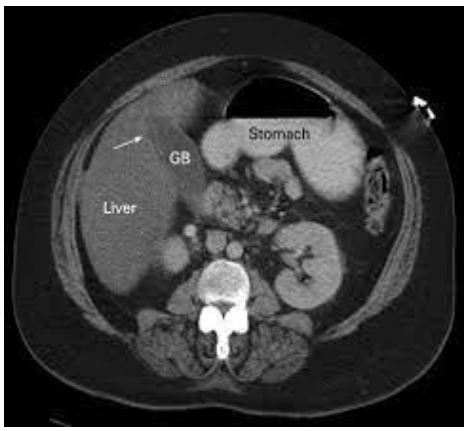
1. Membantu para ahli medis dalam proses identifikasi hati pada citra *CT Scan* secara otomatis.
2. Dapat digunakan sebagai referensi untuk penelitian pada bidang segmentasi, khususnya dalam segmentasi hati.

BAB II

TINJAUAN PUSTAKA

2.1 Hati

Hati merupakan organ yang sangat penting dan sebagai pusat dalam proses metabolisme tubuh. Hati berperan untuk menerima, memproses dan menyimpan makanan sebagai sumber nutrisi (Almotairi *et al*., 2020). Hati berperan dalam proses penyaringan sel darah dari saluran pencernaan yang dialirkan ke organ tubuh yang lainnya (Naraloka *et al*., 2022). Citra hati pada *CT Scan* memiliki bentuk, lokasi, serta tekstur yang berbeda-beda di setiap citra. Selain itu, kualitas hati yang normal memiliki perbedaan intensitas warna yang tipis (Syakrani *et al*., 2018). Hasil CT *Scan* untuk melihat morfologi hati dapat dilihat pada Gambar 2.1 berikut.



Gambar 2.1 Hasil CT *Scan*

2.2 CT *Scan*

CT *Scan* merupakan alat kesehatan yang biasa dimanfaatkan untuk menampilkan gambar bagian tubuh menggunakan sinar-X dengan kerja komputer.

Gambar yang didapatkan membantu ahli radiologi mengamati bagian dalam tubuh pasien (Lubis, 2020). Pada pemeriksaan *CT scan* dibutuhkan pengurangan dosis dan peningkatan kualitas citra yang dihasilkan, hal ini berkaitan dengan proteksi terhadap pasien dan peningkatan kualitas diagnosa yang didapatkan. *CT Scan* menghasilkan gambar dengan resolusi yang baik dan akurat dari berbagai sudut. Radiasi sinar-X yang dipaparkan dalam prosedur CT Scan kurang lebih sebesar 4% dari total radiasi sinar-X. Rasiasi yang dipaparkan muncul disebabkan pergerakan pasien selama perekaman *CT Scan*, penggunaan amalgam atau tambalan gigi berbahan logam oleh pasien, atau kondisi jaringan tertentu dapat menyebabkan munculnya artefak atau gambar yang seharusnya tidak ada tetapi direkam (Lubis, 2020).

2.3 Segmentasi Citra Biner

Segmentasi adalah proses penting dalam pengolahan citra yang bertujuan untuk membagi menjadi beberapa segmen dengan kriteria tertentu (Dalimunthe, 2020). Citra biner merupakan citra digital dengan dua kemungkinan nilai piksel yaitu hitam dan putih (Fernando Ade Pratama *et al*., 2022). Segmentasi citra adalah proses untuk menempatkan label untuk setiap piksel dalam sebuah gambar sehingga piksel berada pada pangsa label yang sama dengan karakteristik visual tertentu (Fernando Ade Pratama *et al*., 2022). Proses segmentasi citra membuat setiap objek dalam citra diidentifikasi secara terpisah sehingga dapat digunakan sebagai masukan (Dalimunthe, 2020), sehingga segmentasi pada suatu citra dapat membagi daerah citra hati dan daerah bukan hati (Armansyah, 2022).

2.4 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah sebuah sistem pengolahan objek dengan pengenalan citra. Metode CNN memiliki hasil yang sangat signifikan karena metode CNN meniru sistem pengenalan citra seperti pengolahan citra yang dilakukan manusia. CNN banyak digunakan untuk mengenali suatu objek dan diharapkan dapat mempermudah kerja manusia serta dapat menghemat waktu yang digunakan. CNN terbukti efektif karena telah banyak penelitian yang mengungkapkan keefektifan pengolahan citra dengan hasil akurasi diatas 70%. Selain itu, CNN menjadi salah satu metode yang populer dan banyak digunakan untuk penelitian (Ibnul Rasidi *et al*., 2022).

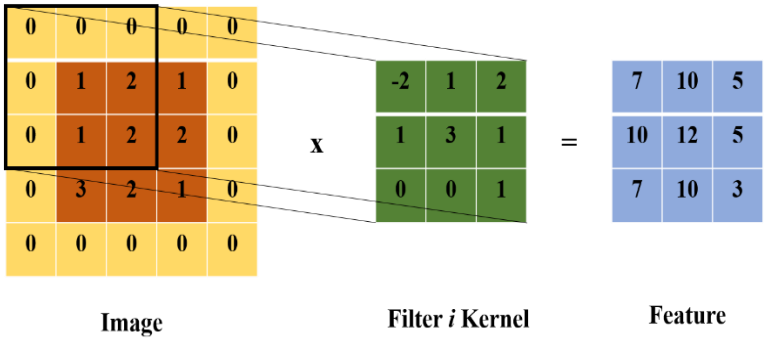
2.4.1 Convolution Layer

Convolutional layer digunakan untuk mengekstraksi fitur dari gambar dengan menerapkan operasi konvolusi sebagai pengganti operasi perkalian matriks yang biasa digunakan dalam jaringan saraf tradisional, untuk mempelajari hubungan antara lapisan input dan output. Pembagian parameter dalam operasi konvolusi memungkinkan jaringan mempelajari satu set parameter yang mengurangi jumlah parameter keseluruhan dan secara signifikan meningkatkan efisiensi waktu dan penggunaan memori (Sanjaya & Ayub, 2020). Perhitungan matrik *output* dari konvolusi dapat dihitung menggunakan persamaan (2.1).

$$D = (P_r \cdot K_s) + b_s \tag{2.1}$$

Pada persamaan (2.1) D merupakan *output feature map,* P adalah matriks input, K matriks kernel, dan b adalah bias, dengan $r = 1,2, \dots, m$ dengan r

merupakan jumlah input dan $s = 1,2,\dots,n$, dengan s merupakan jumlah filter dan · merupakan operator konvolusi.



Gambar 2.2 Proses Konvolusi

Pada gambar 2.2 terdapat *input* citra dengan matriks 3 × 3 yang diberi *padding* dengan *zero padding* pada lapisan terluar yang membuat ukuran dari citra tersebut menjadi matriks 5 × 5 yang akan melalui proses *convolution* oleh filter yang memiliki matriks berukuran 3 × 3 dan menghasilkan *feature map* dengan matriks 3 × 3. Dari gambar tersebut dapat diketahui bahwa *padding* dapat membuat *output* atau hasil pada *feature map* memiliki ukuran yang tetap sama dengan citra pada *input.* Proses perhitungan operasi konvolusi pada *convolution layer* pada setiap entri matriks dapat menggunakan persamaan (2.2) (Chen *et al*., 2020).

$$d_{ij} = \left(\sum_{u=0}^{m-1}\sum_{v=0}^{n-1}(b_{u+i\,v+j} \cdot k_{u+i\,v+j})\right) + b_s \qquad (2.2)$$

Untuk i = 1,2, … , n dan j = 1,2, … , n, dimana d_{ij} adalah entri matriks hasil operasi convolution pada baris ke-i dan kolom ke j, $b_{u+i\,v+j}$ adalah entri matriks *input* baris ke-$u + i$ kolom ke-$v + j$, dan $k_{u+i\,v+j}$ adalah entri matriks *kernel* baris ke-$u + i$ kolom ke-$v + j$, dan b_s adalah bias untuk kernel ke-s.

2.4.2 Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang digunakan pada jaringan saraf untuk mengaktifkan atau tidak mengaktifkan *neuron*. *Rectified Linear Unit* (*ReLU*) merupakan fungsi aktivasi *non-linear* yang diterapkan dalam *deep neural network* (Sanjaya & Ayub, 2020). Fungsi aktivasi *RELU* didefinisikan sebagai:

$$g_{ij} = max\left(0, d_{ij}\right) = \begin{cases} 0, & d_{ij} < 0 \\ d_{ij}, & d_{ij} \geq 0 \end{cases} \quad (2.3)$$

Menurut persamaan g_{ij}, hasil dari fungsi aktivasi *ReLU* adalah angka terbesar diantara nilai nol dan nilai *input*. Ketika *input* bernilai negatif, *output* yang dihasilkan akan nol, sementara ketika *input* positif, *output* yang dihasilkan akan sama dengan nilai *input* tersebut.

2.4.3 Batch Normalization

Batch Normalization merupakan metode yang digunakan untuk menormalkan aktivasi pada lapisan jaringan saraf. *Batch normalization* diterapkan untuk memaksimalkan pelatihan dan meminimalisir kemungkinan akan terjadinya *overfitting*. *Batch normalization* dapat meningkatkan akurasi dan mempercepat *training* (Bjorck *et al*., 2018). *Batch normalization* akan menghitung nilai rata-rata (μ_j) dan variansi (σ_j^2) dimasing-masing *mini batch* terlebih dahulu untuk melakukan tahap normalisasi. Perhitungan rata-rata pada setiap *mini batch* dilakukan dengan menggunakan Persamaan (2.4), sedangkan perhitungan variansi dapat dilakukan dengan Persamaan (2.5) (Kalayeh & Shah, 2019).

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} d_{ij} \quad (2.4)$$

$$\sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(d_{ij} - \mu_j)^2 \quad (2.5)$$

i adalah banyaknya data dalam sebuah *mini batch* untuk i = 1,2, … , n, j adalah jumlah mini batch, serta d_{ij} adalah entri matriks input. , x_{ij} dinormalisasi apabila sudah mendapatkan rata-rata serta variansi dengan Persamaan (2.6):

$$x_{ij} = \frac{d_{ij}-\mu_j}{\sqrt{\sigma_j^2+\varepsilon}} \quad (2.6)$$

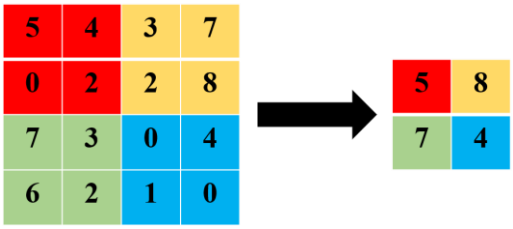
dimana x_{ij} adalah entri matriks *input* yang sudah dinomalisasi dan ε merupakan nilai galat error.

2.4.4 Dropout

Dropout diterapkan saat memilik sejumlah *neuron* secara acak pada model yang *overfitting*, kemudian melatih sub-model dengan mengabaikannya pada setiap *batch* pelatihan (Garbin *et al*., 2020).Teknik ini bertujuan untuk mencegah terjadinya *overfitting* saat proses pelatihan (C. Wang *et al*., 2020).

2.4.5 Max Pooling

Max pooling merupakan *pooling* paling umum yang digunakan untuk mengambil nilai terbesar dari setiap bagian yang dipilih. *Max pooling* digunakan untuk mencegah *overfitting* dengan memberikan representasi yang lebih abstrak. *Max pooling* mampu mengurangi biaya komputasi karena mampu mengecilkan parameter yang harus dipelajari dari layer sebelumnya (Sanjaya & Ayub, 2020).

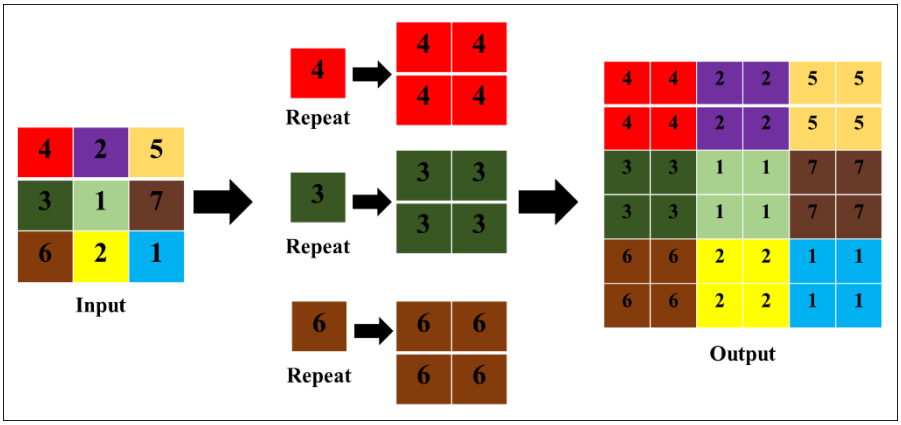


Gambar 2.3 Proses Operasi Max Pooling

Pada Gambar 2.3 terlihat bahwa operasi *max pooling* dilakukan pada matriks *input* berdimensi 4 × 4 yang dibagi menjadi empat submatriks menggunakan *stride* sebesar 2. Submatriks tersebut diberi warna merah, kuning, hijau, dan biru. Nilai maksimum dalam submatriks merah adalah 5, sementara dalam submatriks kuning adalah 8. Submatriks hijau memiliki nilai maksimum 7, sedangkan submatriks berwarna biru memiliki nilai maksimum 4. Nilai maksimum dari setiap submatriks digunakan untuk membentuk matriks *output* berukuran 2 × 2.

