DETEKSI TUMOR OTAK MENGGUNAKAN MODIFICATION LU-NET BERBASIS CITRA MRI

SKRIPSI

Oleh :
AHMAD FAYYADH QAIMUL HAQ
NIM. 17650091

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2022

DETEKSI TUMOR OTAK MENGGUNAKAN MODIFICATION LU-NET

BERBASIS CITRA MRI

SKRIPSI

Oleh :
AHMAD FAYYADH QAIMUL HAQ
NIM. 17650091

Diajukan kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang
untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S. Kom)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2022

HALAMAN PERSETUJUAN

DETEKSI TUMOR OTAK MENGGUNAKAN MODIFICATION LU-NET BERBASIS CITRA MRI

SKRIPSI

Oleh :
AHMAD FAYYADH QAIMUL HAQ
NIM. 17650091

Telah Diperiksa dan Disetujui untuk Diuji

Tanggal : 09 Desember 2022

Pembimbing I	Pembimbing II
Dr. Irwan Budi Santoso, M.Kom	Dr. M. Amin Hariyadi, M.T
NIP. 19770103 201101 1 004	NIP. 19670018 200501 1 001

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim

Dr. Fachrul Kurniawan M.MT.,IPM
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

DETEKSI TUMOR OTAK MENGGUNAKAN MODIFICATION LU-NET BERBASIS CITRA MRI

SKRIPSI

Oleh:

AHMAD FAYYADH QAIMUL HAQ
NIM. 17650091

Telah Dipertahankan di Depan Dewan Penguji Skripsi dan
Dinyatakan Diterima Sebagai Salah Satu Persyaratan untuk
Memperoleh Gelar Sarjana Komputer (S. Kom)
Tanggal: 16 Desember 2022

Susunan Dewan Penguji

Ketua Penguji	: Dr. Cahyo Crysdian NIP. 19740424 200901 1 008	( )
Anggota Penguji I	: Dr. Ririen Kusumawati, S.Si., M.Kom NIP. 19720309 200501 2 002	( )
Anggota Penguji II	: Dr. Irwan Budi Santoso, M.Kom NIP. 19770103 201101 1 004	( )
Anggota Penguji III	: Dr. M. Amin Hariyadi, M.T NIP. 19670018 200501 1 001	( )

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim

Dr. Fachrul Kurniawan M.MT.,IPM
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama : Ahmad Fayyadh Qaimul Haq
Nim : 17650091
Program Studi : Teknik Informatika
Fakultas : Sains dan Teknologi
Judul Skripsi : Deteksi Tumor Otak Menggunakan *Modification Lu-Net* Berbasis Citra MRI

Menyatakan dengan sebenarnya bahwa skripsi ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilan daya, tulisan atau pikiran orang lain yang saya akui menjadi hasil tulisan atau pikiran saya sendiri , kecuali dengan mencantumkan sumber cuplikan atau referensi pada daftar pustaka.

Apabila pada kemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan saya.

Malang, 16 Desember 2022

Yang Membuat Pernyataan

Ahmad Fayyadh Qaimul Haq
NIM. 17650091

HALAMAN PERSEMBAHAN

Bismillahirrohmanirrohim, Alhamdulillah puji syukur atas nikmat yang telah Allah SWT berikan sehingga penulis dapat menuntaskan skripsi ini untuk menyelesaikan program S1 di kampus Ulul Albab tercinta dengan lancar dan baik pula. Tak lupa pula shalawat serta salam dijunjungkan kepada baginda Rasulullah SAW yang selalu dirindukan dan diidolakan.

Terima kasih banyak kepada kedua orang tua penulis, teruntuk ayah tercinta yaitu Abdul Rahim yang selalu memberikan dorongan dan motivasi serta mendidik anaknya untuk menjadi pribadi yang tangguh, disiplin, jujur dan berwibawa serta harus selalu siap dengan kerasnya kehidupan. Teruntuk ibu tercinta Suti'ah yang selalu mengajarkan kasih sayang, penyabar, dan juga rendah hati kepada anaknya. Terima kasih juga teruntuk kakak laki-laki Muflih Khallab Al Mustaqim yang selalu memberikan motivasi dan bimbingan.

Teruntuk seluruh guru dan dosen mulai dari sekolah dasar hingga perguruan tinggi. Terkhusus untuk dosen pembimbingku bapak Dr. Irwan Budi Santoso, M.Kom dan Dr. M. Amin Hariyadi, M.T yang selalu tulus, ikhlas dan sabar dalam membimbing dan selalu mengarahkan dengan sangat baik. Pesan baik kalian akan selalu kuingat dan tak putus doa untuk seluruh guru dan dosen tercinta hingga akhir hayatku.

Dan teruntuk seluruh rekan Unocore dan rekan tercinta yang tidak bisa penulis sebutkan satu persatu.

HALAMAN MOTTO

"*Be The Change You Want to See in The World*"

- Mahatma Gandhi

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Segala puji bagi Allah subhanahu wa ta'ala yang senantiasa memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi ini dengan baik. Tujuan dari penyusunan skripsi ini guna memenuhi salah satu syarat untuk bisa menempuh ujian sarjana komputer pada Fakultas Sains dan Teknologi (F-SAINTEK) Program Studi Teknik Informatika di Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang. Di dalam pengerjaan skripsi ini telah melibatkan banyak pihak yang sangat membantu dalam banyak hal. Oleh sebab itu, disini penulis sampaikan rasa terima kasih sedalam-dalamnya kepada

1. Prof. Dr. H. M. Zainuddin, MA selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini, M.Si selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Fachrul Kurniawan ST., M.MT., IPM selaku Ketua Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Dr. Irwan Budi Santoso, M.Kom. selaku dosen pembimbing I yang selalu memberikan penulis dorongan, motivasi yang sangat kuat dan selalu membantu penulis dalam menyelesaikan permasalahan yang penulis temukan dalam bimbingan untuk menyelesaikan skripsi ini.

5. Dr. M. Amin Hariyadi, M.T selaku dosen pembimbing II yang selalu membantu penulis dalam menyelesaikan kesulitan dan selalu memberikan solusi setiap masalah yang penulis hadapi.
6. Dr. Cahyo Crysdian, dan Dr. Ririen Kusumawati, M.Kom selaku dosen penguji yang sangat profesional dalam melakukan pengujian terhadap proses ujian skripsi mulai dari seminar proposal hingga sidang skripsi yang berjalan dengan baik.
7. Seluruh jajaran dosen dan staf jurusan teknik informatika UIN Malang baik secara langsung maupun tidak langsung dalam menyelesaikan skripsi ini.
8. Orang tua tercinta serta keluarga yang selalu memberikan semangat dan kasih sayang sehingga penulis dapat menyelesaikan skripsi ini tepat waktu.
9. Rekan-rekan seperjuangan Teknik Informatika 2017 Unocore yang penulis cintai dan penulis banggakan

Malang, 16 Desember 2022

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Haq, Ahmad Fayyadh Qaimul. 2022. **Deteksi Tumor Otak Menggunakan Modification Lu-Net Berbasis Citra MRI.** Skripsi. Jurusan Teknik Informatika, Fakultas Sains dan Teknologi. Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Dr. Irwan Budi Santoso, M.Kom., (II) Dr. M. Amin Hariyadi, M.T.

Kata Kunci: *Deteksi Tumor Otak, Modification Lu-Net, Citra MRI*

Kasus tumor otak memiliki angka yang sangat tinggi, yaitu 34 per 100.000 penduduk per tahun di seluruh dunia. Sedangkan di Indonesia kasus kematian yang disebabkan tumor otak juga tergolong tinggi, yaitu sebanyak 5405 kasus kematian dari 6337 kasus, atau 85,29% di tahun 2016. Diagnosis penyakit pada tumor otak dapat diketahui dengan menggunakan citra MRI. Salah satu pilihan untuk mengolah hasil MRI adalah dengan menggunakan *software* deteksi tumor otak yang bisa mendiagnosis secara otomatis hasil citra MRI. Sistem ini akan dibangun dengan mengimplementasikan algoritma *Modification Lu-Net*, dimana algoritma yang digunakan merupakan hasil modifikasi dari algoritma Lu-*Net* yang dikembangkan Rai & Chatterjee pada tahun 2020. Penelitian ini bertujuan untuk mengukur seberapa tinggi *accuracy, f-score, precision, recall*, dan *specificity* algoritma *Modification Lu-Net* untuk mendeteksi tumor otak berdasarkan citra MRI dan mengetahui faktor-faktor apa yang mempengaruhi hasil *accuracy* algoritma *Modification Lu-Net*. Total data yang digunakan terdiri dari 3929 data citra MRI yang setiap citra memiliki *mask* sebagai *ground truth* atau lokasi objek tumor yang benar, dimana sebanyak 2556 citra MRI yang memiliki tumor otak dan 1373 citra MRI otak normal. Pengujian pertama dilakukan dengan metode *K-Fold Cross Validation* dan pengujian kedua dilakukan dengan beberapa data *training* yang berbeda, beberapa *leraning rate*, dan beberapa *optimizer*. Penelitian ini berhasil memperoleh hasil lebih baik dibandingkan dengan penelitian yang dilakukan Rai & Chatterjee di tahun 2020, dimana penelitian ini menghasilkan nilai *recall* 81,96%, *precision* 86,03%, *specificity* 99,83%, *accuracy* 99,64%, dan *f-score* 83,91%. Pengujian kedua mendapatkan hasil bahwa semakin banyak jumlah data *training* yang digunakan maka hasil *accuracy* akan semakin baik, selain itu pemilihan *learning rate* dan *optimizer* yang tepat akan mampu menghasilkan nilai *accuracy* yang lebih baik.

ABSTRACT

Haq, Ahmad Fayyadh Qaimul. 2022. **Brain Tumor Detection Using Modification Lu-Net With MRI Image Based.** Undergraduate Thesis. Informatics Engineering Department, Faculty of Science and Technology. Islamic State University Maulana Malik Ibrahim Malang. Supervisor: (I) Dr. Irwan Budi Santoso, M.Kom., (II) Dr. M. Amin Hariyadi, M.T.

Keywords*: Brain Tumor Detection, Modification Lu-Net, MRI Image*

Cases of brain tumors have a very high rate, which is 34 per 100,000 population per year worldwide. Meanwhile, in Indonesia, cases of death caused by brain tumors are also relatively high, namely as many as 5405 cases of death out of 6337 cases, or 85.29% in 2016. Diagnosis of brain diseases including brain tumors can be known using MRI imagery. One option to process MRI results is to use brain tumor detection software that can automatically diagnose MRI image results. This system will be built by implementing the Modification Lu-Net algorithm, where the algorithm used is the result of a modification of the Lu-Net algorithm developed by Rai & Chatterjee in 202. This study aims to measure how high the accuracy, f-score, precision, recall, and specificity of the Modification Lu-Net algorithm is to detect brain tumors based on MRI images and find out what factors influence the accuracy results of the Modification Lu-Net algorithm. The total data used consisted of 3929 MRI image data, each image of which had a mask as the correct ground truth or location of tumor objects, of which 2556 MRI images had brain tumors and 1373 normal brain MRI images. The first test was conducted with the K-Fold Cross Validation method and the second test was conducted with several different training data, some learning rates, and several optimizers. This study managed to obtain better results compared to the research conducted by Rai & Chatterjee in 2020, where this study produced a recall value of 81.96%, a precision of 86.03%, a specificity of 99.83%, an accuracy of 99.64%, an f-score of 83.91%. The second test found that the more the amount of training data used, the better the accuracy results will be, besides that the selection of the right learning rate and the optimizer will be able to produce better accuracy.

مستخلص البحث

حق، أحمد فياض قيمول. 2022. **الكشف عن ورم الدمغي باستخدام *Modification Lu-Net* القائم على صورة التصوير بالرنين المغناطيسي**. اطروحه. قسم الهندسة المعلوماتية، كلية العلوم والتكنولوجيا. جامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف: (ط) د. إيروان بودي سانتوسو، م. كوم، (الثاني) د. م. أمين هاريادي، م. ت.

الكلمات الدالة: كشف ورم الدمغي ، *Modification Lu-Net* ، صورة التصوير بالرنين المغناطيسي

ان ورم الدمغي هو مسألة بعدد عالي جدا يعنى 34 من 100000 من السكان العالمي في السنة. واما في إندونيسيا هذه المسألة كثيرة بأن تسبب بوفات الانسن ، في بحث *Patel* في عام 2019 ، كان في إندونيسيا عام 2016 السكانه ماتوا بسبب ورم الدمغي بعدد5405 من 6337 حالات الوفاة ، أو 85.29 ٪ نسبة الموت من جميع الحالات. عرف عن التشخيص أمراض الدماغ وكذالك بورم الدمغي باستخدام صور التصوير بالرنين المغناطيسي. خيارمن خيارات المجاهز على النتائج التصوير بالرنين المغناطيسي يعنى بإستعمال البرنامج الجاهزة لكشف ورم الدمغي الذي يكشف نتائج صور التصوير بالرنين المغناطيسي تلقائيا أو أوتوماتيكيا. سيبنهذا النظام بإطباق خوارزمية *Modification Lu-Net*, كانت هذه الخوارزمية هي النتيجة التعديل من خوارزمية *Lu-Net* التي طورتها *Rai & Chatterjee* في عام 2020. الهدف من هذا البحث ليقيس الإرتفاع من استرجاع دقة ، دقة ، درجة ، خصوصية خوارزمية *Modification Lu-Net* للكشف عن ورم الدمغي بصور التصوير من الرنين المغناطيسي وليعرف العوامل التي تؤثر على نتيجة صحة خوارزمية *Modification Lu-Net*. المجموع من البيانات التي تفيدبها يعنى3929 من صورة تصوير بالرنين المغناطيسي ، و لكل صورة كا الحقيقة الأساسيةأومواقع الورم الصحيحا ، منها 2556 صورة بالرنين المغناطيسي بها أورام دماغية و 1373 صورة تصوير بالرنين المغناطيسي للدماغ سالما. كانت تجربة الأولى بطريقة *K-Fold Cross Validation* وتم إجراء الاختبار الثاني باستخدام العديد من بيانات التدريب المختلفة ، وبعض معدلات التنفيذ ، والعديد من المحسنات. تمكنت هذه الدراسة من الحصول على نتائج أفضل مقارنة بالبحث الذي أجراه *Rai & Chatterjee* في عام 2020، حيث أنتجت هذه الدراسة قيمة استدعاء 81.96٪ ، والدقة 86.03٪ ، والنوعية 99.83٪ ، والدقة 99.64٪ ، ودرجة 83.91٪. وجد الاختبار الثاني أنه كلما زادت كمية بيانات التدريب المستخدمة ، كانت نتائج الدقة أفضل ، إلى جانب أن اختيار معدل التعلم الصحيح والمحسن سيكون قادرا على إنتاج قيمة دقة أفضل.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan dunia medis berkembang dengan pesat mengikuti perkembangan teknologi dalam segala lini. Perkembangan teknologi memiliki peran aktif dalam dunia kedokteran yang mempermudah dokter dalam mendiagnosa suatu penyakit. Salah satu penyakit yang biasa dilakukan diagnosa dari dokter adalah tumor otak.