2.4.6 Upsampling Layer

Upsampling mengembalikan ukuran asli suatu citra dengan memperbesar ukuran citra tersebut (Sun & Chen, 2020). Dapat dilihat contoh operasi *upsampling* pada Gambar 2.4 berikut.

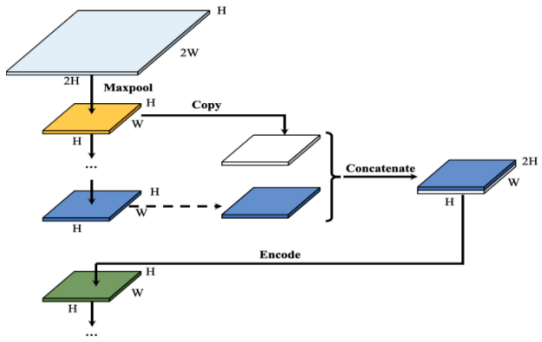


Gambar 2.4 Proses Operasi *Upsampling Layer*

Gambar 2.4 menunjukkan bahwa sebuah matriks *input* yang berdimensi 3 × 3 diperlakukan dengan lapisan *upsampling* menggunakan filter berukuran 2 × 2. Setiap nilai dalam matriks *input* diperbanyak sebagai submatriks berukuran 2 × 2, kemudian seluruh submatriks digabungkan menjadi matriks *output* yang dengan ukuran 6 × 6.

2.4.7 Concatenate Layer

Concatenate berfungsi agar dapat menyatukan dua matriks masukan menjadi satu matriks masukan baru dengan dimensi yang diperbarui (C. Wang *et al*., 2020). Ilustrasi penerapan *Concatenate layer* diberikan pada Gambar 2.5.



Gambar 2.5 Proses *Concatenate Layer*

Gambar 2.5 dapat dilihat ilustrasi *Concatenate layer* yang dilakukan di 2 *feature map* dengan melakukan penyalinan matriks yang telah mengalami konvolusi yang berukuran H × W dijalur *encoder* dan digabungkan bersama matriks hasil konvolusi kerukuran sama H × W pada jalur *decoder.* Hasil yang diperoleh yaitu matriks berukuran 2H × W.

2.4.8 Fungsi Aktivasi *Sigmoid*

Fungsi aktivasi *Sigmoid* pada kelas biner memiliki nilai pada range 0 sampai 1 didefinisikan sebagai berikut:

$$s_{ij} = \frac{1}{1+e^{-d_{ij}}} \quad (2.7)$$

s_{ij} menunjukkan masukkan data yang sesuai ke dalam matriks pada baris ke-i kolom ke-j sesuai dengan label yang benar entri matriks dimana $d_{ij} \in (-\infty, \infty), s_{ij} \in (0,1)$.

2.4.9 Loss Function: Binary Cross-entropy

Loss function adalah metrik untuk menilai kesalahan selama proses pelatihan yang membandingkan nilai prediksi dengan nilai sebenarnya. Untuk segmentasi biner dengan dua kelas objek, digunakan *binary cross entropy* sebagai *loss function* (Naraloka *et al*., 2022). Secara matematis, perhitungan fungsi *loss binary cross entropy* dapat dilakukan dengan menggunakan Persamaan (2.8) berikut (Jadon, 2020).

$$\mathrm{L} = -\frac{1}{m \times n}\left[\sum_{i=1}^{m}\sum_{j=1}^{n}\left(b_{ij}\log(s_{ij})\right) + (1-b_{ij})(1-\log(s_{ij}))\right] \quad (2.8)$$

dengan L merupakan nilai *binary cross-entropy.* s_{ij} menunjukkan entri matriks label *ground truth* baris ke-i kolom ke-j, b_{ij} merupakan entri matriks hasil prediksi baris ke-i kolom ke-j, serta m menunjukkan jumlah baris, n jumlah kolom matriks pada *binary cross-entropy.*

2.4.10 Optimization Function: Adaptive Moment Estimation (Adam)

Optimizer function digunakan untuk meningkatkan pembelajaran dengan memperbarui pengetahuan di jaringan untuk mengambil data input. *Adam* adalah salah satu algoritma pengoptimalan mutakhir yang digunakan oleh banyak praktisi pembelajaran mesin (Wirapati *et al*., 2022). *Adam* membuat

ukuran parameter baru tidak berubah ketika proses dioperasi ulang, ukuran langkah secara kasar dibatasi oleh *hyperparameter* ukuran langkah. Untuk menghitung nilai momen pertama pada *epoch* ke-s (m_s) dapat dilakukan dengan menerapkan Persamaan (2.9) dan momen kedua *epoch* ke-s (v_s) dapat menerapkan Persamaan (2.10) (Iqbal, 2022).

$$m_s = \beta_1 m_{s-1} + (1 - \beta_1) g_s \tag{2.9}$$

$$v_s = \beta_2 v_{s-1} + (1 - \beta_2) g_s^2 \tag{2.10}$$

β_1 dan β_2 merupakan *exponential decay rates* pertama dan kedua. m merupakan momen pertama dan v sebagai momen kedua. Perhitungan nilai penduga untuk momen pertama ($\widehat{m_s}$) dan penduga untuk momen kedua ($\widehat{v_s}$) diperoleh melalui tahap momen yang ada pada Persamaan (2.11) dan (2.12) berikut (Iqbal, 2022):

$$\widehat{m_s} = \frac{m_s}{1-\beta_1^s} \tag{2.11}$$

$$\widehat{v_s} = \frac{v_s}{1-\beta_2^s} \tag{2.12}$$

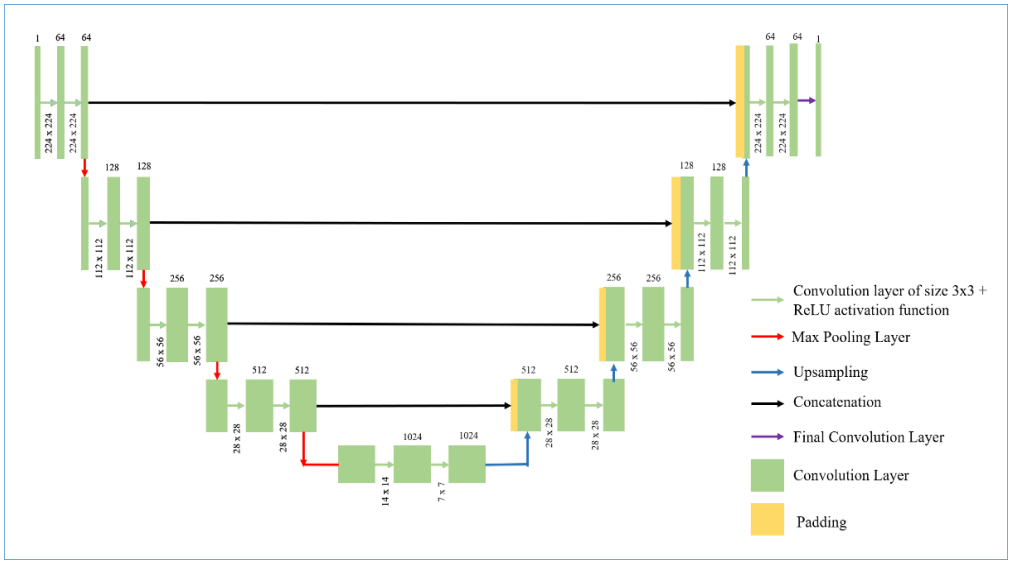
Memperbarui nilai bobot *epoch* ke-s (w_s) menggunakan *learning rate* senilai η dengan Persamaan (2.13).

$$w_s = w_{s-1} - \eta \frac{\widehat{m_s}}{\sqrt{\widehat{v_s}+\varepsilon}} \tag{2.13}$$

2.5 *UNet*

Arsitektur *UNet* terbagi menjadi dua komponen utama, yakni *encoder* dan *decoder*. Proses *encoder* bertujuan untuk mereduksi dimensi dari matriks *input*, sementara *decoder* bertanggung jawab untuk mengembalikan dimensi

matriks ke ukuran asal dengan meminimalkan jumlah peta fitur, sehingga memungkinkan segmentasi gambar. *Encoder* berperan dalam mengekstraksi informasi fitur dari citra masukan, sedangkan *decoder* berfungsi untuk mengekstraksi informasi fitur dari keluaran *encoder* dan sebagai tempat di mana segmentasi hasilnya dikeluarkan (Naraloka *et al*., 2022).

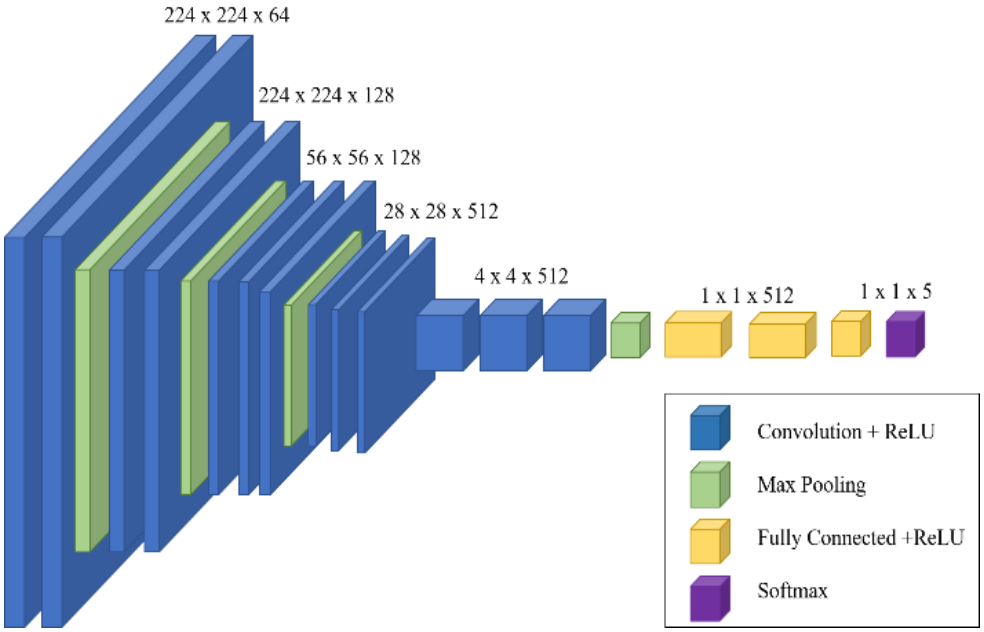


Gambar 2.6 Arsitektur *UNet*

Gambar 2.6 merupakan ilustrasi arsitektur *UNet* yang terdapat *input* gambar dipisah menjadi dua bagian, yaitu bagian *encoder* dan bagian *decoder*. Tiap blok dalam bagian *encoder* terdiri dari dua konvolusi berukuran 3 × 3, diikuti dengan aktivasi *ReLU* dan proses *max pooling*. Di jalur *decoder*, ukuran peta fitur ditingkatkan dari hasil *max pooling* dengan menggunakan *upsampling* dengan matriks berukuran 2 × 2. *Feature map* dari bagian *encoder* kemudian disatukan dengan layer pada pathway decoder melalui operasi concatenate. Setelah itu, proses konvolusi dilakukan lagi dengan matriks berukuran 3 × 3 yang diikuti oleh aktivasi ReLU.

2.6 *VGG-16*

VGG merupakan suatu arsitektur *deep learning* yang biasa diterapkan pada *ImageNet*, VGG merupakan singkatan dari *Visual Geometry Group*. Proses ekstraksi fitur dari VGG-16 yang dilakukan pada lapisan konvolusional model VGG-16 yang terdiri dari lima blok lapisan yang terdiri dari *convolutional layer* dengan fungsi aktivasi *ReLU* dan *max pooling* (Atabansi *et al*., 2021).

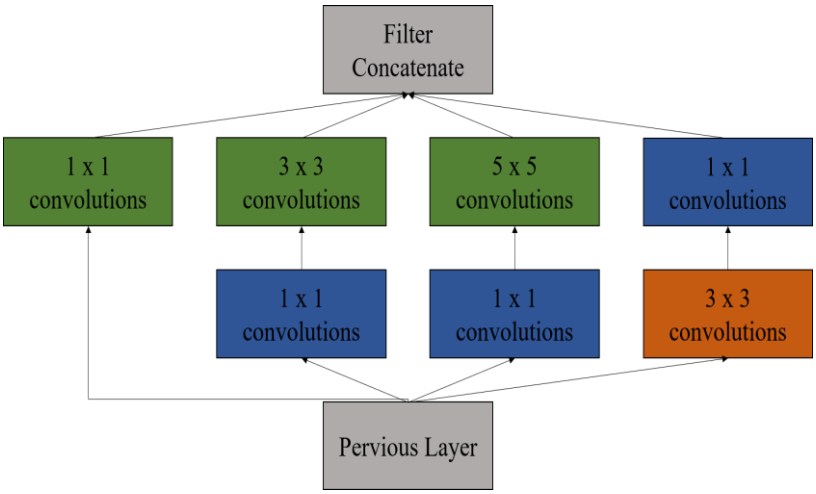


Gambar 2.7 Arsitektur VGG-16

Pada Gambar 2.7 menggambarkan struktur arsitektur VGG-16 yang terdiri dari 13 lapisan konvolusi. Langkah pertama melibatkan pengolahan data masukan, kemudian dilanjutkan dengan konvolusi pada setiap lapisan yang diikuti dengan operasi *maxpooling*. VGG-16 menerapkan kernel dengan matriks berukuran 3 × 3 dalam konvolusi dan kernel dengan matriks berukuran 2 × 2 dalam *maxpooling*.

2.7 Inception

Inception terletak pada beberapa kernel dengan ukuran berbeda secara paralel dengan cara yang efisien (Ansari *et al*., 2021). *Inception* menangkap konvolusi dengan besaran yang berbeda-beda yaitu 1 × 1, 3 × 3, 5 × 5 (Ansari *et al*., 2021). Modul *Inception* dapat dilihat pada Gambar 2.8.



Gambar 2.8 *Inception Module*

Berdasarkan Gambar 2.8 modul *Inception* memiliki empat varian yang berbeda untuk memproses input yang sama. Salah satunya adalah dengan melakukan penyaringan input menggunakan konvolusi 1 × 1 pada bagian pertama bermanfaat untuk memperkecil dimensi dan menggabungkan informasi dari jalur sebelumnya. Pada bagian kedua serta ketiga juga menerapkan konvolusi kernel 1 × 1 agar dapat mengurangi dimensi yang dilanjutkan lapisan konvolusi kernel ukuran 3 × 3 dan 5 × 5. Bagian keempat menerapkan *max-pooling* serta konvolusi untuk kernel 1 × 1 dan 3 × 3. *Output* dari masing-masing bagian sebelumnya disatukan lalu diteruskan sebagai *input* ke lapisan selanjutnya.

2.8 *Confusion Matrix*

Confusion matrix adalah teknik yang digunakan untuk mengukur tingkat keakuratan dalam data mining dengan memecah prediksi menjadi beberapa kategori, termasuk *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). *True positive* merujuk pada prediksi *positive* yang tepat yang dibuat oleh model, *True negative* dengan prediksi *negative* yang tepat yang dibuat oleh model (Wahyudi *et al*., 2019). *False positive* merupakan prediksi *negative* yang salah yang dibuat oleh model, serta *false negative* merupakan prediksi *positive* yang salah yang dibuat oleh model (Wahyudi *et al*., 2019).

Tabel 2.1. *Confusion Matrix*

Kelas		Nilai Prediksi	
		Positive	*Negative*
Nilai Aktual	*Positive*	*True Positive* (TP)	*False Negative* (FN)
	Negative	*False Positive* (FP)	*True Negative* (TN)

Berdasarkan *confusion matrix* evaluasi kerja data pada segmentasi ini akan diukur berdasarkan nilai akurasi, sensitivitas, spesifisitas, *IoU,* dan *F1-Score.* Akurasi merupakan nilai ketepatan berdasarkan nilai yang diukur dengan nilai sesungguhnya untuk mengukur tingkat keakuratan (Brilian Argario *et al*., 2018). Nilai akurasi dapat dihitung dengan menggunakan Persamaan (2.14).

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad (2.14)$$

Sensitivitas sebagai rasio sampel yang diprediksi termasuk dalam suatu kelas dengan semua sampel yang benar-benar termasuk dalam kelas tersebut. Sensitivitas memperkirakan kemampuan model untuk memprediksi sisi *positive* sebenarnya dari setiap kelas (Balaha & Hassan, 2023). Nilai sensitivitas dapat dihitung dengan menggunakan Persamaan (2.15).

$$\text{Sensitivitas} = \frac{TP}{TP + FP} \times 100\% \quad (2.15)$$

Spesifisitas memperkirakan kemampuan model untuk memprediksi nilai *negative* sebenarnya dari setiap kelas (Balaha & Hassan, 2023). Nilai spesifisitas dapat dihitung dengan menggunakan Persamaan (2.16).

$$\text{Spesifisitas} = \frac{TP}{TP + FN} \times 100\% \quad (2.16)$$

IoU merupakan nilai yang mengevaluasi keakuratan sistem dalam mendeteksi objek pada *dataset* yang telah dilatih (G. Zhang *et al*., 2019). *IoU* membandingkan *ground truth* dengan hasil prediksi segmentasi dari model (Kusuma *et al*., 2021). Nilai *IoU* dapat dihitung dengan menggunakan Persamaan (2.17).

$$IoU = \frac{TP}{TP + FP + FN} \times 100\% \quad (2.17)$$

F-1 Score merupakan perbandingan rata-rata dari nilai presisi dan nilai *recall*. *F-1 Score* memiliki nilai tertinggi sebesar 1 dan terendah sebesar 0. Nilai *F-1 Score* yang semakin mendekati 1 menunjukkan bahwa maka semakin baik pula kinerja model yang digunakan dalam melakukan segmentasi (Kusuma *et al*., 2021). Nilai *F-1 Score* dapat dihitung dengan menggunakan Persamaan (2.18).

$$F\text{-}1\ Score = \frac{2TP}{2TP + FP + FN} \times 100\% \quad (2.18)$$

Nilai akurasi, sensitivitas, spesifisitas, *IoU,* dan *F1-Score* dikategorikan dalam beberapa kategori yang dapat dilihat pada Tabel 2.2 (Wahyudi *et al*., 2019).

Tabel 2.2. Kategori nilai evaluasi kinerja model

Nilai Kinerja (%)	Kategori
> 90	Sangat Baik
81 - 90	Baik
71 – 80	Cukup
61 - 70	Kurang baik
< 60	Gagal

Berdasarkan Tabel 2.2 ada lima kategori yang mengklasifikasikan ukuran evaluasi kinerja model, terdiri dari kategori sangat baik, baik, cukup, kurang baik, dan gagal.

BAB III

METODE PENELITIAN

3.1 Tempat

Penelitian ini dilakukan di Laboratorium Komputasi Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Sriwijaya.

3.2 Waktu

Waktu yang dibutuhkan untuk pelaksanaan penelitian ini ialah 7 bulan yaitu dari November 2023 hingga Mei 2024.

3.3 Alat

Alat-alat yang digunakan dalam penelitian ini yaitu sebagai berikut:

1. Komputer dengan sistem operasi *windows* 10, 64 bit, *processor* Intel CoreTM I9-9900X, RAM 32 GB, memori 2TB, serta grafika Dual NVIDIA Geoforce RTX 2080.
2. Bahasa pemrograman *python.*
3. Aplikasi *Spyder* pada *Anaconda*.

3.4 Tahap Penelitian

1. Pengumpulan Data

Dataset yang digunakan pada penelitian ini merupakan data segmentasi hati. Pada *dataset* ini terdapat hasil *CT Scan* dari hati yang diperoleh melalui laman web

Kaggle dan dapat diakses melaui link: https://www.kaggle.com/*dataset* s/javariatahir/litstrain-valdata. Pada data *liver segmentation* ini, berisi data berbentuk tiga dimensi dengan jumlah 131 file citra. Selain citra asli, juga terdapat 131 *ground truth* dari setiap citra pada data *liver segmentation*. Bagian citra yang disegmentasi adalah area hati dengan citra dan *ground truth.*

2. *Preprocessing*

Pada tahap *preprocessing* masing-masing citra dilakukan pemotongan dari citra 3D beserta *ground truth* menjadi citra 2D berdasarkan sumbu aksial, koronal, dan sagital. Tahap ini dimulai dengan proses *input* citra *CT Scan* dari *dataset*. Lalu, masing-masing citra asli dan *ground truth* dipotong dengan menerapkan teknik MPR *slicing* yang dipotong searah sumbu aksial, koronal, dan sagital sehingga dihasilkan suatu potongan berbentuk citra 2D untuk masing-masing citra dan *ground truth*.

3. Implementasi Arsitektur VGG-16, *UNet* dan *Inception*

Arsitektur yang digunakan merupakan kombinasi dari arsitektur VGG-16 dan *UNet,* dengan *Inception*. Arsitektur VGG-16 digunakan pada bagian *encoder* dan *UNet* pada bagian *decoder*. Arsitektur *Inception* disisipkan diantara kedua bagian arsitektur *UNet* yaitu pada bagian *bridge*. Pada tahap implementasi ini dilakukan dua proses, yaitu *training* dan *testing*.

a. *Training*

Proses *training* dilakukan dengan tujuan agar model dapat terlatih untuk mempelajari pola dan mengenali fitur-fitur data. Adapun langkah-langkahnya adalah sebagai berikut:

i. Melakukan *split* data hasil *preprocessing* menjadi 80% data *training* dan 20% data *testing*.

ii. Menginisialisasi jumlah *epoch* dan *batch size* yang digunakan.

iii. Melakukan inisialisasi nilai bobot pada *epoch* pertama.

iv. Memasukkan data hasil *preprocessing* ke dalam model arsitektur VGG-16 *UNet, Inception*.

Melakukan proses konvolusi pada bagian *encoder* arsitektur VGG-16 dengan menggunakan Persamaan (2.1) dan (2.2) lalu dilakukan fungsi aktivasi *ReLU* menggunakan Persamaan (2.3), *Batch Normalization* dengan menghitung rata-rata *mini batch* menggunakan Persamaan (2.4) dan variansi Persamaan (2.5) yang dinormalisai menggunakan Persamaan (2.6), *dropout*, dan *max pooling*. Pada bagian *bridge*, dilakukan konvolusi dengan *Inception*. Pada bagian *decoder* dilakukan operasi *concatenate*, konvolusi, *dropout*, fungsi aktivasi *ReLU*, dan fungsi aktivasi *Sigmoid* dengan Persamaan (2.7).

v. Menghitung *Loss Function: Binary Cross Entropy* menggunakan Persamaan (2.8).

vi. Menghitung nilai *training* untuk setiap *epoch* dengan fungsi optimisasi *Adam* menghitung nilai momen pertama menggunakan Persamaan (2.9), momen kedua menggunakan Persamaan (2.10), penduga dari momen pertama menggunakan Persamaan (2.11), penduga dari momen kedua menggunakan Persamaan (2.12), dan memperbarui nilai bobot menggunakan Persamaan (2.13).

vii. Memberikan bobot pada *epoch* berikutnya, kemudian lakukan langkah yang sama seperti pada *epoch* pertama. Apabila nilai *loss validation* yang dihasilkan kurang dari nilai *loss validation* pada *epoch* sebelumnya maka bobot disimpan untuk digunakan pada *epoch* berikutnya. Jika tidak, maka bobot diperbarui untuk *epoch* selanjutnya.

viii. Mengulang proses yang sama seperti pada langkah (iii) sampai langkah (vii) hingga *epoch* terakhir.

b. *Testing*

Proses *testing* dilakukan untuk menguji model yang telah dilatih pada proses *training* dengan menggunakan data *testing* yang telah ditentukan.

4. Evaluasi

Evaluasi kinerja model dilakukan berdasarkan *confusion matrix* yang dihasilkan pada tahap *testing*. Evaluasi kinerja yang diukur diantaranya yaitu nilai akurasi menggunakan Persamaan (2.14), sensitivitas menggunakan Persamaan (2.15), spesifisitas menggunakan Persamaan (2.16), *IoU* menggunakan Persamaan (2.17), dan *F1-Score* menggunakan Persamaan (2.18).