Tumor otak adalah pertumbuhan jaringan akibat adanya sel-sel abnormal di otak dan di sekitar otak. Penyebabnya masih belum diketahui. Berdasarkan hasil dugaan para peneliti, tumor otak diduga disebabkan oleh faktor keturunan dan akibat paparan radiasi bahan kimia berbahaya. Tumor otak juga bisa dialami seseorang akibat penyebaran sel kanker dari bagian tubuh lain yang kemudian menyebar ke otak (metastasis). Untuk itu perlu dilakukan diagnosis yang berkaitan dengan tumor otak.

Kasus tumor otak memiliki angka yang tinggi, dalam penelitian Aman *et al.* (2016) di Amerika Serikat kasus tumor otak terdapat 2.142 per 100.000 penduduk per tahun, sedangkan untuk angka kasus tumor otak di seluruh dunia berdasarkan angka standar populasi dunia adalah 34 per 100.000 penduduk per tahun. Sedangkan di Indonesia kasus kematian yang disebabkan tumor otak cukup tinggi, menurut penelitian Patel *et al,* (2019), pada tahun 2016 di Indonesia kasus kematian

yang disebabkan tumor otak terdapat 5405 kasus kematian dari 6337 kasus, atau 85,29% pasien meninggal dari seluruh kasus.

Diagnosis penyakit pada otak termasuk tumor otak dapat diketahui oleh para radiolog dan dokter ahli dengan menggunakan MRI. MRI adalah metode non-invasif untuk memetakan struktur internal dan aspek fungsi tertentu di dalam tubuh (Wen *et al*., 2010). MRI menggunakan radiasi frekuensi radio (RF) pada medan magnet dan dikontrol dengan baik untuk menghasilkan gambar berkualitas tinggi. Gambar MRI dibuat dengan menempatkan pasien di dalam magnet besar, yang menginduksi medan magnet eksternal yang relatif kuat. Hal ini menyebabkan inti banyak atom dalam tubuh, termasuk Hidrogen, menyelaraskannya dengan medan magnet dan kemudian penerapan sinyal RF, Energi dilepaskan dari tubuh, dideteksi dan digunakan untuk membangun citra MR oleh Komputer (Wen *et al*., 2010).

Convolutional Neural Network (ConvNet/CNN) adalah *algoritma Deep Learning* yang dapat mengambil input gambar, menetapkan kepentingan (bobot dan bias yang dapat dipelajari) ke berbagai aspek/objek dalam suatu gambar dan dapat membedakan satu dari yang lain. Pra-pemrosesan yang diperlukan di CNN jauh lebih sedikit daripada algoritma klasifikasi lainnya. Sedangkan pada metode primitif, filter direkayasa dengan tangan, dengan pelatihan yang memadai, CNN memiliki kemampuan untuk mempelajari filter/karakteristik tersebut. (Wu, 2017).

Arsitektur CNN dapat di analogikan dengan pola konektivitas Neuron di Otak Manusia dan terinspirasi oleh Visual Cortex. Neuron individu merespons rangsangan hanya di wilayah terbatas bidang visual yang dikenal sebagai Bidang

Reseptif. Kumpulan bidang tersebut tumpang tindih untuk menutupi seluruh area visual (Wu, 2017).

Berdasarkan penelitian Agostinelli (2015), algoritma *Convolutional Neural Network* mampu menghasilkan nilai akurasi yang tinggi. Salah satu model *CNN* yang memiliki performa tinggi adalah *Lu-Net, Lu-Net* menggunakan lapisan konvolusi untuk melakukan deteksi. Jaringan simetris dan dapat dibagi menjadi dua bagian: *encoder* dan *decoder*. Berdasarkan penelitian Rai & Chatterjee (2020), model *Lu-Net* mampu menghasilkan performa yang tinggi dibandingkan dengan Model *Le-Net* dan *VGG-16*. Akurasi keseluruhan model *Le-Net, VGG-16* dan *Lu-Net* masing-masing adalah 88%, 90%, dan 98%. Penelitian ini melanjutkan modifikasi Lu-Net yang dikembangkan oleh Rai & Chatterjee (2020).

خَيْرُ الناسِ أَنْفَعُهُمْ لِلناسِ

"Sebaik-baik manusia adalah yang paling bermanfaat bagi manusia"
(HR. Ahmad, ath-Thabrani, ad-Daruqutni).

Dari hadist di atas dijelaskan bahwa sebaik-baiknya manusia adalah yang bermanfaat bagi manusia lain. Oleh karena itu output dari penelitian Deteksi Tumor Otak Berdasarkan Citra MRI Menggunakan Algoritma Convolutional Neural Network adalah hasil diagnosa tumor otak dari citra MRI.

Prinsip dalam penelitian ini adalah mengenali tumor dari pemindaian MRI citra otak menggunakan teknik pemrosesan citra digital dan menghitung area

tumor dengan proses otomatis penuh dan analisis simetrinya. Data yang diinput nantinya akan diproses menggunakan algoritma *CNN* berdasarkan dataset yang telah diatur sebelumnya sehingga software bisa mendiagnosa dengan tepat.

1.2 Pernyataan Masalah

Berdasarkan latar belakang, penelitian ini mengangkat masalah:

1. Seberapa tinggi nilai *accuracy, f-score, precision, recall*, dan *specificity* algoritma *modification Lu-Net* untuk mendeteksi tumor otak berdasarkan citra MRI?
2. Faktor-faktor apa yang mempengaruhi hasil *accuracy* Deteksi Tumor Otak menggunakan *Modification Lu-Net* berbasis citra MRI?

1.3 Tujuan Penelitian

Tujuan yang didapat dari penelitian ini adalah:

1. Mengetahui seberapa tinggi *accuracy, f-score, precision, recall,* dan *specificity* algoritma *modification Lu-Net* untuk mendeteksi tumor otak berdasarkan citra MRI.
2. Mengetahui faktor-faktor yang mempengaruhi hasil *accuracy* Deteksi Tumor Otak menggunakan *Modification Lu-Net* berbasis citra MRI

1.4 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian Deteksi Tumor Otak menggunakan *Modification Lu-Net* berbasis Citra MRI adalah:

1. Penelitian ini dapat dikembangkan dalam bidang IOT untuk mendiagnosis hasil MRI tumor otak secara otomatis.

2. Sebagai sumber referensi bagi para peneliti lain yang melakukan penelitian dalam bidang yang berkaitan.

1.5 Batasan Penelitian

1. Data yang digunakan pada penelitian ini hanya berfokus pada citra digital hasil MRI yang dapat diakses secara publik melalui *website* Kaggle.com.
2. Penelitian ini hanya mendeteksi tumor otak dan belum bisa mendeteksi segala jenis tumor.

BAB II

STUDI PUSTAKA

2.1 Perkembangan Penelitian Deteksi Tumor

Pada saat ini penelitian deteksi tumor sudah banyak dilakukan, dalam penelitian Kurmi & Chaurasia (2020), Deteksi citra (MRI) adalah proses penting untuk visualisasi dan pemeriksaan jaringan abnormal, terutama selama analisis klinis. Kompleksitas dan variasi struktur tumor memperbesar tantangan dalam deteksi otomatis tumor otak di MRI. Eksperimen yang dilakukan menggunakan lima set data MRI. Akurasi rata-rata segmentasi dan klasifikasi, masing-masing adalah 94,5 dan 91,76%,.

Dalam penelitian Sarkar *et al*. (2020), mengenai deteksi tumor otak dengan CNN. Glioma dan tumor otak metastatik ditemukan mewakili 30% dari semua tumor otak yang didiagnosis pada manusia. Magnetic resonance imaging (MRI) telah menjadi salah satu yang paling efektif untuk diagnosis klinis. Dataset yang digunakan dalam penelitian ini berisi data dari tiga tumor otak yang paling sering didiagnosis yaitu, glioma, meningioma dan tumor hipofisis. Untuk tujuan klasifikasi, Jaringan Neural Convolutional 2D (CNN) dirancang yang mendorong akurasi keseluruhan 91,3% dan penarikan kembali 88%, 81% dan 99% untuk mendeteksi meningioma, glioma dan tumor hipofisis masing-masing.

Dalam penelitian Lou *et al*. (2021), Convolution Neural Network (CNN) telah membawa terobosan di bidang segmentasi citra, terutama untuk citra medis.

U-Net merupakan salah satu model CNN yang banyak digunakan. U-Net tidak hanya

berkinerja baik dalam mensegmentasi citra medis multimodal secara umum, tetapi juga dalam beberapa kasus sulit. Namun, arsitektur U-Net klasik memiliki keterbatasan dalam beberapa aspek, sehingga perlu adanya modifikasi dan menciptakan arsitektur CNN baru yang efektif dan membangun U-Net berdasarkan CNN ini. penelitian ini menggunakan tiga kumpulan data dengan kasus yang sulit dan telah memperoleh peningkatan relatif dalam kinerja masing-masing 2,90%, 1,49% dan 11,42% dibandingkan dengan U-Net klasik..

Menurut Rai & Chatterjee (2020), Identifikasi tumor dalam pikiran manusia dari gambar MR pada tahap awal memainkan peran penting dalam diagnosis penyakit tersebut. Karya ini menyajikan jaringan Deep Neural baru dengan jumlah lapisan yang lebih sedikit dan kurang kompleks dalam dirancang bernama U-Net (LU-Net) untuk mendeteksi tumor. Gambar MR diubah ukurannya, dipotong, diproses, dan ditambah untuk pelatihan yang akurat dan cepat dari model deep neural. Kinerja model Lu-Net dievaluasi menggunakan lima jenis metrik penilaian statistik Precision, Recall, Specificity, F-score, dan Accuracy, dan dibandingkan dengan dua jenis model Le-Net dan VGG-16 lainnya. Model CNN dilatih dan diuji pada gambar tambahan dan validasi dilakukan pada 50 data yang tidak terlatih. Akurasi keseluruhan model Le-Net, VGG-16 dan Lu_Net yang diusulkan masing-masing adalah 88%, 90%, dan 98%.

2.2 Pengertian Tumor Otak

Tumor otak adalah massa atau pertumbuhan sel atau jaringan yang tidak normal pada otak. Ada banyak jenis tumor otak yang berbeda. Beberapa tumor otak bersifat non-kanker (jinak), dan beberapa tumor otak bersifat kanker (ganas).

Tumor otak dapat tumbuh di otak secara alami (tumor otak primer), atau dapat tumbuh di bagian lain dari tubuh dan menyebar ke otak atau disebut sebagai tumor otak sekunder (metastasis) (McFaline-Figueroa & Lee, 2018).

Seberapa cepat tumor otak tumbuh bisa sangat bervariasi. Tingkat pertumbuhan serta lokasi tumor otak menentukan bagaimana hal itu akan mempengaruhi fungsi sistem saraf Anda. Pilihan pengobatan tumor otak tergantung pada jenis tumor otak yang Anda miliki, serta ukuran dan lokasinya (McFaline-Figueroa & Lee, 2018).

1. Penyebab Tumor Otak

Tumor otak primer berasal dari otak itu sendiri atau di jaringan yang dekat dengannya, seperti di selaput penutup otak (meninges), saraf kranial, kelenjar pituitari atau kelenjar pineal. Tumor otak primer dimulai ketika sel-sel normal mengembangkan perubahan (mutasi) dalam DNA mereka. DNA sel berisi instruksi yang memberi tahu sel apa yang harus dilakukan. Mutasi memberitahu sel-sel untuk tumbuh dan membelah dengan cepat dan untuk terus hidup ketika sel-sel sehat akan mati. Hasilnya adalah massa sel abnormal, yang membentuk tumor. Pada orang dewasa, tumor otak primer jauh lebih jarang daripada tumor otak sekunder, di mana kanker dimulai di tempat lain dan menyebar ke otak (Aman *et al*., 2016).

Menurut Aman *et al*. (2016) ada banyak jenis tumor otak primer yang berbeda. Masing-masing mendapatkan namanya dari jenis sel yang terlibat. Contohnya meliputi:

a. Glioma. Tumor ini dimulai di otak atau sumsum tulang belakang dan termasuk astrositoma, ependymoma, glioblastoma, oligoastrocytomas, dan oligodendroglioma.

b. Meningiomas. Meningioma adalah tumor yang muncul dari selaput yang mengelilingi otak dan sumsum tulang belakang (meninges). Kebanyakan meningioma tidak bersifat kanker.

c. Acoustic neuromas (schwannomas). Ini adalah tumor jinak yang berkembang pada saraf yang mengontrol keseimbangan dan pendengaran yang mengarah dari telinga bagian dalam ke otak Anda.

d. Pituitary adenomas. Ini adalah tumor yang berkembang di kelenjar pituitari di dasar otak. Tumor ini dapat mempengaruhi hormon hipofisis dengan efek ke seluruh tubuh.

e. Medulloblastomas. Tumor otak kanker ini paling sering terjadi pada anak-anak, meskipun dapat terjadi pada usia berapa pun. Medulloblastoma dimulai di bagian punggung bawah otak dan cenderung menyebar melalui cairan tulang belakang.

f. Germ cell tumors. Tumor sel germinal dapat berkembang selama masa kanak-kanak di mana testis atau ovarium akan terbentuk. Tapi terkadang tumor sel germinal mempengaruhi bagian lain dari tubuh, seperti otak.

g. Craniopharyngiomas. Tumor langka ini mulai di dekat kelenjar pituitari otak, yang mengeluarkan hormon yang mengontrol banyak fungsi tubuh. Saat craniopharyngioma tumbuh perlahan, hal itu dapat mempengaruhi kelenjar pituitari dan struktur lain di dekat otak.

2.3 Convolutional Neural Network

CNN adalah salah satu algoritma pembelajaran terbaik untuk memahami konten gambar dan telah menunjukkan kinerja yang patut dicontoh dalam deteksi gambar, klasifikasi, deteksi, dan tugas terkait pengambilan (Cireşan *et al*., 2012). Keberhasilan CNN telah menarik perhatian di luar akademisi. Di industri, perusahaan seperti *Google, Microsoft, AT&T, NEC, dan Facebook* telah mengembangkan kelompok penelitian aktif untuk mengeksplorasi arsitektur baru CNN (Deng & Yu, 2013). Saat ini, sebagian besar pelopor kompetisi pemrosesan gambar dan *Computer Vision* (CV) menggunakan model berbasis CNN yang mendalam.

CNN dianggap sebagai salah satu teknik *Machine Learning* yang paling banyak digunakan, terutama dalam aplikasi yang berhubungan dengan *computer vision*. CNN dapat mempelajari representasi dari data seperti grid, dan baru-baru ini telah menunjukkan peningkatan kinerja yang substansial dalam berbagai aplikasi *Machine Learning*. Karena CNN memiliki generasi fitur yang baik dan kemampuan diskriminasi, oleh karena itu dalam sistem *Machine Learning* yang khas, kemampuan CNN dieksploitasi untuk pembuatan fitur dan klasifikasi (Khan *et al*., 2020).

2.4 Desain Arsitektur *Lu-Net*

Algoritma Lu-Net mirip dengan FCN dan SegNet, LU-Net menggunakan lapisan konvolusi untuk melakukan segmentasi semantik. Jaringan simetris dan dapat dibagi menjadi dua bagian: encoder dan decoder. Encoder mengikuti arsitektur khas jaringan convolutional, yang digunakan untuk mengekstrak fitur

spasial dari gambar. Sebuah blok konvolusi melibatkan urutan dua operasi konvolusi 3 × 3, diikuti oleh operasi max-pooling dengan ukuran pooling 2 × 2 dan langkah 2. Blok ini diulang empat kali; dan, setelah setiap down-sampling, jumlah filter dalam konvolusi menjadi dua kali lipat. Akhirnya, perkembangan dua operasi konvolusi 3 × 3 menghubungkan encoder ke decoder (Lou *et al*., 2021). Semua convolutional layer pada U-Net menggunakan ReLU (Rectified Linear Unit) sebagai fungsi aktivasi, kecuali layer terakhir menggunakan convolutional layer 1×1, dan fungsi aktivasi Sigmoid (Agostinelli *et al*., 2015).

Selain itu, arsitektur U-Net memperkenalkan koneksi lewat untuk mentransfer output dari encoder ke decoder. Fitur map ini digabungkan dengan output dari operasi up-sampling; dan fitur map gabungan disebarkan ke lapisan yang berurutan. Koneksi lewati memungkinkan jaringan untuk mengambil fitur spasial yang hilang oleh operasi penyatuan (Lou *et al*., 2021).

2.5 K Fold Cross Validation

Cross Validation adalah prosedur pengambilan sampel ulang yang digunakan untuk mengevaluasi model machine learning pada sampel data terbatas. Sejak diperkenalkannya leave-one-out cross-validation (LOOCV) oleh Stone (1974), validasi silang telah menjadi salah satu metode yang paling populer untuk pemilihan model. Validasi silang (Cross Validation) menunjukan hasil yang memuaskan dalam berbagai situasi seperti kuadrat terkecil standar, data yang berkorelasi, dan untuk bandwidth seleksi dalam estimasi kepadatan (Carmack *et al*., 2012).

Untuk mengurangi biaya komputasi LOOCV, Geisser (1975) mengusulkan validasi silang K-fold. Ini mempartisi kumpulan data menjadi K dengan ukuran yang hampir sama, kemudian (K 1) fold digunakan untuk membangun model, sedangkan sampel yang ditinggalkan digunakan untuk memvalidasi. Selama iterasi prosedur ini untuk K kali, masing-masing lipatan K secara berurutan ditetapkan sebagai data validasi. Dalam prakteknya, pilihan K adalah antara 5 dan 10, sedangkan K <5 dapat menyebabkan masalah (Jung, 2018).

BAB III

DESAIN DAN IMPLEMENTASI

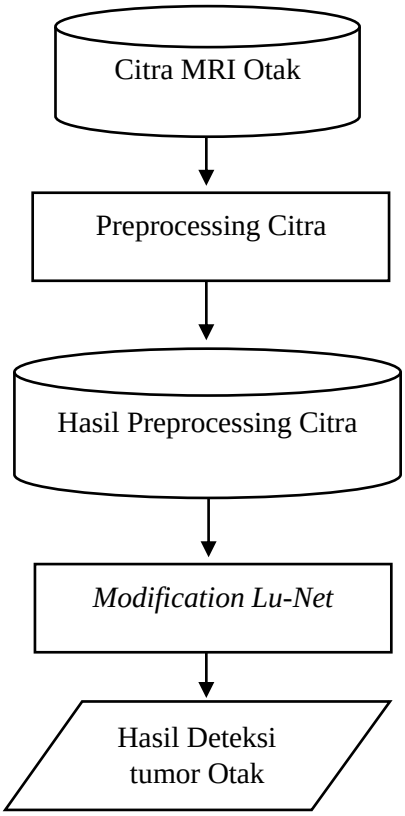
Bab ini akan menjelaskan tentang bagaimana perancangan system dan analisa pada penelitian ini. Dalam penelitian ini memiliki beberapa tahapan yaitu tahapan kebutuhan system yang akan dibuat dan solusi masalah Deteksi Tumor Otak Menggunakan Algoritma *Modification Lu-Net* Berbasis Citra MRI.

3.1 Pengumpulan Data

Dalam penelitian ini terdapat beberapa data yang diperlukan untuk mempersiapkan penelitian ini. Penelitian ini menggunakan data primer dan data sekunder. Data primer adalah data yang didapatkan langsung oleh peneliti, dimana data didapatkan dari website https://www.kaggle.com. Pada penelitian ini data primer yang digunakan merupakan data hasil scan MRI otak pasien yang digunakan untuk *dataset* dan untuk uji coba testing pada sistem. Data sekunder, adalah data yang didapatkan dari lembaga/pihak/peneliti lainnya. Dalam penelitian ini peneliti menggunakan sumber dari jurnal internasional dan buku sebagai bahan penunjang pada penelitian ini.

3.2 Desain Sistem

Dalam melakukan penelitian perlu adanya sebuah desain sistem yang dibangun sebelum melakukan penelitian. Dalam desain sistem ini nantinya akan menjadi gambaran secara keseluruhan tentang sistem yang akan dibangun. Implementasi algoritma *Modification Lu-Net* juga direpresentasikan dalam desain sistem pada gambar 3.1.



Gambar 3.1 Desain Sistem

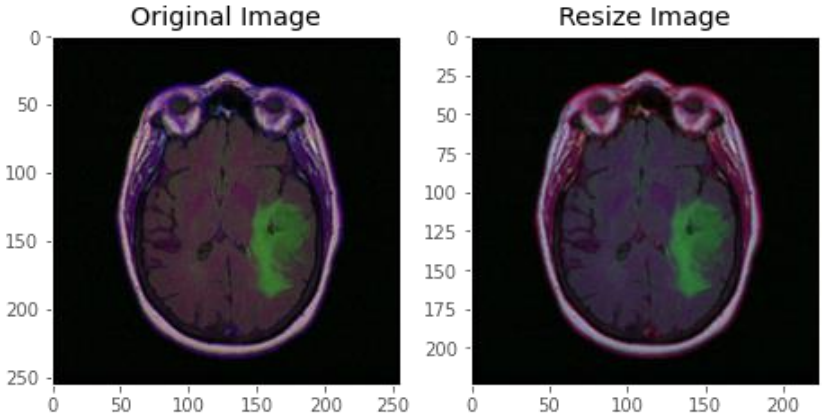
Dalam subbab pengumpulan data telah dijelaskan bahwa data yang telah dikumpulkan akan diproses pada *preprocessing* citra dengan menyamakan ukuran citra dan hasil *preprocessing citra* akan menjadi *input* yang akan digunakan pada sistem. Data citra hasil *preprocessing* diinput dan selanjutnya diproses pada bagian *image processing* menggunakan algoritma *Modification Lu-Net*. Setelah bagian *image processing* selesai maka akan menghasilkan output berupa hasil diagnosa tumor dan non-tumor dari input citra MRI.

3.2.1 *Input Image*

Dari data *image* MRI yang diperoleh kemudian data akan digunakan sebagai data sistem yang akan dibangun. Dalam penggunaan data MRI digunakan sebagai dua inputan, inputan data *image* sebagai data *training* dan inputan data *image* sebagai data *testing.*

3.2.2 *Preprocessing* Citra

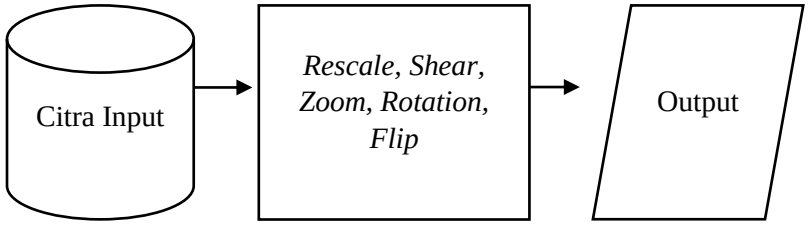
Preprocessing citra adalah langkah-langkah yang diambil untuk memformat gambar sebelum digunakan pada *training* model. *Preprocessing* citra pada peneilitan ini menggunakan pengubahan ukuran citra dengan cara mendefinisikan ukuran piksel citra dengan ukuran 224x224 pada awal program. Hal ini berfungsi untuk mengurangi waktu *training* model dan meningkatkan kecepatan inferensi model. *Fully connected layer* dalam *convolutional neural network*, mengharuskan semua gambar memiliki ukuran array yang sama. Jika ukuran gambar tidak sama, maka akan mempengaruhi hasil deteksi citra. Untuk perbandingan gambar asal dan hasil dari *preprocesing* dapat dilihat pada gambar 3.2.



Gambar 3.2 Data Asli dan Data Setelah Resize

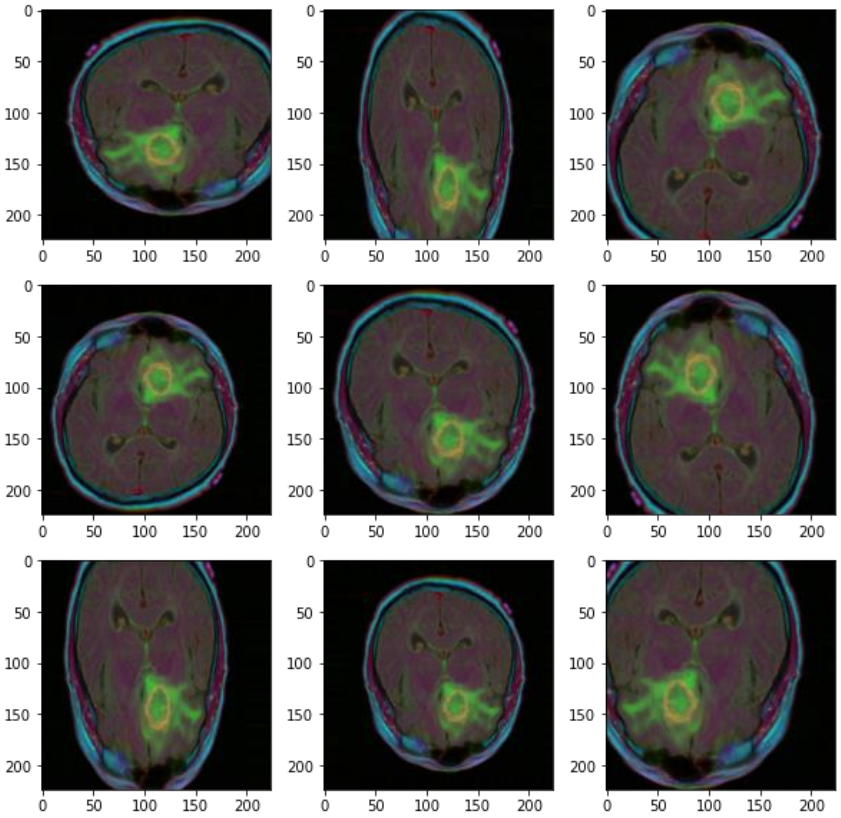
3.2.3 *Augmentation* Citra

Pada penelitian ini tahap setelah *preprocessing* citra adalah *augmentasi* citra. Setelah citra disamakan ukuran pikselnya pada tahap *preprocesiing* citra maka selanjutnya akan diproses dengan *augmentasi* citra.



Gambar 3.3 Augmentasi Citra

Augmentasi citra merupakan teknik manipulasi data yang berguna untuk memperbanyak data tanpa kehilangan inti data tersebut sehingga algoritma bisa menghasilkan performa yang maksimal. Adapun hasil augmentasi citra dapat dilihat pada gambar 3.4 sedangkan implementasi dalam program bisa dilihat pada gambar 3.5.



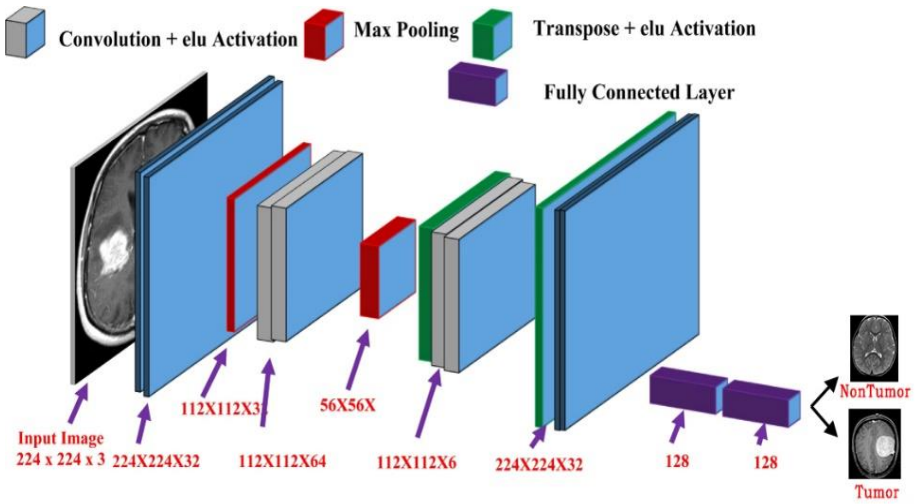
Gambar 3.4 Hasil Augmentasi

```
train_generator_args = dict(rotation_range=0.2,
                            width_shift_range=0.05,
                            height_shift_range=0.05,
                            shear_range=0.05,
                            zoom_range=0.05,
                            horizontal_flip=True,
                            fill_mode='nearest')
```

Gambar 3.5 Kode Program Augmentasi Citra

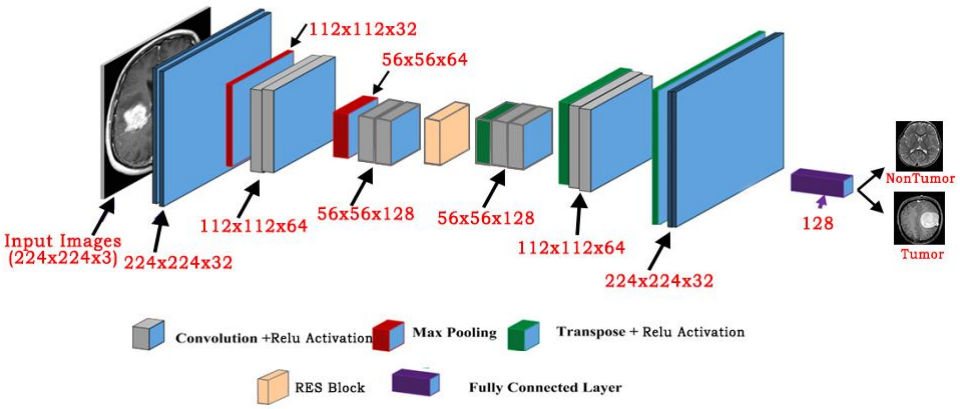
3.2.4 Implementasi *Modification Lu-Net*

Arsitektur *Modification Lu-Net* yang digunakan dalam penelitian ini berasal dari penelitian Rai & Chatterjee (2020) yang terdapat dalam gambar 3.6, dan kemudian dikembangkan oleh peneliti menjadi seperti gambar 3.7.