5. Analisis dan Interpretasi Hasil

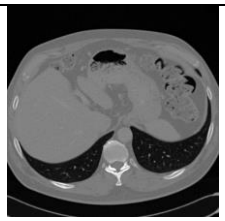
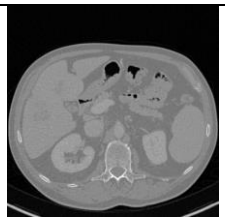
Hasil evaluasi kinerja model dianalisis berdasarkan Tabel 2.2, kemudian ditentukan kategori hasil penelitian.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Deskripsi Data

Data yang diterapkan dalam penelitian ini merupakan data segmentasi hati yang berisi hasil *CT Scan* dari hati yang didapat dari laman web https://www.kaggle.com/datasets/javariatahir/litstrain-val. Data yang digunakan pada data *Liver Segmentation* terdiri dari data dalam bentuk tiga dimensi dengan total 131 file citra. Sampel data citra pada data *Liver Segmentation* ditunnjukkan pada Tabel 4.1 berikut.

Tabel 4.1. Sampel Data Citra *Liver Segmentation*

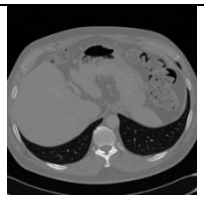
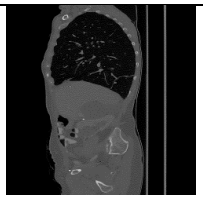
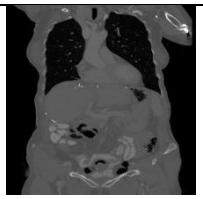
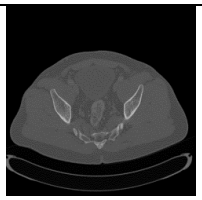
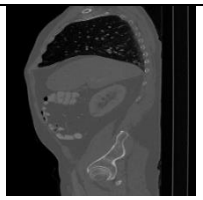
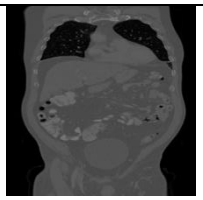
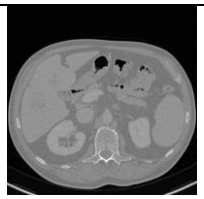
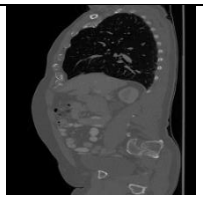
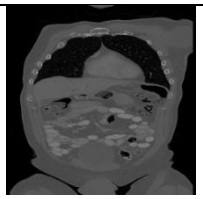
No.	Nama File	Citra Asli
1.	volume-7.nii	
2.	volume-20.nii	
3.	volume-97.nii	

4.2 *Preprocessing* Data

Langkah awal *preprocessing*, citra 3D beserta dengan *ground truth* dipotong menjadi citra 2D dengan menerapkan teknik MPR *slicing* yang dipotong

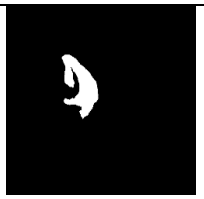
berdasarkan tiga sumbu yaitu sumbu aksial, sagital dan koronal. Masing-masing citra asli beserta *ground truth* dipotong berdasarkan sumbu aksial, sagital, serta koronal sehingga didapatkan hasil pemotongan 2D untuk setiap citra beserta *ground truth*. Contoh hasil pemotongan citra ditampilkan pada Tabel 4.2 dan contoh hasil pemotongan *ground truth* ditampilkan pada Tabel 4.3.

Tabel 4.2. Contoh Potongan Citra

No.	Citra Asli	Sumbu Aksial	Sumbu Sagital	Sumbu Koronal
1.				
2.				
3				

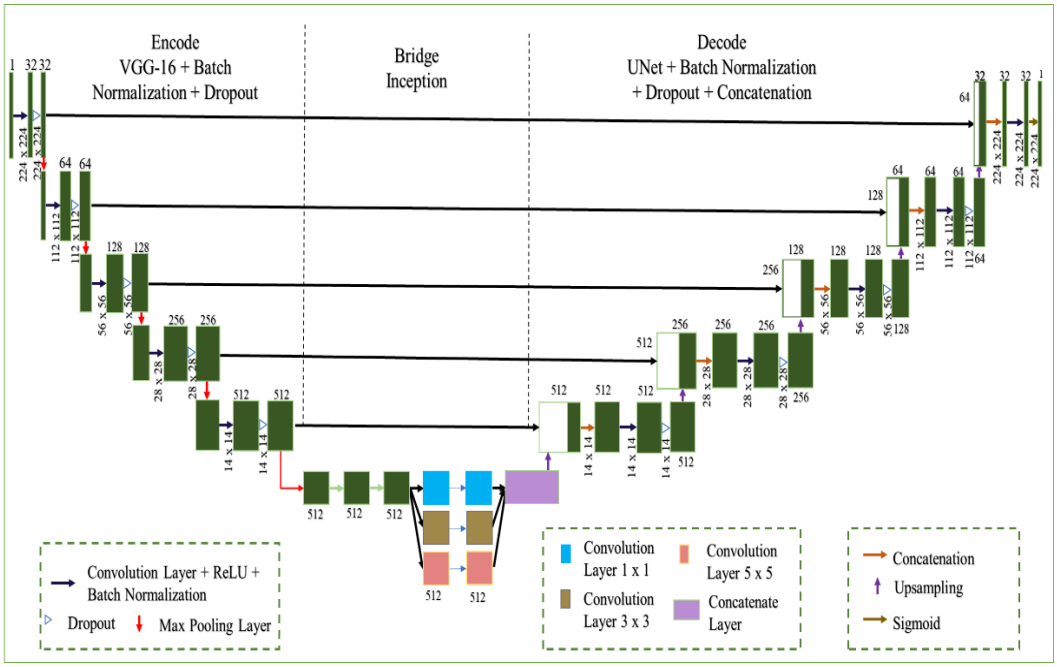
Tabel 4.3. Contoh Potongan *Ground Truth*

No.	Ground Truth (3D)	Sumbu Aksial	Sumbu Sagital	Sumbu Koronal
1.				
2.				

3				

4.3 Kombinasi Arsitektur VGG-16, *UNet,* dan *Inception*

Seletah dilakukan *preprocessing,* maka selanjutnya mengkombinasikan arsitektur VGG-16, *UNet,* dan *Inception* dengan menjalankan tahapan *coding* dengan *Jupyter*. Modifikasi VGG-16, *UNet,* dan *Inception* yang dimodifikasi adalah *UNet* yang jalur *encoder* diisi dengan arsitektur VGG-16 dan *bridge* diisi oleh arsitektur *Inception* dengan tambahan *batch normalization* dan *dropout* seperti yang ditunjukkan pada Gambar 4.1 berikut.



Gambar 4.1. Kombinasi Arsitektur VGG-16, *Inception,* dan *UNet*

Dapat dilihat pada Gambar 4.1 pada kombinasi arsitektur VGG-16, *Inception* dan *UNet* terdiri dari jalur *encoder*, *bridge*, dan *decoder*. Jalur *encoder* memiliki

lima blok yang setiap bloknya berisikan lapisan konvolusi 3 × 3, ReLU, *batch normalization*, dan *dropout*, yang diakhiri penerapan *max pooling*. Pada jalur *bridge* berisikan tiga *layer* konvolusi 3 × 3, lalu digunakan pada blok *Inception* yang berisikan dua blok konvolusi. Setiap blok pada *Inception* berisikan tiga *layer* konvolusi 1 × 1, 3 × 3, dan 5 × 5 serta penerrapan fungsi *dropout* yang diteruskan penerapan *max pooling*. Jalur *decoder* berisikan lima blok yang pada masing-masing blok terdapat konvolusi, *batch normalization*, fungsi aktivasi. Bagian terakhir dari decoder diakhiri dengan lapisan konvolusi berukuran 1 × 1 yang menggunakan fungsi aktivasi *Sigmoid*.

Berikut ini adalah tahapan-tahapan yang dilakukan pada jalur *encoder* :

1. Langkah awal dilakukan input citra berukuran 224 × 224. Lalu pada blok pertama dilakukan proses konvolusi dengan padding same, fungsi aktivasi *ReLU*, dan operasi *batch normalization*. Pada lapisan konvolusi kedua, diterapkan teknik *dropout*. Terakhir, diterapkan operasi *max pooling* 2 × 2 dengan stride 2.
2. Langkah yang sama diterapkan diblok kedua, ketiga, keempat, dan kelima yang akan dilanjutkan sebagai *input* pada *bridge*.

Bagian berikutnya yaitu *bridge*, dengan tahapan yang diterapkan pada *bridge* adalah sebagai berikut:

1. *Output* pada langkah sebelumnya diteruskan ke blok *Inception*, dimana *output* dari *layer* pada blok terakhir *encoder* dilanjutkan ke operasi konvolusi

matriks kernel 1×1, 3×3, dan 5×5 secara bersamaan, kemudian dilakukan operasi *max pooling* 3×3.

2. Mengkombinasikan hasil atau *feature maps* sebelum jalur *bridge* dengan *feature maps* dari langkah kedua dengan *concatenate.*

Bagian terakhir adalah jalur *decoder* dengan tahapan berikut:

1. Pada blok pertama, diterapkan *upsampling*, dilanjutkan melakukan proses konvolusi dengan *padding same* dan fungsi aktivasi *ReLU*. Selanjutnya, dilakukan operasi *batch normalization*. *Layer* selanjutnya dilakukan operasi konvolusi dengan *dropout* yang kemudian ditransformasikan.
2. Langkah yang sama diterapkan diblok kedua, ketiga, keempat, dan kelima. Terakhir, penerapan fungsi aktivasi *Sigmoid* pada lapisan.

4.4 Operasi Manual *Convolutional Neural Network* (CNN)

Perhitungan secara manual pada *Convolutional Neural Network* (CNN) dengan parameter yang digunakan dalam penerapan kombinasi arsitektur *VGG-16, UNet,* dan *Inception* adalah sebagai berikut:

1. Convolutional Layer

a. Padding same

Penerapan *padding same* dimanfaatkan agar dapat menyelaraskan ukuran matriks dalam konvolusi dengan menggunakan entri 0 di sekeliling entri matriks *input*. Contoh dari proses *padding same* adalah sebagi berikut:

Contoh *Padding Same*

Sebagai contoh, dimisalkan matriks *input P* dari citra asli dengan entri acak berukuran 3 × 3 seperti berikut.

$$P = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 0 \\ 3 & 2 & 1 \end{bmatrix}$$

Kemudian, proses *padding same* dilanjutkan dengan penambahan entri 0 di sekeliling entri matriks *P* hingga ukuran matriks *P* menjadi 5 × 5.