Gambar 3.6 Arsitektur Lu-Net Rai & Chatterjee (2020)

Arsitektur *Modification Lu-Net* dalam penelitian ini memiliki 20 layer, yang terdiri dari 12 layer Konvolusi, 2 layer Max pooling, 1 layer RES block, 3 layer Transposed konvolusi dan 2 layer Fully connected. Adapun susunan layer pada arsitektur *Modification Lu-Net* terdapat dalam gambar 3.5.



Gambar 3.7 Arsitektur *Modification Lu-Net*.

Tabel 3.1 Perbandingan Arsitektur Lu-Net Rai & Chatterjee (2020) dan *Modification Lu-Net*

Langkah Sistem	***Lu-Net* Rai & Chatterjee (2020)**		***Modification Lu-Net***	
	Karakteristik	**Feature Map**	**Karakteristik**	**Feature Map**
1	Input Citra MRI	224x224x3	Input Citra MRI	224x224x3
2	2x Convuliton Layer	224x224x32	2x Convuliton Layer	224x224x32
3	Max Pooling	112x112x32	Max Pooling	112x112x32
4	2x Convuliton	112x112x64	2x Convuliton Layer	112x112x64

	Layer			
5	Max Pooling	56x56x64	Max Pooling	56x56x64
6	- Upsampling - 2x Convuliton Layer	112x112x64	2x Convuliton Layer	56x56x128
7	- Upsampling - 2x Convuliton Layer	224x224x32	- Upsampling with ResBlock - 2x Convuliton Layer	56x56x128
8	2x Fully Connected Layer	128	- Upsampling with ResBlock - 2x Convuliton Layer	112x112x64
9			- Upsampling with ResBlock - 2x Convuliton Layer	224x224x32
10			Fully Connected Layer	128

Pengembangan arsitektur Lu-Net pada penelitian ini meliputi penambahan dua lapis konvolusi, penambahan RES Block dan Conv2DTranspose, selain itu fungsi aktivasi pada tiap lapis konvolusi dan Conv2DTranspose juga diubah menggunakan fungsi aktivasi Relu. Penambahan RES Block dilakukan karena menurut (Rehman *et al*., 2020), daerah tumor otak memiliki variasi ukuran yang tinggi di mana luas residual melewati proses agregasi kontekstual pada beberapa skala, yang menjadikannya skala-invarian. RES Block meningkatkan bidang reseptif yang valid, dan memungkinkan *Modification Lu-Net* untuk memiliki hasil yang lebih baik. Penggunaan kode program algoritma *Modification Lu-Net* dapat dilihat pada gambar 3.8 sedangkan penjelasan mengenai sistem algoritma dijelaskan pada paragraf setelahanya.

```
def unet(input_size=(224, 224, 3)):
    inputs = Input(input_size)

    conv_1 = Conv2D(32, 3, activation='relu', padding='same',
              kernel_initializer ='he_normal')(inputs)
    conv_1 = BatchNormalization()(conv_1)
    conv_1 = Conv2D(32, 3, activation='relu', padding='same',
              kernel_initializer='he_normal')(conv_1)
    conv_1 = BatchNormalization()(conv_1)
    pool_1 = MaxPool2D((2,2))(conv_1)

    conv_2 = resblock(pool_1, 64)
    pool_2 = MaxPool2D((2,2))(conv_2)

    conv_3 = resblock(pool_2, 128)
    up_3   = upsample_concat(conv_3, conv_2)
    up_3   = resblock(up_3, 64)

    up_4   = upsample_concat(up_3, conv_1)
    up_4   = resblock(up_4, 32)

    # final output
    out    = Conv2D(1, (1,1), kernel_initializer='he_normal',
              padding='same',activation='sigmoid')(up_4)
```

Gambar 3.8 Kode Program Algoritma *Modification Lu-Net*

1. Input citra MRI berukuran piksel 224x224x3 dimana 224x224 adalah ukuran piksel citra, sedangkan 3 merupakan jumlah channel yang merupakan citra Red, Green, Blue (RGB), citra ini kemudian diproses dengan dua lapisan konvolusi berukuran 2x2 dan menghasilkan feature map berukuran 224x224x32, dimana hasil 32 merupakan banyaknya chanel setelah citra dikonvolusi dengan rumus seperti pada persamaan 3.9.
2. Pada proses kedua, feature map hasil proses pertama diproses dengan 1 lapisan max-pooling berukuran 2x2 dan diproses dengan RES Block sehingga menghasilkan feature map berukuran 112x112x64.
3. Kemudian pada proses ketiga, feature map hasil konvolusi kedua diproses dengan 1 lapisan max-pooling berukuran 2x2 dan diproses dengan RES Block

sehingga menghasilkan feature map berukuran 56x56x128. Adapun rumus konvolusi dan max-pooling terdapat pada persamaan 3.1 dan 3.2.

$$f_l^k(p,q) = \sum_c \sum_{x,y} i_c(x,y).\, e_l^k\,(u,v) \tag{3.1}$$

$$\mathbf{Z}_l^k = g_p\,(\boldsymbol{F}_l^k) \tag{3.2}$$

4. Selanjutnya feture map hasil proses terakhir di proses dengan RES Block dan outputnya masuk pada bagian sisi decoder. Pada sisi decoder terdiri dari 3 blok, blok pertama diproses dengan 1 lapisan transpose dan diproses dengan RES Block yang menghasilkan feature map berukuran 56x56x128. Adapun rumus dari RES Block terdapat pada persamaan 3.3.

$$x_i = F(x_i, W_i),\ x_{i+1} = x_i + F(x_i, W_i) \tag{3.3}$$

5. Selanjutnya blok kedua diproses dengan 1 lapisan transpose berukuran 2x2 dan diproses dengan RES Block yang menghasilkan feature map berukuran 112x112x64.
6. Pada blok ketiga diproses dengan 1 lapisan transpose berukuran 2x2 dan diproses dengan RES Block yang menghasilkan feature map berukuran 224x224x3.
7. Semua lapisan konvolusi dan transpose *Modification Lu-Net* diikuti oleh fungsi aktivasi ReLU dengan rumus seperti pada persamaan 3.4.

$$f_{Relu}(h_{i,k}) = \max(0, h_{i,k}) \tag{3.4}$$

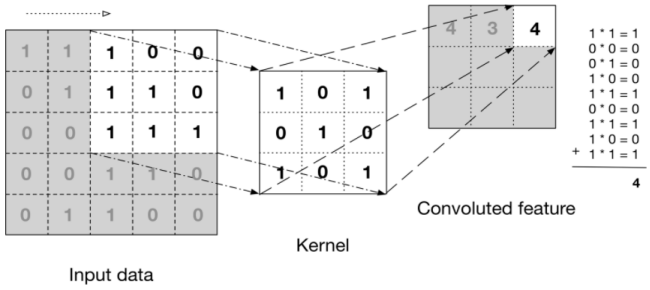
8. Setelah proses decoder selesai, feature map hasil decoder masuk ke fully connected layer dan diproses dengan konvolusi 1x1.

9. Pada proses fully connected menggunakan fungsi aktivasi sigmoid. Fungsi aktivasi sigmoid berguna untuk menentukan hasil deteksi tumor dan menghasilkan output layer.

Penggunaan tiap layer dalam algoritma memiliki peranan penting, dimana tiap layer yang berbeda memiliki fungsi tersendiri. Penjelasan pada tiap layer akan dijelaskan pada paragraf berikut.

a. Convolutional layer

Convolutional layer terdiri dari satu set kernel convolutional di mana setiap neuron bertindak sebagai kernel. Kernel convolutional bekerja dengan membagi gambar menjadi irisan kecil seperti pada gambar 3.9.



Gambar 3.9 Proses Convolutional Layer

Pembagian gambar menjadi blok-blok kecil membantu dalam mengekstraksi motif fitur. Kernel bergabung dengan gambar menggunakan serangkaian bobot tertentu dengan mengalikan elemennya dengan elemen yang sesuai dari bidang reseptif.

$$f_l^k(p,q) = \sum_c \sum_{x,y} i_c(x,y).\, e_l^k\,(u,v) \qquad (3.5)$$

di mana, i_c (x , y) adalah elemen tensor citra input i_c, yang merupakan elemen wise dikalikan dengan $e_l^k(u,v)$ indeks dari k^{th} kernel konvolusi k_l dari

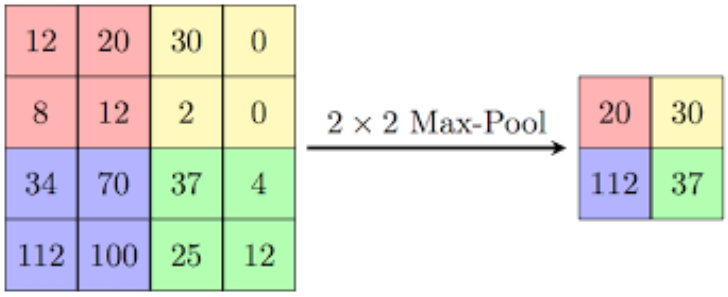
layer l^{th}. Sedangkan fitur map keluaran dari k^{th} operasi konvolusi dapat dinyatakan sebagai berikut (Khan *et al*., 2020)

$$\mathbf{F}_l^k = [f_l^k(1,1), \ldots\ldots, f_l^k(p,q), \ldots\ldots, f_l^k(P,Q)] \tag{3.6}$$

Karena kemampuan berbagi bobot dari operasi konvolusi, kumpulan fitur yang berbeda dalam suatu gambar dapat diekstraksi dengan menggeser kernel dengan kumpulan bobot yang sama pada gambar dan dengan demikian membuat parameter CNN lebih efisien dibandingkan dengan jaringan yang sepenuhnya terhubung.

b. Pooling layer

Feature Map yang dihasilkan sebagai output dari operasi konvolusi, dapat terjadi di lokasi yang berbeda dalam gambar. Setelah fitur diekstraksi, lokasi persisnya menjadi kurang penting selama perkiraan posisinya relatif terhadap yang lain dipertahankan. Pooling atau down-sampling adalah operasi lokal yang menarik. Ini merangkum informasi serupa di sekitar bidang reseptif dan menampilkan respons dominan di wilayah lokal ini.



Gambar 3.10 Max Pooling

$$\mathbf{Z}_l^k = g_p\,(\boldsymbol{F}_l^k) \tag{3.7}$$

Persamaan di atas menunjukkan operasi pooling di mana $\mathbf{Z}_l^k$ merupakan pooled fitur-map lapisan l^{th} untuk input k fitur-map $\boldsymbol{F}_l^k$, sedangkan g_p (.) mendefinisikan jenis operasi pooling. Penggunaan operasi pooling membantu untuk mengekstrak kombinasi fitur, yang invarian untuk pergeseran translasi dan distorsi kecil. Pengurangan ukuran fitur map ke set fitur invarian tidak hanya mengatur kompleksitas jaringan tetapi juga membantu dalam meningkatkan generalisasi dengan mengurangi overfitting. Max-Pooling diproses seperti pada gambar 3.10.

c. RES Block

Residual Extended Skip (RES) berasal dari Residual Networks (ResNets), merupakan salah satu variant dari skip connection. ResNets diusulkan oleh He *et al.* (2015), untuk memecahkan masalah klasifikasi citra. Dalam ResNets, informasi dari lapisan awal diteruskan ke lapisan yang lebih dalam dengan penambahan matriks. Operasi ini tidak memiliki parameter tambahan sebagai output dari lapisan sebelumnya. Representasi lapisan yang lebih dalam dari ResNets menghasilkan bobot yang telah dilatih sebelumnya dari jaringan ini dapat digunakan untuk menyelesaikan banyak kasus dan tidak hanya terbatas pada klasifikasi gambar tetapi juga dapat memecahkan berbagai masalah pada segmentasi gambar, deteksi keypoint & deteksi objek. Oleh karena itu, ResNet adalah salah satu arsitektur paling baik. Rumus dasar RES Block adalah sebagai berikut.

$$x_i = F(x_i, W_i),\ x_{i+1} = x_i + F(x_i, W_i) \tag{3.8}$$

Dimana x_i adalah input dan x_{i+1} adalah output dari jaringan. F menunjukkan fungsi residual dan W_i adalah parameter block. Adapun penerapan RES Block dalam program bisa dilihat pada gambar 3.11.

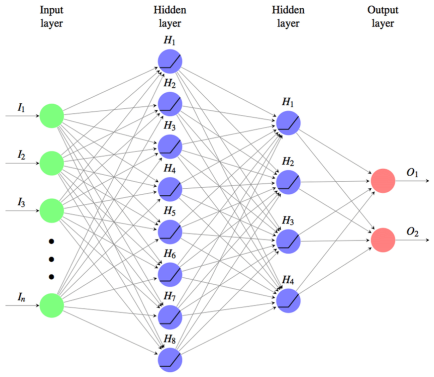
```
def resblock(X, f):
    X_copy = X  #copy of input

    # main path
    X = Conv2D(f, kernel_size=(1,1),
       kernel_initializer='he_normal')(X)
    X = BatchNormalization()(X)
    X = Activation('relu')(X)

    X = Conv2D(f, kernel_size=(3,3), padding='same',
kernel_initializer='he_normal')(X)
    X = BatchNormalization()(X)

    # shortcut path
    X_copy = Conv2D(f, kernel_size=(1,1),
kernel_initializer='he_normal')(X_copy)
    X_copy = BatchNormalization()(X_copy)

    # Adding the output from main path and short path together
    X = Add()([X, X_copy])
    X = Activation('relu')(X)

    return X

def upsample_concat(x, skip):
    X = UpSampling2D((2,2))(x)
    merge = Concatenate()([X, skip])

    return merge
```

Gambar 3.11 Kode Program RES Block

d. Fully Connected Layer

Fully Connected Layer merupakan salah satu fungsi dalam deteksi citra, dalam tahap ini setiap neuron di tingkat atas saling berhubungan dengan setiap neuron di tingkat berikutnya. Fully Connected Layer menerima output dari lapisan atas (yang mewakili peta fitur dari fitur tingkat yang lebih tinggi) dan menentukan output yang paling cocok. Gambaran proses fully connected

layer terlihat dalam gambar 3.12, dalam proses ini menggunakan fungsi aktivasi ReLU dan Sigmoid.