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

b. Operasi konvolusi

Dalam proses operasi konvolusi digunakan *stride*, jumlah filter, *padding*, dan matriks kernel yang dihasilkan komputer secara *random*. Berikut contoh bagaimana proses operasi konvolusi dilakukan:

Contoh Operasi Konvolusi

Contohnya, matriks *P* dari citra asli sebagai matriks *input* ke-1 (P_1) yang telah ditambahkan *padding same* dengan *stride* 1.

$$P_1 = P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Selanjutnya dimisalkan matriks kernel ke-1(K_1) berukuran 3 x 3 berikut.

$$K_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 3 & 1 \\ 2 & 2 & 1 \end{bmatrix}$$

Selanjutnya, perhitungan operasi konvolusi diproses dengan menggunakan Persamaan (2.1).

$$D = (P_r \cdot K_s) + b_s$$

$$D = (P_1 \cdot K_1) + b_1$$

$$D = \left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 3 & 1 \\ 2 & 2 & 1 \end{bmatrix} \right) + b_1$$

Pada *convolutional layer*, operasi konvolusi dihitung dan bias ditambahkan pada setiap entri dalam matriks hasil operasi konvolusi. Misalnya, nilai bias pada kernel ke-1 (b_1) = 0,5. Untuk mendapatkan hasil pada baris pertama dan kolom pertama dari *convolution layer*, dilakukan perhitungan berikut:

$$d_{ij} = \left(\sum_{u=0}^{m-1} \sum_{v=0}^{n-1} (p_{u+i\,v+j} \cdot k_{u+i\,v+j})\right) + b_s$$

$$d_{11} = \left(\sum_{u=0}^{3-1} \sum_{v=0}^{3-1} (p_{u+1\,v+1} \cdot k_{u+1\,v+1})\right) + b_s$$

$$d_{11} = \left(\sum_{u=0}^{2} \sum_{v=0}^{2} (p_{u+1\,v+1} \cdot k_{u+1\,v+1})\right) + b_s$$

$$d_{11} = \left((p_{0+1\,0+1} \cdot k_{0+1\,0+1}) + \cdots + (p_{2+1\,2+1} \cdot k_{2+1\,2+1})\right) + b_s$$

$$d_{11} = \left((p_{11} \cdot k_{11}) + \cdots + (p_{33} \cdot k_{33})\right) + 0{,}5$$

$$d_{11} = \left((0 \cdot 1) + \cdots + (0 \cdot 1)\right) + 0{,}5$$

$$d_{11} = 3 + 0{,}5$$

$$d_{11} = 3{,}5$$

Langkah yang sama diterapkan pada entri-entri baris dan kolom selanjutnya hingga entri baris ke-3 kolom ke-3, dengan asumsi bahwa semua bias yang digunakan pada kernel bernilai 0,5. Hasil dari perhitungan convolution layer untuk input matriks pertama dan kernel pertama dapat dilihat pada matriks D berikut.

$$D = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} = \begin{bmatrix} 3{,}5 & 7{,}5 & 7{,}5 \\ -0{,}5 & 0{,}5 & 7{,}5 \\ -6{,}5 & 5{,}5 & 5{,}5 \end{bmatrix}$$

2. Fungsi Aktivasi *ReLU*

Pada proses ini, masing-masing nilai entri matriks *input* disubstitusikan pada fungsi aktivasi *ReLU* dengan Persamaan (2.3). Contoh operasi *ReLU* dapat dilihat sebagai berikut:

Contoh *ReLU* :

Misalkan matriks D hasil perhitungan Konvolusi, maka entri matriks d_{11} = 3,5 . Kemudian entri matriks d_{11}disubtusikan kedalam fungsi aktivasi *ReLU* sehingga diperoleh

$$g(3{,}5) = max\ (0; 3{,}5) = 3{,}5$$

Diterapkan langkah yang sama pada masing-masing nilai entri matriks D untuk baris dan kolom seterusnya ampai baris ke-3 kolom ke-3, hingga idapatkan matriks G berikut.

$$G = \begin{bmatrix} 3{,}5 & 7{,}5 & 7{,}5 \\ 0 & 0{,}5 & 7{,}5 \\ 0 & 5{,}5 & 5{,}5 \end{bmatrix}$$

3. Batch *Normalization*

Pada proses ini, matriks input dilakukan normalisasi dengan menghitung nilai rata-rata (μ_j) menggunakan Persamaan (2.4) dan varians(σ_j^2) pada setiap mini batch menggunakan Persamaan (2.5) yang kemudian dinormalisasi dengan batch normalization menggunakan Persamaan (2.6). Contoh operasi *batch normalization* diberikan sebagai berikut:

Contoh Proses *Batch Normalization*

Misal, matriks hasil perhitungan *convolutional layer* (D) digunakan untuk matriks berukuran 3 × 3.

$$D = \begin{bmatrix} 3{,}5 & 7{,}5 & 7{,}5 \\ -0{,}5 & 0{,}5 & 7{,}5 \\ -6{,}5 & 5{,}5 & 5{,}5 \end{bmatrix}$$

Kemudian, proses normalisasi matriks *X* dengan teknik *batch normalization* dijalankan dengan tahapan berikut:

a. Menentukan ukuran *mini batch* (n) dan jumlah data pada satu *mini batch* (m) matriks *input* serta menentukan nilai ε. Matriks *input* yang diberikan (*X*) berukuran 3 x 3, sehingga didapatkan n = 3 dan m = 3 untuk i = 1, 2, 3 dan j = 1, 2, 3. Dimisalkan nilai $\varepsilon = 10^{-3}$.

b. Melakukan perhitungan nilai rata-rata (μ_j) pada masing-masing *mini batch* matriks *input* dengan Persamaan (2.4).

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} d_{ij}$$

$$\mu_1 = \frac{1}{3}\sum_{i=1}^{3} d_{i1} = \frac{1}{3}(3{,}5 + (-0{,}5) + (-6{,}5)) = \frac{1}{3}(-3{,}5) = -1{,}17$$

$$\mu_2 = \frac{1}{3}\sum_{i=1}^{3} d_{i2} = \frac{1}{3}(7{,}5 + 0{,}5 + 5{,}5) = \frac{1}{3}(13{,}5) = 4{,}5$$

$$\mu_3 = \frac{1}{3}\sum_{i=1}^{3} d_{i3} = \frac{1}{3}(7{,}5 + 7{,}5 + 5{,}5) = \frac{1}{3}(20{,}5) = 6{,}83$$

c. Melakukan perhitungan variansi (σ_j^2) pada masing-masing *mini batch* matriks *input* menggunakan Persamaan (2.5).

$$\sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(d_{ij} - \mu_j)^2$$

$$\sigma_1^2 = \frac{1}{3}\sum_{i=1}^{3}(d_{i1} - \mu_1)^2$$

$$\sigma_1^2 = \frac{1}{3}((3{,}5 - (-1{,}17))^2 + ((-0{,}5) - (-1{,}17))^2 + ((-6{,}5) - (-1{,}17))^2)$$

$$\sigma_1^2 = \frac{1}{3}(21{,}81 + 0{,}45 + 28{,}41) = \frac{1}{3}(50{,}67) = 16{,}89$$

$$\sigma_2^2 = \frac{1}{3}\sum_{i=1}^{3}(d_{i2} - \mu_2)^2$$

$$\sigma_2^2 = \frac{1}{3}((7{,}5 - 4{,}5)^2 + (0{,}5 - 4{,}5)^2 + (5{,}5 - 4{,}5)^2)$$

$$\sigma_2^2 = \frac{1}{3}(9 + 16 + 1) = \frac{1}{3}(26) = 8{,}67$$

$$\sigma_3^2 = \frac{1}{3}\sum_{i=1}^{3}(d_{i3} - \mu_3)^2$$

$$\sigma_3^2 = \frac{1}{3}((7{,}5 - 6{,}83)^2 + (7{,}5 - 6{,}83\ )^2 + (5{,}5 - 6{,}83)^2)$$

$$\sigma_3^2 = \frac{1}{3}(0{,}45 + 0{,}45 + 1{,}77) = \frac{1}{3}(2{,}67) = 0{,}89$$

d. Menormalisasi setiap entri matriks *input* menggunakan Persamaan (2.6).

$$x_{ij} = \frac{d_{ij} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

$$x_{11} = \frac{d_{11} - \mu_1}{\sqrt{\sigma_1^2 + \varepsilon}} = \frac{3{,}5 - (-1{,}17)}{\sqrt{16{,}89 + 10^{-3}}} = \frac{4{,}67}{\sqrt{16{,}891}} = 1{,}14$$

$$x_{12} = \frac{d_{12} - \mu_2}{\sqrt{\sigma_2^2 + \varepsilon}} = \frac{7{,}5 - 4{,}5}{\sqrt{8{,}67 + 10^{-3}}} = \frac{3}{\sqrt{8{,}671}} = 1{,}02$$

$$x_{13} = \frac{d_{13}-\mu_3}{\sqrt{\sigma_3^2+\varepsilon}} = \frac{7{,}5-6{,}83}{\sqrt{0{,}89+10^{-3}}} = \frac{0{,}67}{\sqrt{0{,}891}} = 0{,}71$$

$$x_{21} = \frac{d_{21}-\mu_1}{\sqrt{\sigma_1^2+\varepsilon}} = \frac{-0{,}5-(-1{,}17)}{\sqrt{16{,}89+10^{-3}}} = \frac{0{,}67}{\sqrt{16{,}891}} = 0{,}16$$

$$x_{22} = \frac{d_{22}-\mu_2}{\sqrt{\sigma_2^2+\varepsilon}} = \frac{0{,}5-4{,}5}{\sqrt{8{,}67+10^{-3}}} = \frac{-4}{\sqrt{8{,}671}} = -1{,}36$$

$$x_{23} = \frac{d_{23}-\mu_3}{\sqrt{\sigma_3^2+\varepsilon}} \frac{7{,}5-6{,}83}{\sqrt{0{,}89+10^{-3}}} = \frac{0{,}67}{\sqrt{0{,}891}} = 0{,}71$$

$$x_{31} = \frac{d_{31}-\mu_1}{\sqrt{\sigma_1^2+\varepsilon}} = \frac{-6{,}5-(-1{,}17)}{\sqrt{16{,}89+10^{-3}}} = \frac{-5{,}33}{\sqrt{16{,}891}} = -1{,}3$$

$$x_{32} = \frac{d_{12}-\mu_2}{\sqrt{\sigma_2^2+\varepsilon}} = \frac{5{,}5-4{,}5}{\sqrt{8{,}67+10^{-3}}} = \frac{1}{8{,}671} = 0{,}34$$

$$x_{33} = \frac{d_{33}-\mu_3}{\sqrt{\sigma_3^2+\varepsilon}} = \frac{5{,}5-6{,}83}{\sqrt{0{,}89+10^{-3}}} = \frac{-1{,}33}{\sqrt{0{,}891}} = -1{,}41$$

sehingga didapat matriks hasil proses *batch normalization* berikut.

$$X = \begin{bmatrix} 1{,}14 & 1{,}02 & 0{,}71 \\ 0{,}16 & -1{,}36 & 0{,}71 \\ -1{,}3 & 0{,}34 & -1{,}41 \end{bmatrix}$$

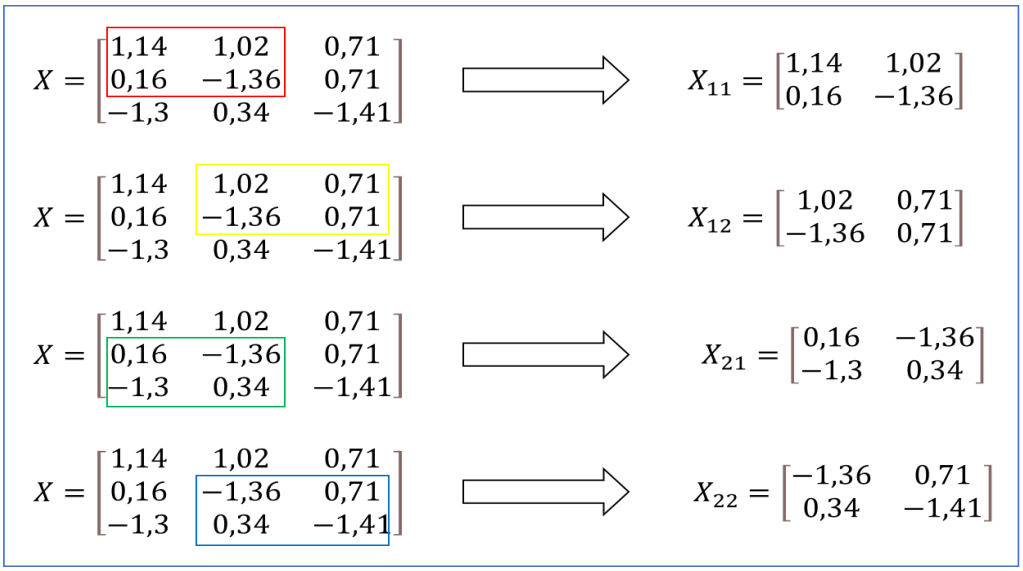
4. Max Pooling

Max pooling bekerja membagi matriks *input* menjadi sub-sub matriks dengan ukuran $n \times n$ berdasarkan dimensi filter *max pooling* yang dipilih, serta melakukan pergesaran *stride* baik secara horizontal maupun vertikal.

Contoh *Max Pooling* :

Misalkan diambil matriks *X* dari perhitungan *Batch Normalization* sebagai matriks *input*. Selanjutnya, menentukan dimensi filter *max pooling* dan *stride* yang diterapkan. Contoh mengambil filter *max pooling* engan ukuran 2 x 2 dan *stride* 1,

lalu matriks *input* X dipecah menjadi sub-sub matriks dengan ukuran 2×2 dan pergesaran *stride* secara horizontal maupun vertikal sebesar 1. Contoh proses pembagian sub-sub matriks dilihat pada Gambar 4.2 berikut ini.



Gambar 4.2 *Max Pooling*

Gambar 4.2 menunjukkan kotak berwarna merah, kuning, hijau dan biru sebagai matriks filter *max pooling* berukuran 2×2 yang dipartisi agar dihasilkan submatriks X_{11}, X_{12}, X_{21}, dan X_{22}. Kemudian diambil nilai entri terbesar dari setiap submatriks.

$$m_{11} = \max(X_{11}) = 1{,}14$$

$$m_{12} = \max(X_{12}) = 1{,}02$$

$$m_{21} = \max(X_{21}) = 0{,}34$$

$$m_{22} = \max(X_{22}) = 0{,}71$$

Dari nilai tertinggi pada entri submatriks yang diperoleh, dapat dihasilkan matriks M dengan ukuran 2 x 2 hasil operasi *max pooling* berikut:

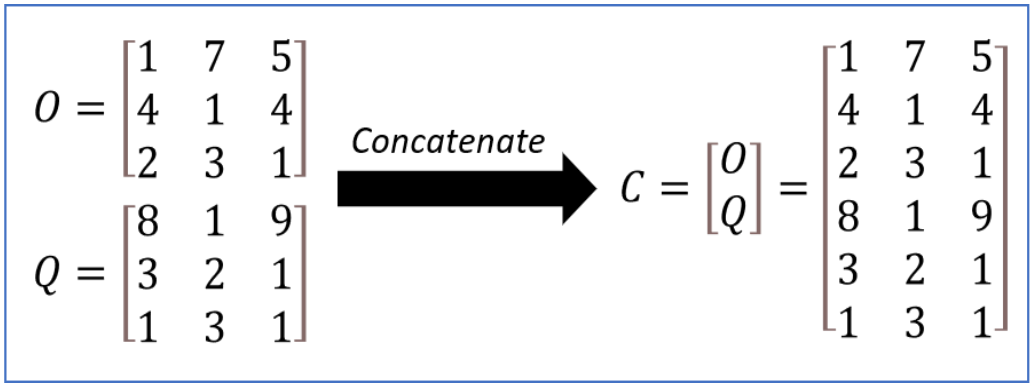
$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = \begin{bmatrix} 1{,}14 & 1{,}02 \\ 0{,}34 & 0{,}71 \end{bmatrix}$$

5. *Concatenate*

Concatenate bermanfaat untuk menyatukan dua matriks *input* menjadi suatu matriks *input* baru dengan ukuran matriks berbeda. Cara menyatukan dengan penerapan *concatenate* dapat dilihat dalam contoh berikut.

Contoh Proses *Concatenate* :

Dimisalkan matriks O dan Q dengan ukuran 3 × 3. Berikut proses *Concatenate* yang dilakukan pada kedua matriks tesebut:



Gambar 4.3 *Concatenate*

Gambar 4.3 menunjukkan matriks O dengan ukuran 3 × 3 dan matriks Q dengan ukuran 3 × 3 kemudian disatukan dengan penggabungan *concatenate* dan dihasilkan matriks C berukuran 6 × 3.

6. *Upsampling*

Penerapan *upsampling* igunakan untuk mningkatkan kembali ukuran matriks *input*. Contoh penerapan *Upsampling* dapat dilihat sebagai berikut:

Contoh Proses *Upsampling*:

Dimisalkan sembarang matriks A dengan ukuran 3 × 3. Berikut proses *upsampling* yang dilakukan pada matriks tesebut:

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 3 & 2 & 0 \\ 3 & 3 & 4 \end{bmatrix}$$

Masing-masing nilai entri pada matriks A digunakan kembali pada submatriks baru yang dengan ukuran 2 × 2. Misal $a_{11} = 2$ digunakan pada submatriks lainnya (z_{11}) yang dengan ukuran 2 × 2.

$$z_{11} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

Lalu diterapkan langkah yang sama pada nilai entri matriks Z lainnya, hingga membentuk submatriks z_{12}, z_{13}, z_{21}, z_{22}, z_{23}, z_{31}, z_{32}, dan z_{33}. Selanjutnya, submatriks-submatriks yang dihasilkan disatukan sebagai matriks Z dengan ukuran 6 × 6 menjadi hasil proses *upsampling*.

$$Z = \begin{bmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ z_{31} & z_{32} & z_{33} \end{bmatrix} = \begin{bmatrix} 2 & 2 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 1 & 1 \\ 3 & 3 & 2 & 2 & 0 & 0 \\ 3 & 3 & 2 & 2 & 0 & 0 \\ 3 & 3 & 3 & 3 & 4 & 4 \\ 3 & 3 & 3 & 3 & 4 & 4 \end{bmatrix}$$

7. Fungsi Aktivasi *Sigmoid*

Dalam tahap ini, tiap nilai dalam matriks *input* akan diganti dengan fungsi aktivasi *Sigmoid* menggunakan Persamaan (2.7), sehingga nilai-nilai dalam matriks *input* akan terletak di antara 0 dan 1. Contoh implementasi proses *Sigmoid* dapat diilustrasikan sebagai berikut:

Contoh Proses *Sigmoid*

Untuk contoh dimisalkan matriks D dengan ukuran 3×3 menjadi matriks *input*:

$$D = \begin{bmatrix} 3{,}5 & 7{,}5 & 7{,}5 \\ -0{,}5 & 0{,}5 & 7{,}5 \\ -6{,}5 & 5{,}5 & 5{,}5 \end{bmatrix}$$

Berdasarkan matriks R didapatkan nilai untuk entri d_{11} = 3,5. Selanjutnya, digunakan nilai yang dihasilkan pada fungsi aktivasi *Sigmoid* engan Persamaan (2.7) hingga dihasilkan sebagai berikut :

$$s_{ij} = \frac{1}{1+e^{-d_{ij}}}$$

$$s_{11} = \frac{1}{1+e^{-d_{11}}} = \frac{1}{1+e^{-3{,}5}} = 0{,}971$$

Lakukan langkah sebelumnya pada entri matriks D untuk baris dan kolom selanjutnya, sehingga menghasilkan sebuah matriks S sebagai berikut :

$$S = \begin{bmatrix} 0{,}971 & 0{,}999 & 0{,}999 \\ 0{,}378 & 0{,}622 & 0{,}999 \\ 0{,}001 & 0{,}996 & 0{,}996 \end{bmatrix}$$

8. *Loss Function: Binary Cross-entropy*

Binary cross entropy dimanfaatkan untuk menilai kesalahan antara prediksi dan kebenaran mutlak dengan memakai Persamaan (2.8). Contoh dari proses *Loss Function: Binary cross entropy* bisa diilustrasikan seperti ini:

Contoh Proses *Loss Function: Binary cross entropy*

Contohnya, jika diambil matriks S dari langkah-langkah Sigmoid sebagai hasil prediksi, dan kita awali matriks B dengan ukuran 3×3 sebagai matriks *ground truth* yang berisi angka 0 atau 1.

$$S = \begin{bmatrix} 0{,}971 & 0{,}999 & 0{,}999 \\ 0{,}378 & 0{,}622 & 0{,}999 \\ 0{,}001 & 0{,}996 & 0{,}996 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Kemudian, dilakukan proses perhitungan *categorical cross-entropy* dengan menggunakan Persamaan (2.8).

$$\mathrm{L} = -\frac{1}{m \times n}\left[\sum_{i=1}^{m}\sum_{j=1}^{n}\left(b_{ij}\log(s_{ij})\right) + (1 - b_{ij})(1 - \log(s_{ij}))\right]$$

$$\mathrm{L} = -\frac{1}{3 \times 3}\left[\sum_{i=1}^{3}\sum_{j=1}^{3}\left(b_{ij}\log(s_{ij})\right) + (1 - b_{ij})(1 - \log(s_{ij}))\right]$$

$$\mathrm{L} = -\frac{1}{9}\left[(b_{11}\log(s_{11})) + (1 - b_{11})(1 - \log(s_{11})) + \ldots + (b_{33}\log(s_{33})) + (1 - b_{33})(1 - \log(s_{33}))\right]$$

$$\mathrm{L} = -\frac{1}{9}\left[(1\log(0{,}971)) + (1 - 1)(1 - \log(0{,}971)) + \cdots + (1\log(0{,}996)) + (1 - 1)(1 - \log(0{,}996))\right]$$

$$\mathrm{L} = -\frac{1}{9}\left[(-0{,}0128) + \cdots + (-0{,}0017)\right]$$

$$\mathrm{L} = -\frac{1}{9}(-0{,}6411) = 0{,}0712$$

Jadi, nilai kesalahan dalam memprediksi hasil segmentasi yang dibandingkan dengan *ground truth* senilai 0,0712.

9. *Optimization Function: Adaptive Moment Estimation (Adam)*

Fungsi optimisasi *Adam* berguna untuk meningkatkan efisiensi bobot yang diterapkan selama pelatihan. Sebagai contoh, metode Adam dapat diilustrasikan sebagai berikut:

Contoh Proses *Adam*

Misal diberikan matriks S, matriks B sebagai matriks *ground truth*, serta *loss function* sebesar 0,0712.

$$S = \begin{bmatrix} 0{,}971 \\ 0{,}378 \\ 0{,}001 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Langkah-langkah perhitungan *Adam optimizer* dilakukan dengan Persamaan (2.9), (2.10), (2.11), (2.12) dan (2.13) dengan tahapan-tahapan berikut:

a. Menginisialisasi nilai pertama untuk momen pertama ($m_0 = 0$), nilai pertama momen kedua ($v_0 = 0$), bobot awal ($w_0 = 0$), *decay rate* untuk momen pertama ($\beta_1 = 0{,}9$), *decay rate* momen kedua ($\beta_2 = 0{,}99$), jumlah *epoch* yang dilakukan sebanyak 50 dan *learning rate* ($\eta = 0{,}01$),

b. Menghitung gradien pertama berdasarkan *loss function* menggunakan *categorical cross entropy*.

$g_s = \sum_{i=1}^{n}(s_{i1} - y_{i1})$

$g_1 = (0{,}971 - 1) + (0{,}378 - 1) + (0{,}001 - 1) = -1{,}65$

c. Melakukan pembaharuan momen pertama *epoch* ke-1

$m_1 = \beta_1 m_{1-1} + (1 - \beta_1) g_1 = 0{,}9(0) + (1 - 0{,}9)(-1{,}65) = -0{,}165$

d. Melakukan pembaharuan nilai momen kedua pada *epoch* ke-1

$v_1 = \beta_2 v_{1-1} + (1 - \beta_2) g_1^2 = 0{,}99(0) + (1 - 0{,}99)(-0{,}165)^2 = 0{,}0003$

e. Melakukan perhitungan nilai penduga momen pertama serta momen kedua

$\widehat{m_1} = \frac{m_1}{1 - \beta_1^1} = \frac{-0{,}165}{1 - 0{,}9} = -1{,}65$

$$\widehat{v_1} = \frac{v_1}{1-\beta_2^1} = \frac{0{,}0003}{1-0{,}99} = 0{,}03$$

f. Lakukan pembaharuan bobot untuk *epoch* ke-1

$$w_1 = w_{1-1} - \eta\frac{m_1}{\sqrt{v_1+\varepsilon}} = 0 - 0{,}01\,\frac{-0{,}165}{\sqrt{0{,}03+10^{-3}}} = 0{,}3012$$

g. substitusikan nilai m_1 ke Persamaan (2,9*),* v_1 ke Persamaan (2.10), dan w_1 ke Persamaan (2.13) untuk menghasilkan bobot baru pada *epoch* kedua.

h. Lakukan langkah yang sama dari Langkah (a) hingga (g) sampai *epoch* ke-50. Bobot terkecil yang dihasilkan dari 50 *epoch* yang dilakukan disimpan.

4.5 *Training*

Selama proses *training*, jumlah *epoch* yang digunakan berjumlah 50 dan *batch size* 32. Jumlah citra 2D saat proses *preprocessing* adalah 192.784. Data yang dihasilkan dibagi menjadi 80% dari data *training* dengan jumlah 161.273 citra, sementara 20% sisanya untuk *testing* dengan jumlah 31.511 citra. *Output* dari proses *training* dapat dilihat dalam Gambar 4.4 di bawah ini.