Gambar 3.12 Proses Fully Connected Layer

e. Activation Function

Fungsi aktivasi berfungsi sebagai fungsi keputusan dan membantu dalam mempelajari pola yang rumit. Pemilihan fungsi aktivasi yang tepat dapat mempercepat proses pembelajaran. Fungsi aktivasi untuk fitur map didefinisikan dalam persamaan dibawah.

$$\mathbf{T}_l^k = g_a\left(\boldsymbol{F}_l^k\right) \tag{3.9}$$

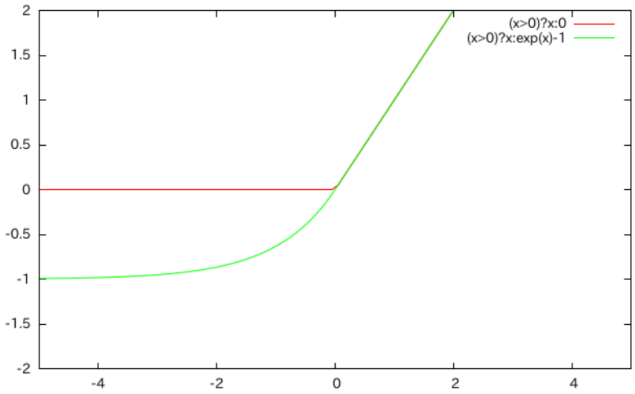
Dalam persamaan di atas, $\boldsymbol{F}_l^k$ adalah output dari konvolusi, yang ditetapkan ke fungsi aktivasi g_a yang menambahkan non-linearitas dan mengembalikan output $\boldsymbol{T}_l^k$ yang ditransformasikan untuk l^{th} lapisan. Dalam literatur, fungsi aktivasi yang berbeda seperti sigmoid, tanh, maxout, SWISH, dan ReLU digunakan untuk menanamkan kombinasi fitur non-linear. Namun, ReLU dan variannya lebih banyak digunakan karena membantu mengatasi masalah gradien yang hilang.

a) ReLu

Diketahui bahwa fungsi aktivasi standar seperti fungsi sigmoid atau fungsi hiperbolik tangen bersifat kontraktif hampir di semua tempat dan gradien pada nilai besar menjadi hampir nol. Dengan demikian pembaruan oleh gradien stokastik yang layak menjadi sangat kecil. Masalah ini dikenal sebagai masalah gradien hilang. Brown *et al.* (2017) memperkenalkan unit linier yang diperbaiki (ReLU) yang didefinisikan sebagai

$$f_{Relu}(h_{i,k}) = \max(0, h_{i,k}) \tag{3.10}$$

Grafik unit liner yang diperbaiki ditunjukkan pada Gambar 3.11 dengan warna merah.



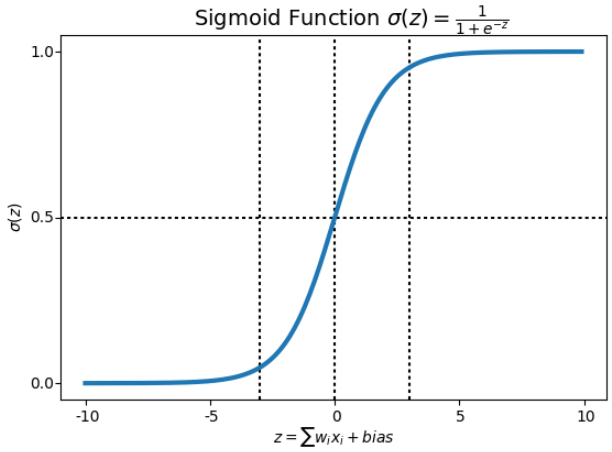
Gambar 3.13 Grafik fungsi aktivasi ReLU

b) Sigmoid

Fungsi sigmoid adalah bentuk khusus dari fungsi logistik dan biasanya dilambangkan dengan (x) atau sig(x) seperti pda persamaan 3.11.

$$x = \frac{1}{1+exp^{-x}} \tag{3.11}$$

Fungsi sigmoid juga disebut fungsi squashing karena domainnya adalah himpunan semua bilangan real, dan jangkauannya adalah (0, 1). Oleh karena itu, jika input ke fungsi tersebut adalah bilangan negatif yang sangat besar atau bilangan positif yang sangat besar, keluarannya selalu antara 0 dan 1. Hal yang sama berlaku untuk bilangan apa pun antara -∞ dan +∞. Ketika fungsi aktivasi untuk neuron adalah fungsi sigmoid, itu adalah jaminan bahwa output unit ini akan selalu antara 0 dan 1.



Gambar 3.14 Grafik Fungsi Sigmoid

Sigmoid adalah fungsi non-linier, output dari unit ini akan menjadi non-linier. Neuron seperti itu yang menggunakan fungsi sigmoid sebagai fungsi aktivasi disebut sebagai unit sigmoid.

3.2.5 Proses Training

Proses training dilakukan untuk melatih sistem deteksi tumor otak menggunakan algoritma *Modification Lu-Net* dan nantinya sistem mampu mengenali objek dengan menghasilkan nilai akurasi yang tinggi. Pada proses training sistem dilatih menggunakan data training sebesar 80% dari data

keseluruhan. Tujuan melatih algoritma *Modification Lu-Net* adalah untuk membuat algoritma mampu menemukan ciri pada citra yang diinputkan dan mengaktifkan neuron mana saja yang diaktifkan untuk mendeteksi citra.

Dalam proses training sistem menggunakan dua model arsitektur yaitu model arsitektur *Modification Lu-Net* dan model arsitektur *Lu-Net* yang dikembangkan oleh Rai & Chatterjee (2020), dimana nantinya kedua model ini akan dibandingkan hasilnya. Adapun kode program pada proses training dapat dilihat pada gambar 3.15.

```
model.compile(optimizer='adam',
              loss="binary_crossentropy",
              metrics=[["accuracy"], tp,tn,fp, fn, recall, precision,
               specificity])

callbacks = [ModelCheckpoint(str(k+1) + '_Modification_Lunet.hdf5',
             verbose=1, save_best_only=True)]
history   = model.fit(train_gen,
            steps_per_epoch=len(train_data_frame) / BATCH_SIZE,
            epochs=EPOCHS,
            callbacks=callbacks,
            validation_data = test_gener,
            validation_steps=len(test_data_frame) / BATCH_SIZE)
```

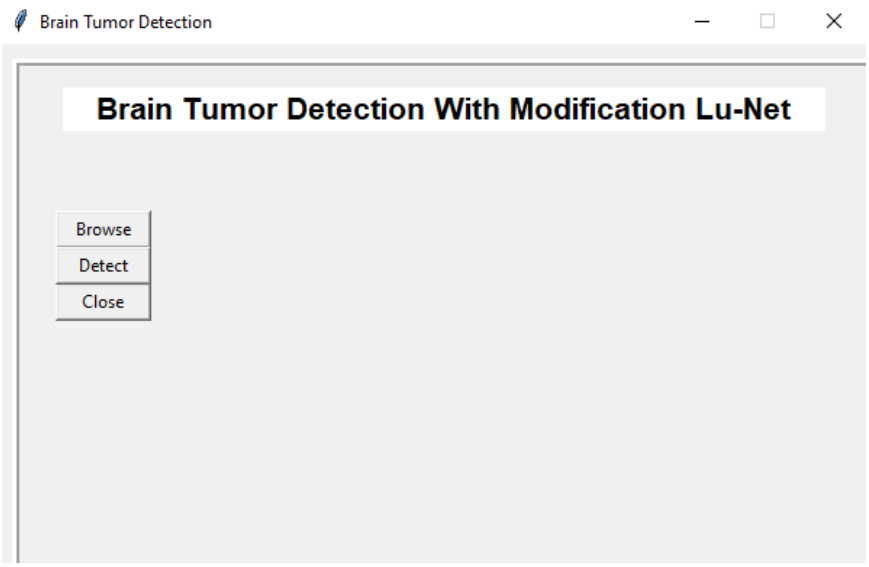
Gambar 3.15. Kode Program *Training*

Parameter yang diberikan pada proses training meliputi iterasi dengan nilai 100 epoch sehingga proses training akan diulang sebanyak 100 kali, kemudian learning rate diberikan parameter sebesar 0,001 yang berfungsi untuk menghitung nilai koreksi bobot pada waktu proses training yang apabila nilainya semakin kecil dengan range 0 sampai 1 maka ketelitian jaringan akan semakin besar atau bertambah namun proses training akan menjadi semakin lama. Selanjutnya

parameter batch size diberikan nilai sebesar 32 yang berarti jumlah sampel data yang dikerjakan sebanyak kelipatan 32.

3.2.6 *Graphical User Interface* Modification Lu-Net untuk Deteksi Tumor Otak Berbasis Citra MRI

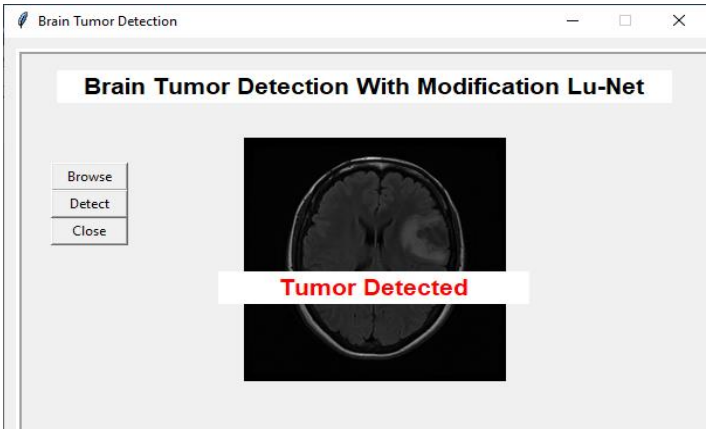
Graphical User Interface (GUI) merupakan suatu tampilan grafis yang berfungsi untuk memudahkan user dalam menggunakan aplikasi yang dibuat peneliti. GUI yang disediakan oleh peneliti dalam penelitian ini menggunakan bahasa pemrograman berbasis Python3, dimana ketika program dijalankan maka akan muncul sebuah tampilan awal seperti pada gambar 3.16.



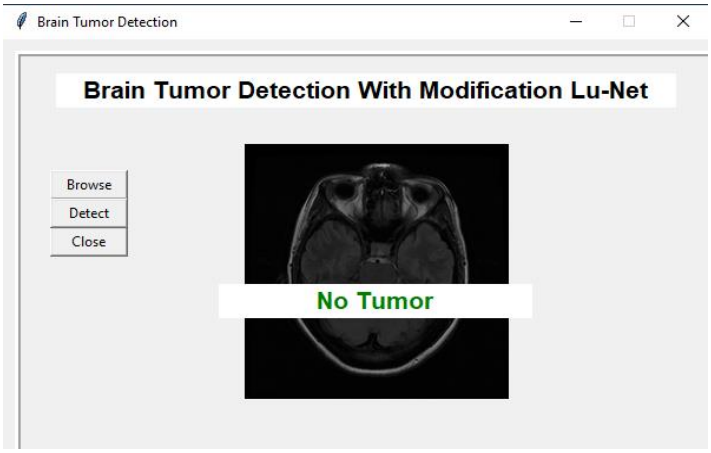
Gambar 3.16 GUI Awal Modification Lu-Net untuk Deteksi Tumor Otak Berbasis Citra MRI

Untuk mendeteksi sebuah citra MRI positive tumor atau negative tumor maka terlebih dahulu user di instruksikan agar mengunggah sebuah citra MRI otak dengan klik tombol browse, kemudian pilih citra MRI yang akan di deteksi dengan sistem kemudian klik open dan akan muncul citra MRI yang dipilih. dan kemudian klik detect, hasilnya sistem akan secara otomatis memberikan hasil

apakah citra MRI yang diunggah merupakan positif tumor atau negative tumor. Adapun tampilan ketika tumor terdeteksi akan muncul seperti pada gambar 3.15. dan ketika negatif tumor maka akan muncul tampilan seperti pada gambar 3.16.



Gambar 3.17. Hasil GUI Positif Tumor



Gambar 3.18. Hasil GUI Negatif Tumor

BAB IV

UJI COBA DAN PEMBAHASAN

Dalam bab ini akan menjelaskan tentang uji coba dan pembahasan Deteksi Tumor Otak Menggunakan Algoritma *Modification Lu-Net* Berbasis Citra MRI dengan percobaan model arsitektur *Modification Lu-Net* dan model arsitektur *Lu-Net* yang dikembangkan oleh Rai & Chatterjee (2020).

4.1 Skenario Uji Coba

Dalam pengujian penelitian ini memiliki beberapa tahapan. Tahapan pertama data yang telah disiapkan diuji dengan menggunakan model arsitektur *Modification Lu-Net*, kemudian pengujian juga dilakuakan dengan menggunakan model arsitektur *Lu-Net* yang dikembangkan oleh Rai & Chatterjee (2020). Pada tahap kedua pengujian dilakukan dengan mengubah jumlah data training yang digunakan, penggunaan learning rate yang berbeda, dan optimizer yang berbeda.

Penelitian ini menggunakan sebanyak 3929 data citra MRI yang setiap citra memiliki mask sebagai *ground truth* atau lokasi objek tumor yang benar, dimana sebanyak 2556 citra MRI yang memiliki tumor otak dan 1373 citra MRI otak normal.

Adapun skenario uji coba pada tahap pertama dalam penelitian ini adalah sebagai berikut:

a. Dataset yang telah disiapkan selanjutnya akan masuk ke *preprocessing* citra, dalam tahap ini citra disamakan ukuran pikselnya dengan ukuran 224x224x3. Kemudian setelah citra selesai di tahap preprocessing citra maka selanjutnya

citra di proses dengan augmentasi citra, proses ini berguna untuk memperbanyak dataset. Selanjutnya dataset dibagi menjadi 3 bagian, dimana 80% data *training*, 10% data *validation*, dan 10% data *testing.*

b. Setelah dataset dibagi selanjutnya akan masuk dalam proses *training*, dalam proses ini algoritma *Lu-Net* dilatih untuk bisa mendeteksi objek tumor pada sebuah citra MRI. Training dilakukan menggunaka model arsitektur *Modification Lu-Net* dan di model *Lu-Net* yang dikembangkan oleh Rai & Chatterjee (2020) dengan epoch sebanyak 100.

c. Proses testing. Dalam proses *testing* algoritma yang telah melalui proses training di uji menggunakan data *testing* yang telah melewati *preprocessing* dan *augmentasi* data kemudian di uji dengan model yang telah melewati *training* untuk melihat hasil akurasi algoritma.