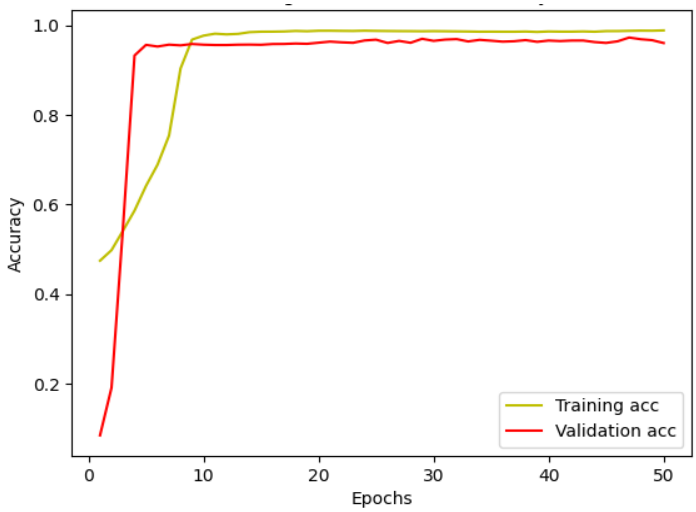
```
Epoch 1/50
100/100 [==============================] - 313s 3s/step - loss: 0.8378 –
accuracy: 0.4743 - val_loss: 0.7332 - val_accuracy: 0.0841
Epoch 2/50
100/100 [==============================] - 274s 3s/step - loss: 0.7292 –
accuracy: 0.4978 - val_loss: 0.7148 - val_accuracy: 0.1906
...
Epoch 49/50
100/100 [==============================] - 197s 2s/step - loss: 0.2703 –
accuracy: 0.9881 - val_loss: 0.2557 - val_accuracy: 0.9669
Epoch 50/50
100/100 [==============================] - 187s 2s/step - loss: 0.2698 –
accuracy: 0.9887 - val_loss: 0.2193 - val_accuracy: 0.9605
```

Gambar 4.4. Proses *Training*

Gambar 4.4 menunjukkan hasil *training* menggunakan kombinasi arsitektur VGG-16, *UNet,* dan *Inception* pada 2 *epoch* awal dan 2 *epoch* terakhir dari 50 *epoch*

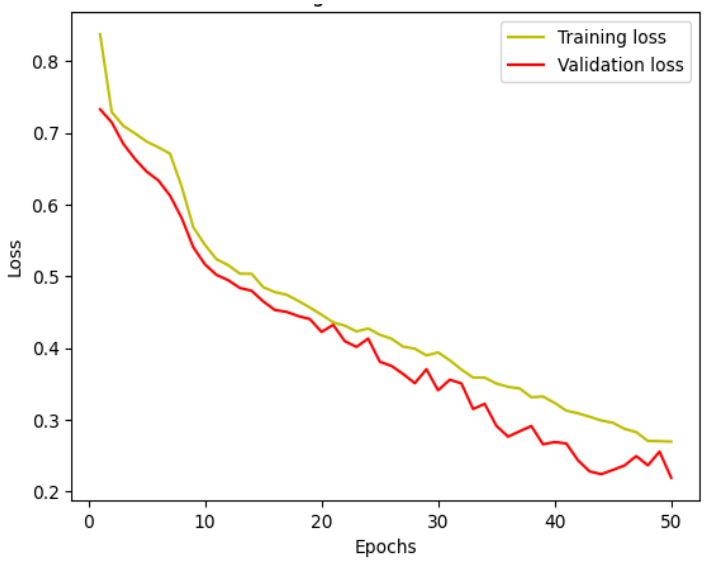
training. Pada *epoch* pertama, akurasi *training* mencapai 47,43% dengan *loss* 0,8378, sementara pada validasi, akurasinya hanya 8,41% dengan *loss* 0,7338. Bobot yang dihasilkan kemudian disimpan dan digunakan kembali untuk *epoch* kedua. Pada *epoch* kedua, akurasi *training* sedikit meningkat menjadi 49,78% dengan *loss* 0,7292, namun akurasi validasi juga mengalami peningkatan menjadi 19,06% dengan *loss* 0,7148.

Bobot yang diperoleh kembali disimpan dikarenakan nilai validasi *loss* yang diperoleh lebih rendah daripada validasi *loss* sebelumnya. Pada *epoch* ke-49, data *training* memiliki akurasi 98,81% dan *loss* 0,2703, sementara data validasi memiliki akurasi 96,69% dan *loss* 0,2557. Bobot baru disimpan kembali karena validasi *loss* lebih rendah daripada validasi *loss* sebelumnya. Pada *epoch* ke-50, akurasi pelatihan meningkat menjadi 98,87% dengan *loss* 0,2698, sementara akurasi validasi adalah 96,05% dengan *loss* 0,2193. Grafik yang menampilkan akurasi data pelatihan dan validasi untuk kombinasi arsitektur VGG-16, *UNet*, dan *Inception* dapat dilihat pada Gambar 4.5.



Gambar 4.5. Grafik Akurasi Proses *Training* pada Data *Training* dan Data Validasi

Pada Gambar 4.5, terlihat bahwa grafik akurasi selama proses *training* mengalami peningkatan yang signifikan hingga mencapai *epoch* ke-9, setelah itu tetap stabil sampai *epoch* terakhir. Awalnya, akurasi pada *epoch* pertama mencapai 47,43%, kemudian naik secara dramatis hingga mencapai puncaknya pada *epoch* ke-9 dengan akurasi mencapai 96,82%, dan selanjutnya tetap stabil sampai *epoch* terakhir. Hal serupa juga terjadi pada data validasi, dimana akurasi meningkat secara signifikan hingga mencapai *epoch* ke-5, dan kemudian tetap stabil sampai akhir *training*. Awalnya, nilai akurasi pada *epoch* pertama adalah 8,41%, dan meningkat secara tajam hingga mencapai 95,65% pada *epoch* ke-5, setelah itu tetap stabil hingga *epoch* terakhir. Grafik yang menunjukkan *loss* dari data *training* dan data validasi dengan kombinasi arsitektur VGG-16, *UNet*, dan *Inception* dapat dilihat pada Gambar 4.6.



Gambar 4.6. Grafik *Loss* Proses *Training* pada Data *Training* dan Data Validasi

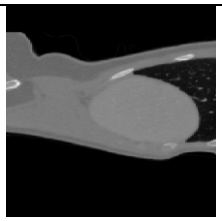
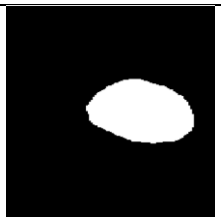
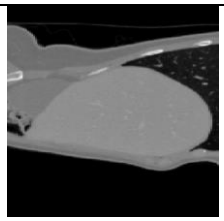
Dapat dilihat pada Gambar 4.6 bahwa grafik nilai *loss* pada proses *training* dan validasi mengalami penurunan dan kenaikan hingga *epoch* terakhir. Nilai *loss*

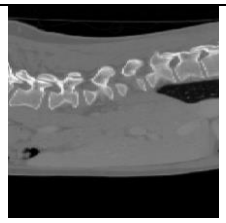
pada data *training* yang dihasilkan sebesar 0,8378 pada *epoch* pertama dan 0,2698 pada *epoch* terakhir. Nilai *loss* pada data validasi yang dihasilkan sebesar 0,7332 pada *epoch* pertama dan 0,2193 pada *epoch* ke-50. Berdasarkan hasil evaluasi akurasi dan *loss* selama proses *training*, dapat disimpulkan bahwa kinerja model sudah sangat baik karena tingkat akurasi melebihi 95% dan nilai *loss* yang mendekati 0.

4.6 *Testing*

Data *testing* yang digunakan berjumlah 31.511 data. Pada langkah ini, didapatkan prediksi segmentasi hati untuk mengevaluasi keakuratan model. Beberapa perbandingan antara citra asli, ground truth, dan hasil segmentasi dapat dilihat dalam Tabel 4.4.

Tabel 4.4. Perbandingan Citra Asli, *Ground Truth*, dan Hasil Segmentasi

No.	Citra Asli	*Ground Truth*	Hasil Segmentasi
1.			
2.			

3.			

Pada Tabel 4.4, menunjukkan bahwa hasil segmentasi menggunakan gabungan arsitektur VGG-16, *UNet,* dan *Inception* sudah menyerupai *ground truth*. *Confusion matrix* diperoleh untuk 2 kelas, yaitu *background* dan hati. *Confusion matrix* yang ihasilkan itunjukkan pada Tabel 4.5.

Tabel 4.5. *Confusion Matrix* dari Proses *Testing*

Kelas		Nilai Prediksi	
		Background	Hati
Nilai Aktual	*Background*	21.083.538	19.093
	Hati	1.043.763	131.750

Dari Tabel 4.5 dapat dilihat bahwa *confusion matrix* yang dihasilkan sejumlah 21.083.538 piksel yang diprediksi benar sebagai *background* dan terdapat 131.750 piksel yang diprediksi benar sebagai hati. Selain itu, ada 19.093 piksel *background* yang terprediksi sebagai hati serta 1.043.763 piksel hati yang terprediksi sebgai *background*.

4.7 Evaluasi

Setelah proses *testing*, evaluasi dilakukan untuk menilai kinerja model dengan menggunakan nilai-nilai yang terdapat dalam *confusion matrix*. Penilaian

kinerja model pada segmentasi citra biner diukur melalui berbagai ukuran evaluasi. diantaranya nilai akurasi menggunakan Persamaan (2.14), sensitivitas dengan Persamaan (2.15), spesifisitas menggunakan Persamaan (2.16), *IoU* dengan Persamaan (2.17), dan *F1-Score* dengan Persamaan (2.18).

1. Akurasi

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

$$\text{Akurasi} = \frac{21.083.538 + 131.750}{21.083.538 + 131.750 + 1.043.763 + 19.093} \times 100\%$$

$$\text{Akurasi} = \frac{21.215.288}{22.278.144} \times 100\%$$

$$\text{Akurasi} = 95{,}23\%$$

2. Sensitivitas

$$\text{Sensitivitas} = \frac{TP}{TP + FP} \times 100\%$$

$$\text{Sensitivitas} = \frac{21.083.538}{21.083.538 + 1.043.763} \times 100\%$$

$$\text{Sensitivitas} = \frac{21.083.538}{22.127.301} \times 100\%$$

$$\text{Sensitivitas} = 95{,}28\%$$

3. Spesifisitas

$$\text{Spesifisitas} = \frac{TP}{TP + FN} \times 100\%$$

$$\text{Spesifisitas} = \frac{21.083.538}{21.083.538 + 19.093} \times 100\%$$

$$\text{Spesifisitas} = \frac{21.083.538}{21.102.631} \times 100\%$$

$$\text{Spesifisitas} = 99{,}91\%$$

4. *IoU*

$$IoU = \frac{TP}{TP + FP + FN} \times 100\%$$

$$IoU = \frac{21.083.538}{21.083.538 + 1.043.763 + 19.093} \times 100\%$$

$$IoU = \frac{21.083.538}{22.146.394} \times 100\%$$

$$IoU = 95{,}20\%$$

5. *F1-Score*

$$F1\text{-}Score = \frac{2TP}{2TP + FP + FN} \times 100\%$$

$$F1\text{-}Score = \frac{2(21.083.538)}{2(21.083.538) + 1.043.763 + 19.093} \times 100\%$$

$$F1\text{-}Score = \frac{42.083.798}{43.229.932} \times 100\%$$

$$F1\text{-}Score = 97{,}54\%$$

Dari perhitungan yang dilakukan, diperoleh nilai akurasi sebesar 95,23%, sensitivitas sebesar 95,28%, spesifisitas sebesar 99,91%, *IoU* sebesar 95,20% dan *F1-Score* sebesar 97,54%.

4.8 Analisis dan Interpretasi Hasil

Setelah evaluasi, langkah selanjutnya dilakukan analisis dan interpretasi hasil. Pada langkah ini diperbandingan hasil evaluasi segmentasi hati menggunakan kombinasi arsitektur VGG-16, *UNet,* dan *Inception* dengan hasil evaluasi segmentasi hati pada penelitian sebelumnya. Perbandingan hasil evaluasi dicantumkan pada Tabel 4.6.