Sedangkan skenario uji coba pada tahap kedua dalam penelitian ini adalah sebagai berikut:

a. Percobaan dilakukan sebanyak 5 kali, dimana dari dataset yang telah melewati proses *preprocessing* dan *augmentasi* data diambil sebanyak 20% untuk data *testing* dan 80% untuk data *training*. Kemudia dari data training dibagi menjadi 5 dan dilakukan 5x percobaan. Hal ini dilakukan untuk mengetahui pengaruh jumlah data *training* terhadap akurasi hasil deteksi. Pada tahap ini menggunakan algoritma *Modification Lu-Net.* Setelah proses *training* selesai maka model akan diuji dengan data *testing* yang sama pada tiap percobaan. Pembagian data pada tahap ini dapat dilihat pada tabel 4.1.

Tabel 4.1 Pembagian Dataset Training dan Testing

No	Jumlah Data Training	Jumlah Data Testing
1	629 Citra MRI (20%)	785 Citra MRI
2	1258 Citra MRI (40%)	785 Citra MRI
3	1887 Citra MRI (60%)	785 Citra MRI
4	2516 Citra MRI (80%)	785 Citra MRI
5	3144 Citra MRI (100%)	785 Citra MRI

b. Selanjutnya percobaan dilakukan dengan menggunakan beberapa learning rate yang berbeda, dimana learning rate yang digunakan adalah 0.01, 0.05, 0.1, 0.15, dan 0.2. Tahap uji coba ini bertujuan untuk mengetehui pengaruh learning rate terhadap hasil akurasi. Pada tahap ini menggunakan algoritma *Modification Lu-Net.* Sedangkan untuk penggunaan dataset dibagi menjadi 3 bagian, dimana 80% data *training,* 10% data *validation,* dan 10% data *testing.*
c. Pengujian juga dilakukan dengan mengunakan optimizer yang berbeda. *Optimizer Adam, optimizer RMSprop, dan optimizer SGD*. Hal ini bertujuan untuk mengetahui pengaruh *optimizer* terhadap hasil akurasi. Penggunaan algoritma dan pembagian dataset yang digunakan sama denngan percobaan sebelumnya.

Dalam pengujian penelitian menggunakan *Kaggle Notebook* yang merupakan salah satu *tools* yang disediakan *Kaggle* berbasis bahasa pemrograman *python* dengan penyimpanan menggunakan *Kaggle Dataset*.

Parameter dalam penelitian ini menggunakan lima matriks perhitungan, antara lain adalah *Recall, Precision, F-score, Specificity, dan Accuracy*, penggunaan dalam program terdapat pada gambar 4.1, sedangkan untuk rumus tiap matriks terdapat pada persamaan 4.1 sampai persamaan 4.5. Perhitungan

matriks di atas menggunakan 4 parameter dasar yaitu adalah True Positive (TP), True Negative (TN), False Positive (FP) dan False Negative (FN).

```
def tp(ytrue, ypred):
    tp = K.sum(K.round(K.clip(ytrue * ypred, 0, 1)))
    tp = tp + K.epsilon()
    return tp

def tn(ytrue, ypred):
    tn = K.sum(K.round(K.clip((1 - ytrue) * (1 - ypred), 0, 1)))
    tn = tn + K.epsilon()
    return tn

def fp(ytrue, ypred):
    fp = K.sum(K.round(K.clip((1 - ytrue) * ypred, 0, 1)))
    fp = fp + K.epsilon()
    return fp

def fn(ytrue, ypred):
    fn = K.sum(K.round(K.clip(ytrue * (1 - ypred), 0, 1)))
    fn = fn + K.epsilon()
    return fn

def recall(ytrue, ypred):
    tp = K.sum(K.round(K.clip(ytrue * ypred, 0, 1)))
    fn = K.sum(K.round(K.clip(ytrue * (1 - ypred), 0, 1)))
    recall_keras = tp / (tp + fn + K.epsilon())
    return recall_keras

def precision(ytrue, ypred):
    tp = K.sum(K.round(K.clip(ytrue * ypred, 0, 1)))
    fp = K.sum(K.round(K.clip((1 - ytrue) * ypred, 0, 1)))
    precision_keras = tp / (tp + fp + K.epsilon())
    return precision_keras

def specificity(ytrue, ypred):
    tn = K.sum(K.round(K.clip((1 - ytrue) * (1 - ypred), 0, 1)))
    fp = K.sum(K.round(K.clip((1 - ytrue) * ypred, 0, 1)))
    return tn / (tn + fp + K.epsilon())
```

Gambar 4.1 Kode Program Matriks

$$Recall\ (Re) = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{4.1}$$

$$Precision(\mathrm{Pr}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \tag{4.2}$$

$$Specifity\ (Sp) = \frac{\mathrm{TN}}{\mathrm{TN+FP}} \tag{4.3}$$

$$Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} \quad (4.4)$$

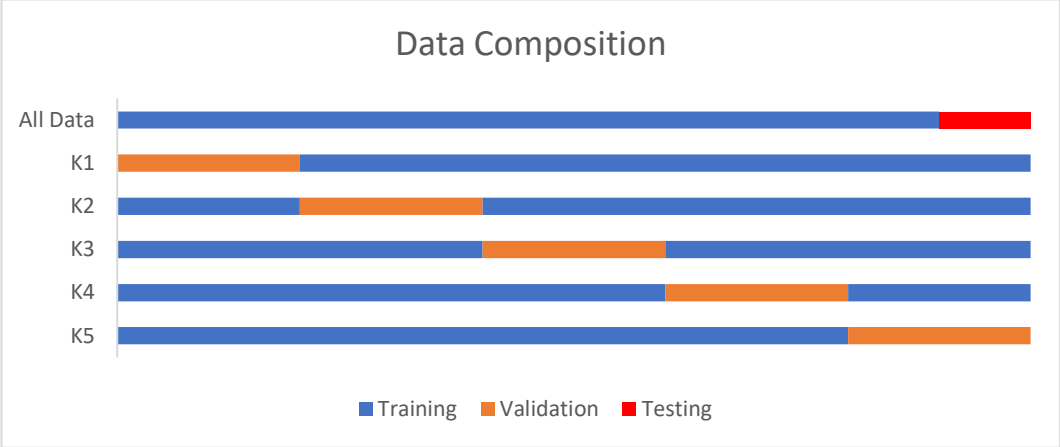
$$F - Score = \frac{2 \times Pr \times Re}{Pr+Re} \quad (4.5)$$

- True Positive (TP): Citra dengan output positif tumor otak, ground truth diprediksi oleh sistem positif tumor otak.
- True Negative (TN): Citra dengan output non tumor otak, ground truth diprediksi oleh sistem non tumor otak.
- False Positive (FP) : Citra dengan output negatif tumor otak, ground truth diprediksi oleh sistem positif tumor otak.
- False Negative (FN): Citra dengan output positif tumor otak, ground truth diprediksi oleh sistem negative tumor otak.

Dalam pengujian suatu model CNN terdapat beberapa metode pengujian, dalam penelitian ini peneliti menggunakan metode *K-Fold Cross Validation. K-Fold Cross Validation* adalah prosedur pengambilan sampel ulang yang digunakan untuk mengevaluasi model pada sampel data terbatas.

Prosedur *K-Fold Cross Validation* memiliki parameter tunggal yang disebut k yang mengacu pada jumlah grup yang akan dibagi menjadi sampel data tertentu. Prosedur ini disebut *k-fold cross-validation*. Ketika nilai spesifik untuk k dipilih, nilai tersebut dapat digunakan sebagai pengganti k dalam referensi model, seperti k=5 menjadi *Cross Validation* 5 kali lipat.

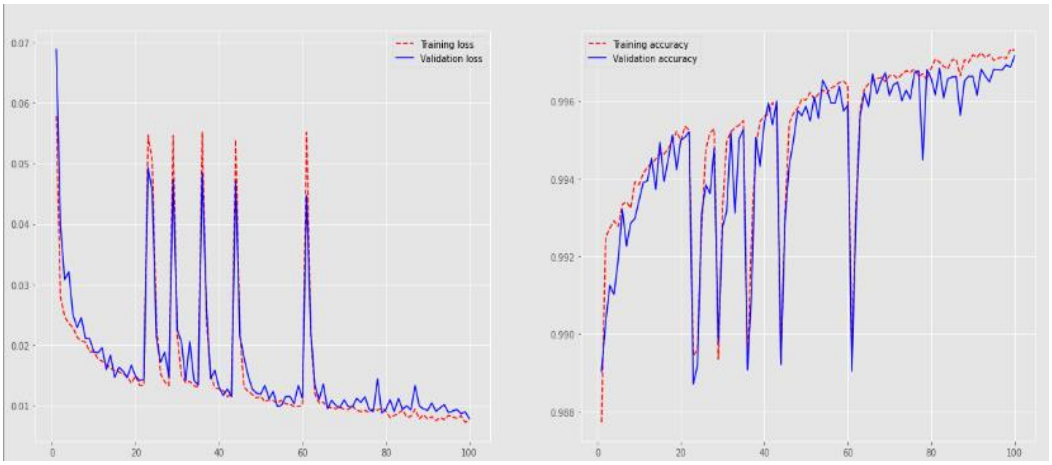
Dalam penelitian ini data yang digunakan dibagi menjadi 10% data *testing*, kemudian sisa data dibagi menjadi training dan validation dengan k=5. Adapun skema pembagian data dapat dilihat pada gambar 4.2.



Gambar 4.2 Komposisi Data

4.2 Hasil *Training* Arsitektur *Modification Lu-Net*

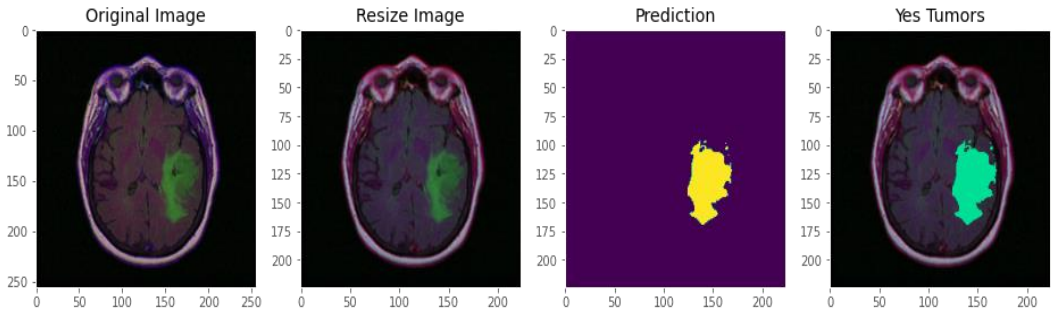
Berdasarkan scenario uji coba dalam sub bab 4.1. dengan menggunakan model arsitektur *Modification Lu-Net* yang menggunakan data training sebanyak 3182, data validasi sebanyak 354, dan data testing sebanyak 393. Pada *training* model *Modification Lu-Net* menghasilkan loss sebesar 0,1090 pada epoch pertama dan nilai loss terus menurun sampai ke epoch ke 100 yang memiliki nilai loss 0,0082. Grafik loss dan accuracy pada proses *training* dengan model arsitektur *Modification Lu-net* dapat dilihat pada gambar 4.3.



Gambar 4.3 Grafik Training Model Arsitektur *Modification Lu-Net*

4.3 Hasil *Testing* Arsitektur *Modification Lu-Net*

Selanjutnya pada tahap ini model yang telah melalui tahap *training* di uji dengan *testing.* Pada tahap testing citra di inputkan kemudian masuk pada tahap *preprocessing* dimana citra akan disesuaikan ukuran pikselnya dengan ukuran 224x224x3, selanjutnya citra diuji dengan menggunakan model yang telah disimpan dan menghasilkan prediksi. Adapun hasil dari tiap prosesnya dapat dilihat pada gambar 4.4 yang merupakan salah satu contoh hasil testing dari total 393 data testing.



Gambar 4.4 Hasil *Testing* Model Arsitektur *Modification Lu-Net*

Pada gambar pertama merupakan citra asli, kemudian pada gambar kedua merupakan citra yang telah disesuaikan ukurannya, pada gambar ketiga merupakan hasil deteksi tumor otak, dan pada gambar keempat merupakan gambar MRI yang digabungkan dengan hasil deteksi sehingga dapat dilihat lokasi tumor otak pada citra MRI. Sedangkan untuk nilai *confusion matrix* percobaan ini mendapatkan hasil sebagai berikut:

1. True Positive = 15188
2. True Negative = 1496283
3. False Positif = 3514
4. False Negative = 1873

Nilai *confusion matrix* di atas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut:

a. $Recall\ (Re) = \frac{\text{TP}}{\text{TP+FN}} x\ 100\% = 89{,}45\%$

b. $Precision(\text{Pr}) = \frac{\text{TP}}{\text{TP+FP}}\ x\ 100\% = 80{,}11\%$

c. $Specificity\ (Sp) = \frac{\text{TN}}{\text{TN+FP}}\ x\ 100\%\ =\ 99{,}77\%$

d. $Accuracy\ (Acc) = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} x100\% = 99{,}64\ \%$

e. $F - Score = \frac{2 \times Pr\ \times Re}{Pr+Re}\ x\ 100\% =\ 84{,}52\%$

Berdasarkan percobaan menggunakan model algoritma *Modification Lu-Net* menghasilkan nilai *confusion matrix* pada *Recall* sebesar 89,45%, *Precission* sebesar 80,11%, *Specificity* sebesar 99,77%, *Accuracy* sebesar 99,64% dan *F-Score* sebesar 84,52%.

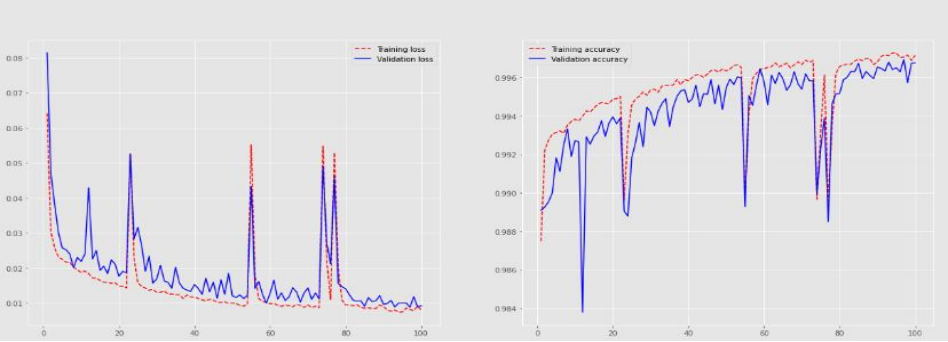
4.4 Hasil *Training K-Fold Cross Validation Modification Lu-Net*

Tahap ini menguji model dengan pengujian dataset yang berbeda, dimana dataset dibagi ke dalam 5 bagian dan tiap bagian digunakan untuk pengujian model *Modification Lu-Net.* Dalam tiap pengujian model Arsitektur yang digunakan selalu sama sehingga tiap Fold mendapatkan hasil sebagai berikut.