Tabel 4.6. Perbandingan Hasil Evaluasi dengan Penelitian Lainnya

Metode	Acc (%)	Sen (%)	Spe (%)	*IoU* (%)	*F1-Score* (%)
UNet (Naraloka *et al*., 2022)	**97,62**	89,84	98,37	-	-
Wide UNet (Zhou *et al*., 2018)	76,58	-	-	-	-
MS-UNet (Kushnure & Talbar, 2021)	-	-	-	91	-
VGG-16, ResNet-50V2, dan *UNet++* (Ibrahim *et al*., 2022)	97	58	-	68	68
Segnet-VGG16 (Elnakib *et al*., 2020)	81	-	-	-	-
UNet (Z. Zhang *et al*., 2023)	-	81	-	-	-
VGG-16, *Inception,* dan *UNet*	95,23	**95,28**	**99,91**	**95,20**	**97,54**

Dari Tabel 4.6, terlihat bahwa penelitian ini mencapai hasil yang lebih unggul dalam akurasi, sensitivitas, spesifisitas, *IoU*, dan F1-Score dibandingkan dengan beberapa penelitian sebelumnya. Penelitian ini berhasil mencapai akurasi sebesar 95,23%, sensitivitas sebesar 95,28%, spesifisitas sebesar 99,91%, *IoU* sebesar 95,20%, dan F1-Score sebesar 97,54%, yang dikategorikan sebagai hasil yang sangat baik. Hasil ini mengindikasikan bahwa penggunaan modifikasi arsitektur VGG-16, UNet, dan Inception memberikan performa segmentasi yang optimal, mendekati ground truth, serta memprediksi objek hati juga sangat baik.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini melakukan segmentasi citra dengan menerapkan teknik MPR *slicing* dan kombinasi arsitektur VGG-16, *UNet,* dan *Inception*. Penerapan Teknik MPR *slicing* menghasilkan citra hati dengan tiga sudut pandang berbeda yang dapat mewakilkan seluruh informasi citra secara detail. Proses segmentasi citra dengan arsitektur VGG-16, *Inception*, dan *UNet* menghasilkan nilai akurasi sebesar 95,23%, sensitivitas sebesar 95,28%, spesifisitas sebesar 99,91%, IoU sebesar 95,20% dan F1-Score sebesar 97,54%. Nilai evaluasi kinerja di atas 95% menunjukkan arsitektur VGG-16, *Inception*, dan *UNet* masuk dalam kategori evaluasi kinerja yang sangat baik. Nilai akurasi yang dihasilkan menunjukkan model dapat mengukur tingkat keakuratan citra dengan sangat baik. Nilai sensitivitas dan spesifisitas yang dihasilkan juga menunjukkan kemampuan model dalam mengenali objek hati yang berwarna putih dan *background* yang berwarna hitam sudah sangat baik. Selain itu, nilai *IoU* yang dihasilkan menunjukkan bahwa model dapat memberikan hasil segmentasi yang sesuai dengan *ground truth*.

5.2 Saran

Penelitian ini hanya melakukan segmentasi menggunakan kombinasi arsitektur VGG-16, *UNet,* dan *Inception* tanpa melakukan tahap klasifikasi. Pada penelitian selanjutnya diharapkan dapat melakukan pengembangan yang lebih baik dan dapat melakuakn proses klasifikasi pada hati.

DAFTAR PUSTAKA

Abdalbagi, F., Viriri, S., & Mohammed, M. T. (2020). Batch Normalized Convolution Neural Network for Liver Segmentation. *Signal & Image Processing : An International Journal*, *11*(5), 21–35.

Abrar, I. N., Abdullah, A., & Sucipto, S. (2023). Liver Disease Classification Using the Elbow Method to Determine Optimal K in the K-Nearest Neighbor (K-NN) Algorithm. *Jurnal Sisfokom (Sistem Informasi Dan Komputer)*, *12*(2), 218–228.

Ansari, S. U., Javed, K., Qaisar, S. M., Jillani, R., & Haider, U. (2021). Multiple Sclerosis Lesion Segmentation in Brain MRI Using *Inception* Modules Embedded in a Convolutional Neural Network. *Journal of Healthcare Engineering*, *2021*.

Armansyah, M. A. (2022). Aplikasi Pengolahan Citra Mri Untuk Deteksi Area Kanker Otak Dengan Menggunakan Metode Robinson. *Journal of Informatics, Electrical and Electronics Engineering*, *1*(3), 91–96.

Atabansi, C. C., Chen, T., Cao, R., & Xu, X. (2021). Transfer Learning Technique with VGG-16 for Near-Infrared Facial Expression Recognition. *Journal of Physics: Conference Series*, *1873*(1), 0–11.

Balaha, H. M., & Hassan, A. E. S. (2023). Skin cancer diagnosis based on deep transfer learning and sparrow search algorithm. In *Neural Computing and Applications* (Vol. 35, Issue 1). Springer London.

Bjorck, J., Gomes, C., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. *Advances in Neural Information Processing Systems*, *2018-Decem*(NeurIPS), 7694–7705.

Brilian Argario, H., Hidayat, N., & Kartika Dewi, R. (2018). Implementasi Metode Naive Bayes Untuk Diagnosis Penyakit Kambing (Studi Kasus : UPTD . Pembibitan Ternak dan Hijauan Makanan Ternak Kec. Singosari Malang). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(8), 2719–2723.

Chandra, A., & Hati, A. (2022). Endoscopic ultrasound: a very important tool in detecting small insulinomas. *QJM: An International Journal of Medicine*, *115*(5), 308–309.

Chen, W., Yang, B., Li, J., & Wang, J. (2020). An Approach to Detecting Diabetic Retinopathy Based on Integrated Shallow Convolutional Neural Networks. *IEEE Access*, *8*, 178552–178562.

Dalimunthe, A. H. (2020). Segmentasi Citra MRI dengan Menggunakan Metode BLOB. *Resolusi: Rekayasa Teknik Informatika*, *1*(2), 103–109.

Desiani, A., Erwin, Suprihatin, B., Efriliyanti, F., Arhami, M., & Setyaningsih, E. (2022). VG-DropDNet a Robust Architecture for Blood Vessels Segmentation on Retinal Image. *IEEE Access*, *10*, 92067–92083.

Du, G., Cao, X., Liang, J., Chen, X., & Zhan, Y. (2020). Medical Image Segmentation based on U-Net: A Review. *Journal of Imaging Science and Technology*, *64*(2), 020508-1-020508–020512.

Elnakib, A., Elmenabawy, N., & Moustafa, H. E.-D. (2020). Automated deep system for joint liver and tumor segmentation using majority voting. *MEJ. Mansoura Engineering Journal*, *0*(0), 0–0.

Fernando Ade Pratama, E., Khairil, K., & Jumadi, J. (2022). Implementasi Metode K-Means Clustering Pada Segmentasi Citra Digital. *Jurnal Media Infotama*, *18*(2), 291–301.

Garbin, C., Zhu, X., & Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, *79*(19–20), 12777–12815.

Ibnul Rasidi, A., Pasaribu, Y. A. H., Ziqri, A., & Adhinata, F. D. (2022). Klasifikasi Sampah Organik dan Non-Organik Menggunakan Convolutional Neural Network. *Jurnal Teknik Informatika Dan Sistem Informasi*, *8*(1), 72–81.

Ibrahim, M., Mahmoud, M., Albadawy, R. M., & Abdulkader, H. (2022). Liver Multi-class Tumour Segmentation and Detection Based on Hyperion Pre-trained Models. *International Journal of Intelligent Engineering and Systems*, *15*(6), 392–405.

Iqbal, M. I. (2022). *Deteksi Kerusakan Ban Menggunakan Algoritma Convolutional Neural Network. Undergraduate thesis, UPN Veteran Jawa Timur*. 5–23.

Jadon, S. (2020). A Survey of Loss Functions for Semantic Segmentation. *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB 2020*.

Kalayeh, M. M., & Shah, M. (2019). Training Faster by Separating Modes of Variation in Batch-normalized Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(6), 1483–1500.

Kamrul Hasan, S. M., & Linte, C. A. (2019). U-NetPlus: A Modified Encoder-Decoder U-Net Architecture for Semantic and Instance Segmentation of Surgical Instruments from Laparoscopic Images. *2019 41st Annual*

International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 7205–7211.

Khan, Z., Khan, F. G., Khan, A., Rehman, Z. U., Shah, S., Qummar, S., Ali, F., & Pack, S. (2021). Diabetic Retinopathy Detection Using VGG-NIN a Deep Learning Architecture. *IEEE Access*, *9*, 61408–61416.

Kushnure, D. T., & Talbar, S. N. (2021). MS-*UNet*: A multi-scale *UNet* with feature recalibration approach for automatic liver and tumor segmentation in CT images. *Computerized Medical Imaging and Graphics*, *89*(February), 101885.

Kusuma, T. A. A. H., Usman, K., & Saidah, S. (2021). PEOPLE COUNTING FOR PUBLIC TRANSPORTATIONS USING YOU ONLY LOOK ONCE METHOD. *Jurnal Teknik Informatika (Jutif)*, *2*(1), 57–66.

Lian, S., Li, L., Lian, G., Xiao, X., Luo, Z., & Li, S. (2019). A Global and Local Enhanced Residual U-Net for Accurate Retina Vessel Segmentation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

Lubis, A. J. (2020). Pemanfaatan CT-Scan (Computer Tomography) Dalam Dunia Medis. *Snastikom*, 393–398.

Mei, Y., Jin, H., Yu, B., Wu, E., & Yang, K. (2021). Visual geometry Group-*UNet*: Deep learning ultrasonic image reconstruction for curved parts. *The Journal of the Acoustical Society of America*, *149*(5), 2997–3009.

Naraloka, T., Kesuma, L. I., Sukmawati, A., & Cristianti, M. (2022). Arsitektur U-Net pada Segmentasi Citra Hati sebagai Deteksi Dini Kanker Liver. *Techno.Com*, *21*(4), 753–764.

Özcan, F., Uçan, O. N., Karaçam, S., & Tunçman, D. (2023). Fully Automatic Liver and Tumor Segmentation from CT Image Using an AIM-*UNet*. *Bioengineering*, *10*(2).

Sanjaya, J., & Ayub, M. (2020). Augmentasi Data Pengenalan Citra Mobil Menggunakan Pendekatan Random Crop, Rotate, dan Mixup. *Jurnal Teknik Informatika Dan Sistem Informasi*, *6*(2), 311–323.

Shen, Y., Sheng, V. S., Wang, L., Duan, J., Xi, X., Zhang, D., & Cui, Z. (2020). Empirical comparisons of deep learning networks on liver segmentation. *Computers, Materials and Continua*, *62*(3), 1233–1247.

Siregar, E. S. ., Sutapa, G. N., & Sudarsana, I. W. B. (2020). Analysis of Radiation Dose of Patients on CT Scan Examination using Si-INTAN Application. *Buletin Fisika*, *21*(2), 53.

Sun, W., & Chen, Z. (2020). Learned Image Downscaling for Upscaling Using Content Adaptive Resampler. *IEEE Transactions on Image Processing*, *29*, 4027–4040.

Sunarya, I. M. G., Antara Kesiman, M. W., & Purnami, I. A. P. (2015). Segmentasi Citra Tulisan Tangan Aksara Bali Berbasis Proyeksi Vertikal Dan Horisontal. *Jurnal Informatika*, *9*(1), 982–992.

Syakrani, N., Widhiyasana, Y., & Efendi, A. A. (2018). Deteksi Tumor Hati dengan Graph Cut dan Taksiran Volume Tumornya. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi (JNTETI)*, *7*(1).

Wahyudi, I., Bahri, S., & Handayani, P. (2019). *Aplikasi Pembelajaran Pengenalan Budaya Indonesia*. *V*(1), 135–138.

Wang, C., Gan, M., Zhang, M., & Li, D. (2020). Adversarial convolutional network for esophageal tissue segmentation on OCT images. *Biomedical Optics Express*, *11*(6), 3095.

Wang, J., He, X., Faming, S., Lu, G., Cong, H., & Jiang, Q. (2021). A Real-Time Bridge Crack Detection Method Based on an IMPRoved *Inception*-Resnet-v2 Structure. *IEEE Access*, *9*, 93209–93223.

Wirapati, S., Luh, D., Astuti, G., & Kom, M. (2022). *Sistem Pakar Untuk Membantu Diagnosis Diabetes Menggunakan Machine Learning Dengan Algoritma Jaringan Saraf Tiruan*. *11*(4), 765–772.

Wu, K., Zhang, S., & Xie, Z. (2020). Monocular Depth Prediction with Residual DenseASPP Network. *IEEE Access*, *8*, 129899–129910.

Yu, Q., Xia, Y., Xie, L., Fishman, E. K., & Yuille, A. L. (2019). *Thickened 2D Networks for Efficient 3D Medical Image Segmentation*. *i*.

Zaeemzadeh, A., & Member, S. (2021). Norm-Preservation : Why Residual Networks Can Become Extremely Deep ? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(11), 3980–3990.

Zhang, G., Ge, L., Yang, Y., Liu, Y., & Sun, K. (2019). Fused Confidence for Scene Text Detection via Intersection-over-Union. *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, 1540–1543.

Zhang, Z., Tian, H., Xu, Z., Bian, Y., & Wu, J. (2023). Application of a pyramid pooling *UNet* model with integrated attention mechanism and *Inception* module in pancreatic tumor segmentation. *Journal of Applied Clinical Medical Physics*, *24*(12), 1–10.

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., & Liang, J. (2018). *UNet++*: A Nested U-Net Architecture. In *Deep learning in medical image analysis and multimodal learning for clinical decision support: Vol. 11045 LNCS*. Springer International Publishing.