1. Fold 1 Training *Modification Lu-Net*

   Proses training Fold ke 1 dengan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0,0641dan pada epoch terakhir nilai loss menurun menjadi 0.0081. Grafik loss dan accuracy

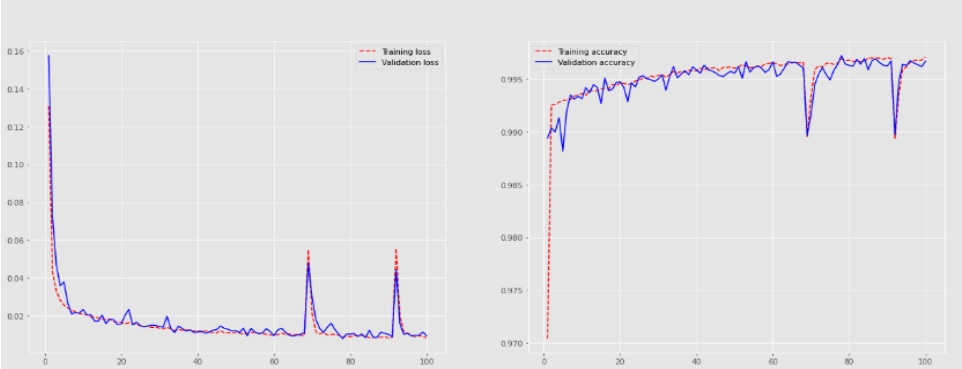
pada training dengan model arsitektur *Modification Lu-net* dapat dilihat pada gambar 4.5.



Gambar 4.5 Grafik Training Model Arsitektur *Modification Lu-Net* Fold 1

2. Fold 2 Training *Modification Lu-Net*

Proses training Fold ke 2 dengan menggunakan arsitektur *Modification Lu-Net*, pada epoch pertama mendapatkan nilai loss sebesar 0,1308dan pada epoch terakhir nilai loss menurun menjadi 0,0084. Grafik loss pada training dengan model arsitektur *Modification Lu-net* dapat dilihat pada gambar 4.6.

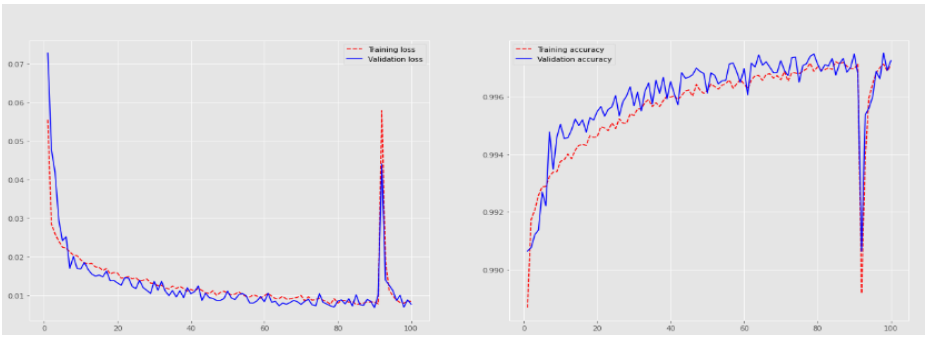


Gambar 4.6 Grafik Training Model Arsitektur *Modification Lu-Net* Fold 2

3. Fold 3 Training *Modification Lu-Net*

Proses training Fold ke 3 dengan menggunakan arsitektur *Modification Lu-Net*, pada epoch pertama mendapatkan nilai loss sebesar 0,0554 dan pada

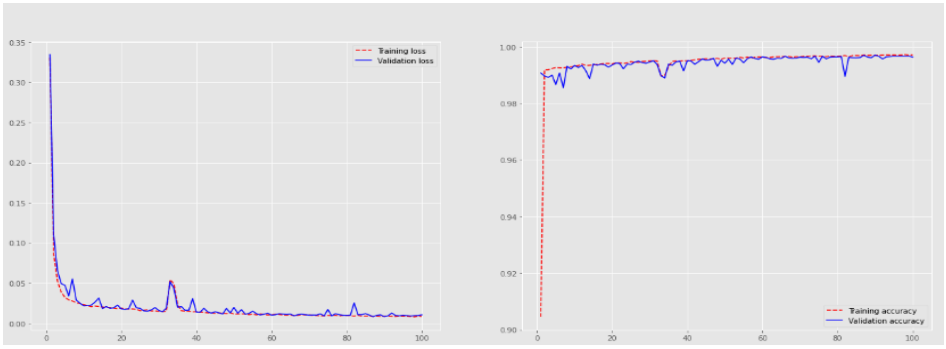
epoch terakhir nilai loss menurun menjadi 0,0084. Grafik loss pada training dengan model arsitektur *Modification Lu-net* dapat dilihat pada gambar 4.7.



Gambar 4.7 Grafik Training Model Arsitektur *Modification Lu-Net* Fold 3

4. Fold 4 Training *Modification Lu-Net*

Proses training Fold ke 4 dengan menggunakan arsitektur *Modification Lu-Net*, pada epoch pertama mendapatkan nilai loss sebesar 0,3309 dan pada epoch terakhir nilai loss menurun menjadi 0,0083. Grafik loss pada training dengan model arsitektur *Modification Lu-net* dapat dilihat pada gambar 4.8.

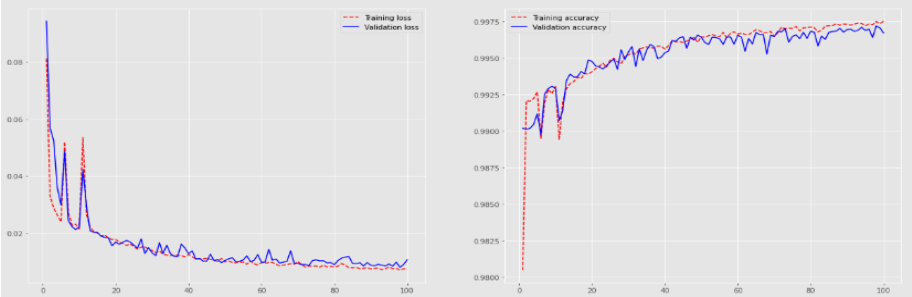


Gambar 4.8 Grafik Training Model Arsitektur *Modification Lu-Net* Fold 4

5. Fold 5 Training *Modification Lu-Net*

Proses training Fold ke 5 dengan menggunakan arsitektur *Modification Lu-Net*, pada epoch pertama mendapatkan nilai loss sebesar 0,0811dan pada

epoch terakhir nilai loss menurun menjadi 0,0072. Grafik loss pada training dengan model arsitektur *Modification Lu-net* dapat dilihat pada gambar 4.9.



Gambar 4.9 Grafik Training Model Arsitektur *Modification Lu-Net* Fold 5

4.5 Hasil *Testing K-Fold Cross Validation Modification Lu-Net*

Pada subbab ini menjelaskan tentang proses *testing* dari pengujian menggunakan *K-Fold Cross Validation* dimana pada tiap fold diuji dengan data testing yang sama sejumlah 393 data citra MRI. Adapun *testing* tiap fold mendapatkan hasil sebagai berikut.

1. Fold 1 *Testing Modification Lu-Net*

Proses *testing* Fold ke 1 dengan menggunakan arsitektur *Modification Lu-Net* mendapatkan nilai hasil pengujian *confusion matrix* sebagai berikut:

a. True Positive = 14288
b. True Negative = 1496804
c. False Positif = 2105
d. False Negative = 3661

Nilai *confusion matrix* di atas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut:

- $Recall\ (Re) = \frac{\text{TP}}{\text{TP+FN}} x\ 100\% = 79{,}27\%$

- $Precision(\mathrm{Pr}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}}\ x\ 100\% = 85{,}99\%$
- $Specifity\ (Sp) = \frac{\mathrm{TN}}{\mathrm{TN+FP}}\ x\ 100\%\ =\ 99{,}84\%$
- $Accuracy\ (Acc) = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}} x100\% = 99{,}62\%$
- $F - Score = \frac{2 \times Pr \times Re}{Pr+Re}\ x\ 100\% =\ 84{,}86\%$

Berdasarkan percobaan menggunakan model algoritma *Modification Lu-Net* pada fold 1, menghasilkan nilai *confusion matrix* pada *Recall* sebesar 79,27%, *Precision* sebesar 85,99%, *Specificity* sebesar 99,84%, *Accuracy* sebesar 99,62% dan *F-Score* sebesar 82,49%.

2. Fold 2 *Testing Modification Lu-Net*

Proses *testing* Fold ke 2 dengan menggunakan arsitektur *Modification Lu-Net* mendapatkan nilai hasil pengujian *confusion matrix* sebagai berikut:

a. True Positive = 14256
b. True Negative = 1497058
c. False Positif = 1851
d. False Negative = 3692

Nilai *confusion matrix* di atas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut:

- $Recall\ (Re) = \frac{\mathrm{TP}}{\mathrm{TP+FN}} x\ 100\% = 79{,}24\%$
- $Precision(\mathrm{Pr}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{170189}{170189+25734}\ x\ 100\% = 87{,}99\%$
- $Specifity\ (Sp) = \frac{\mathrm{TN}}{\mathrm{TN+FP}}\ x\ 100\%\ =\ 99{,}86\%$

- $Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} x100\% = 99{,}63\%$
- $F - Score = \frac{2 \times Pr \times Re}{Pr+Re}\ x\ 100\% =\ 83{,}39\%$

Berdasarkan percobaan menggunakan model algoritma *Modification Lu-Net* pada fold 2, menghasilkan nilai *confusion matrix* pada *Recall* sebesar 79,24%, *Precision* sebesar 87,99%, *Specificity* sebesar 99,86%, *Accuracy* sebesar 99,63% dan *F-Score* sebesar 83,39%.

3. Fold 3 *Testing Modification Lu-Net*

Proses *testing* Fold ke 3 dengan menggunakan arsitektur *Modification Lu-Net* mendapatkan nilai hasil pengujian *confusion matrix* sebagai berikut:

a. True Positive = 15368
b. True Negative = 1496106
c. False Positif = 2803
d. False Negative = 2581

Nilai *confusion matrix* di atas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut:

- $Recall\ (Re) = \frac{\mathrm{TP}}{\mathrm{TP+FN}} x\ 100\% = 85{,}43\%$
- $Precision(\mathrm{Pr}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{170189}{170189+25734}\ x\ 100\% = 83{,}73\%$
- $Specifity\ (Sp) = \frac{\mathrm{TN}}{\mathrm{TN+FP}}\ x\ 100\%\ =\ 99{,}80\%$
- $Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} x100\% =\ 99{,}65\%$
- $F - Score = \frac{2 \times Pr \times Re}{Pr+Re}\ x\ 100\% =\ 84{,}57\%$

Berdasarkan percobaan menggunakan model algoritma *Modification Lu-Net* pada fold 3, menghasilkan nilai *confusion matrix* pada *Recall* sebesar 85,43%, *Precision* sebesar 83,73%, *Specificity* sebesar 99,80%, *Accuracy* sebesar 99,65% dan *F-Score* sebesar 84,57%.

4. Fold 4 *Testing Modification Lu-Net*

Proses *testing* Fold ke 4 dengan menggunakan arsitektur *Modification Lu-Net* mendapatkan nilai hasil pengujian *confusion matrix* sebagai berikut:

a. True Positive = 14895
b. True Negative = 1496432
c. False Positif = 2477
d. False Negative = 3053

Nilai *confusion matrix* di atas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut:

- $Recall\ (Re) = \frac{\mathrm{TP}}{\mathrm{TP+FN}} x\ 100\% = 82{,}79\%$
- $Precision(\mathrm{Pr}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}}\ x\ 100\% = 85{,}22\%$
- $Specifity\ (Sp) = \frac{\mathrm{TN}}{\mathrm{TN+FP}}\ x\ 100\%\ =\ 99{,}82\%$
- $Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} x100\% =\ 99{,}64\%$
- $F - Score = \frac{2 \times Pr\ \times Re}{Pr+Re}\ x\ 100\% =\ \ 83{,}99\%$

Berdasarkan percobaan menggunakan model algoritma *Modification Lu-Net* pada fold 4, menghasilkan nilai *confusion matrix* pada *Recall* sebesar

82,79%, *Precision* sebesar 85,22%, *Specificity* sebesar 99,82%, *Accuracy* sebesar 99,64% dan *F-Score* sebesar 83,99%.

5. Fold 5 *Testing Modification Lu-Net*

Proses *testing* Fold ke 5 dengan menggunakan arsitektur *Modification Lu-Net* mendapatkan nilai hasil pengujian *confusion matrix* sebagai berikut:

a. True Positive = 14820
b. True Negative = 1496859
c. False Positif = 2050
d. False Negative = 3128

Nilai *confusion matrix* di atas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut:

- $Recall\ (Re) = \frac{\mathrm{TP}}{\mathrm{TP+FN}} x\ 100\% = 83{,}09\%$
- $Precision(\mathrm{Pr}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{170189}{170189+25734}\ x\ 100\% = 87{,}23\%$
- $Specifity\ (Sp) = \frac{\mathrm{TN}}{\mathrm{TN+FP}}\ x\ 100\%\ =\ 99{,}85\%$
- $Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} x100\% =\ 99{,}66\%$
- $F-Score = \frac{2\times Pr\ \times Re}{Pr+Re}\ x\ 100\% =\ 85{,}11\%$

Berdasarkan percobaan menggunakan model algoritma *Modification Lu-Net* pada fold 5, menghasilkan nilai *confusion matrix* pada *Recall* sebesar 83,09%, *Precision* sebesar 87,23%, *Specificity* sebesar 99,85%, *Accuracy* sebesar 99,66% dan *F-Score* sebesar 85,11%.

Dari pengujian model arsitektur *Modification Lu-Net* dengan menggunakan *K-Fold Cross Validation* didapatkan hasil *confusion matrix* seperti table berikut.

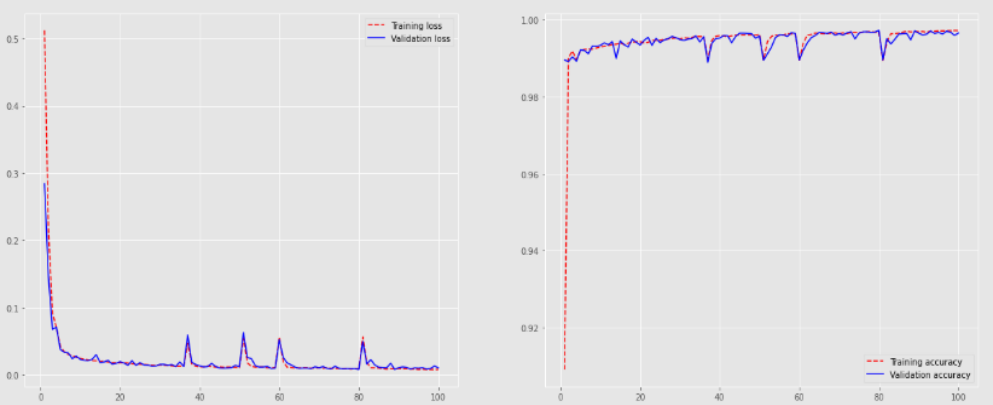
Tabel 4.2 Nilai Hasil Confusion Matrix K Fold Cross Validation Model Arsitektur *Modification Lu-Net*

K-Fold	Recall	Precision	Specificity	Accuracy	F-Score
Fold 1	79,27%	85,99%	99,84%	99,62%	82,49%
Fold 2	79,24%	87,99%	99,86%	99,63%	83,39%
Fold 3	85,43%	83,73%	99,8%	99,65%	84,57%
Fold 4	82,79%	85,22%	99,82%	99,64%	83,99%
Fold 5	83,09%	87,23%	99,85%	99,66%	85,11%
Rata-Rata	**81,96%**	**86,03%**	**99,83%**	**99,64%**	**83,91%**

Berdasarkan data table di atas maka didapatkan nilai Recall tertinggi pada fold ke 3 dengan nilai 85,43%, nilai terendah pada fold ke 2 dengan nilai 79,24%, dan nilai rata-rata 81,96%. Pada Precision nilai tertinggi didapatkan pada Fold ke 2 dengan nilai 87,99%, nilai terendah pada Fold 3 dengan nilai 83,73% dan nilai rata-rata 86,03%. Nilai specificity tertinggi didapatkan pada fold ke 2 dengan nilai 99,86%, nilai terendah pada fold ke 3 dengan nilai 99,80% dan nilai rata-rata 99,83%. Pada Accuracy nilai tertinggi didapatkan pada fold ke 5 dengan nilai 99,66%, nilai terendah pada fold ke 1 dengan nilai 99,62% dan nilai rata-rata 99,64%. Selanjutnya pada F-Score nilai tertinggi didapatkan pada fold ke 5 dengan nilai 85,11%, nilai terendah pada fold ke 1 dengan nilai 82,49% dan nilai rata-rata 83,91%.

4.6 Hasil *Training* Arsitektur *Lu-Net* Rai & Chatterjee (2020)

Berdasarkan scenario uji coba dalam sub bab 4.1. dengan menggunakan model arsitektur *Lu-Net* Rai & Chatterjee (2020) yang menggunakan data training sebanyak 3182, data validasi sebanyak 354, dan data testing sebanyak 393. Pada *training* model *Lu-Net* Rai & Chatterjee (2020) menghasilkan loss sebesar 0,5122 pada epoch pertama dan nilai loss terus menurun sampai epoch ke 100 dengan nilai loss 0,0079. Grafik loss dan accuracy pada training dengan model arsitektur *Lu-Net* Rai & Chatterjee (2020) dapat dilihat pada gambar 4.10.

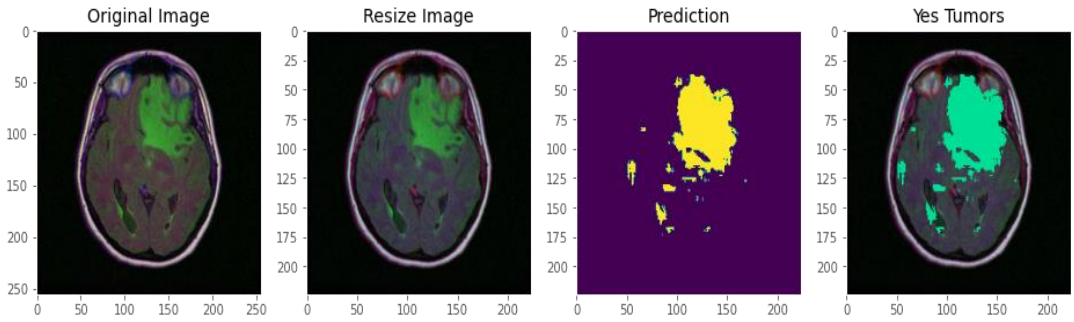


Gambar 4.10 Grafik Training Model Arsitektur *Lu-Net* Rai & Chatterjee (2020)

4.7 Hasil *Testing* Arsitektur *Lu-Net* Rai & Chatterjee (2020)

Selanjutnya pada tahap ini model yang telah melalui tahap *training* di uji dengan *testing.* Pada tahap testing citra di inputkan kemudian masuk pada tahap *preprocessing* dimana citra akan disesuaikan ukuran pikselnya dengan ukuran 224x224x3, selanjutnya citra diuji dengan menggunakan model yang telah disimpan dan menghasilkan prediksi. Adapun hasil dari tiap prosesnya dapat

dilihat pada gambar 4.11 yang merupakan salah satu contoh hasil testing dari total 393 data testing.



Gambar 4.11 Hasil *Testing Lu-Net* Rai & Chatterjee (2020)

Pada gambar pertama merupakan citra asli, kemudian pada gambar kedua merupakan citra yang telah disesuaikan ukurannya, pada gambar ketiga merupakan hasil deteksi tumor otak, dan pada gambar keempat merupakan gambar MRI yang digabungkan dengan hasil deteksi sehingga dapat dilihat lokasi tumor otak pada citra MRI. Sedangkan untuk nilai *confusion matrix* percobaan ini mendapatkan hasil sebagai berikut:

a. True Positive = 14121
b. True Negative = 1497223
c. False Positif = 2250
d. False Negative = 3263

Nilai *confusion matrix* di atas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut: :

- $Recall\ (Re) = \frac{\mathrm{TP}}{\mathrm{TP+FN}} x\ 100\% = 81{,}83\%$
- $Precision(\mathrm{Pr}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{170189}{170189+25734}\ x\ 100\% = 85{,}15\%$

- $Specificity\ (Sp) = \frac{\text{TN}}{\text{TN+FP}}\ x\ 100\% = 99{,}85\%$
- $Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} x100\% = 99{,}64\%$
- $F-Score = \frac{2 \times Pr \times Re}{Pr+Re}\ x\ 100\% = 83{,}46\%$

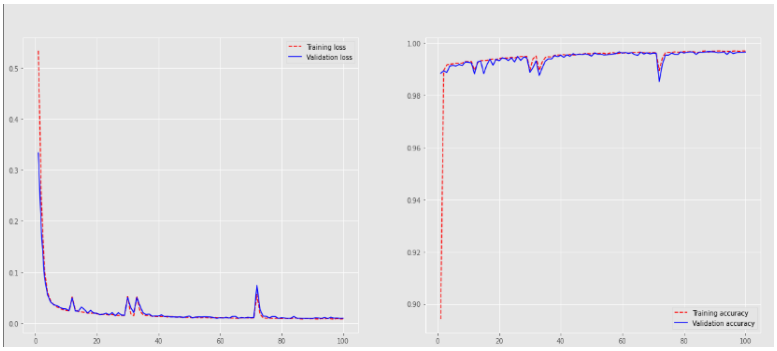
Berdasarkan percobaan menggunakan model algoritma *Lu-Net* Rai & Chatterjee (2020) menghasilkan nilai *confusion matrix* pada *Recall* sebesar 81,83%, *Precision* sebesar 85,15%, *Specificity* sebesar 99,85%, *Accuracy* sebesar 99,64% dan *F-Score* sebesar 83,46%.

4.8 Hasil *Training K-Fold Cross Validation Lu-Net* Rai & Chatterjee (2020)

Tahap ini menguji model dengan pengujian dataset yang berbeda, dimana dataset dibagi ke dalam 5 bagian dan tiap bagian digunakan untuk pengujian model *Lu-Net.* Dalam tiap pengujian model Arsitektur yang digunakan selalu sama sehingga tiap Fold mendapatkan hasil sebagai berikut.

1. Fold 1 Training *Lu-Net* Rai & Chatterjee (2020)

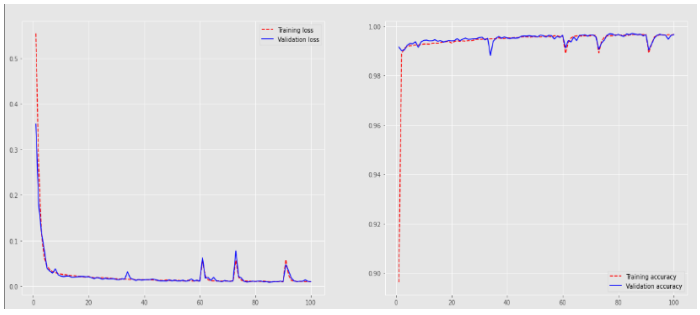
   Proses training Fold ke 1 dengan menggunakan arsitektur *Lu-Net* Rai & Chatterjee (2020), pada epoch pertama mendapatkan nilai loss sebesar 0,5332 dan pada epoch terakhir nilai loss menurun menjadi 0,0085. Grafik loss dan accuracy pada training dengan model arsitektur *Lu-Net* Rai & Chatterjee (2020) dapat dilihat pada gambar 4.12.



Gambar 4.12 Grafik Training Model Arsitektur *Lu-Net* Rai & Chatterjee (2020) Fold 1

2. Fold 2 Training *Lu-Net* Rai & Chatterjee (2020)

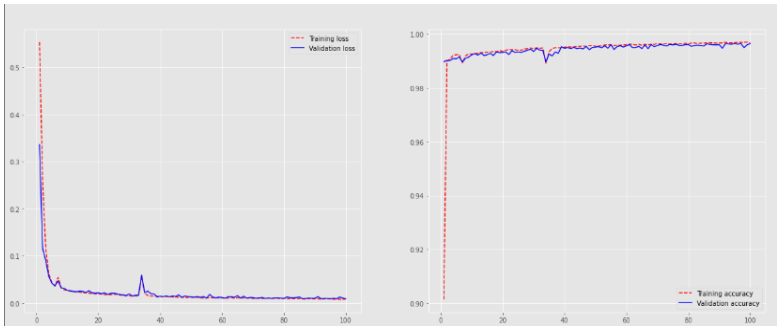
Proses training Fold ke 2 dengan menggunakan arsitektur *Lu-Net* Rai & Chatterjee (2020), pada epoch pertama mendapatkan nilai loss sebesar 0,5538 dan pada epoch terakhir nilai loss menurun menjadi 0,0097. Grafik loss dan accuracy pada training dengan model arsitektur *Lu-Net* Rai & Chatterjee (2020) dapat dilihat pada gambar 4.13.



Gambar 4.13 Grafik Training Model Arsitektur *Lu-Net* Rai & Chatterjee (2020) Fold 2

3. Fold 3 Training *Lu-Net* Rai & Chatterjee (2020)

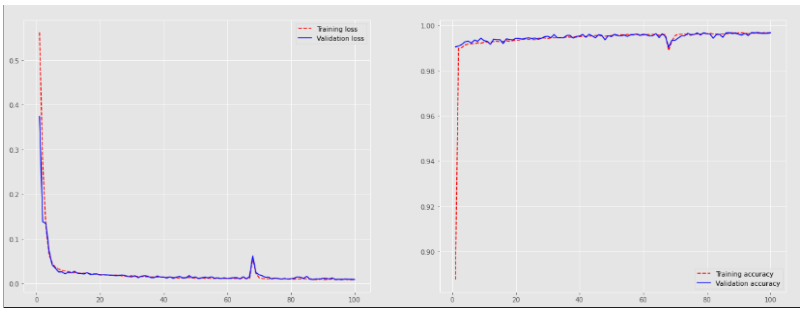
Proses training Fold ke 3 dengan menggunakan arsitektur *Lu-Net* Rai & Chatterjee (2020), pada epoch pertama mendapatkan nilai loss sebesar 0,5532 dan pada epoch terakhir nilai loss menurun menjadi 0,0088. Grafik loss dan accuracy pada training dengan model arsitektur *Lu-Net* Rai & Chatterjee (2020) dapat dilihat pada gambar 4.14.



Gambar 4.14 Grafik Training Model Arsitektur *Lu-Net* Rai & Chatterjee (2020) Fold 3

4. Fold 4 Training *Lu-Net* Rai & Chatterjee (2020)

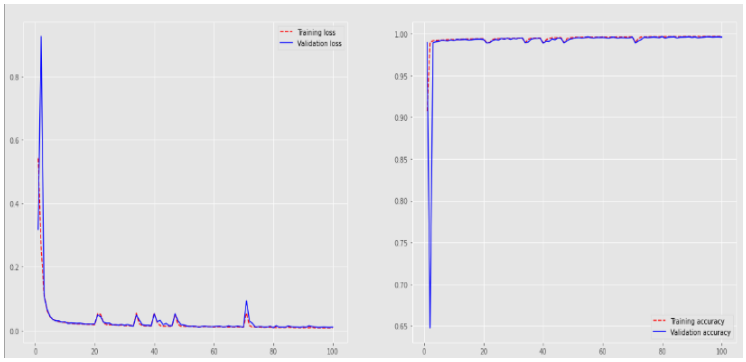
Proses training Fold ke 4 dengan menggunakan arsitektur *Lu-Net* Rai & Chatterjee (2020), pada epoch pertama mendapatkan nilai loss sebesar 0,5619 dan pada epoch terakhir nilai loss menurun menjadi 0,0093. Grafik loss dan accuracy pada training dengan model arsitektur *Lu-Net* Rai & Chatterjee (2020) dapat dilihat pada gambar 4.15.



Gambar 4.15 Grafik Training Model Arsitektur *Lu-Net* Rai & Chatterjee (2020) Fold 4

5. Fold 5 Training *Lu-Net* Rai & Chatterjee (2020)

Proses training Fold ke 5 dengan menggunakan arsitektur *Lu-Net* Rai & Chatterjee (2020), pada epoch pertama mendapatkan nilai loss sebesar 0,5417 dan pada epoch terakhir nilai loss menurun menjadi 0,0091. Grafik loss dan accuracy pada training dengan model arsitektur *Lu-Net* Rai & Chatterjee (2020) dapat dilihat pada gambar 4.16.



Gambar 4.16 Grafik Training Model Arsitektur *Lu-Net* Rai & Chatterjee (2020) Fold 5

4.9 Hasil *Testing K-Fold Cross Validation Lu-Net* Rai & Chatterjee (2020)

Pada subbab ini menjelaskan tentang proses *testing* dari pengujian menggunakan *K-Fold Cross Validation* dimana pada tiap fold diuji dengan data testing yang sama sejumlah 393 data citra MRI. Adapun *testing* tiap fold mendapatkan hasil sebagai berikut.

1. Fold 1 *Testing Lu-Net* Rai & Chatterjee (2020)

   Proses *testing* Fold ke 1 dengan menggunakan arsitektur *Lu-Net* Rai & Chatterjee (2020) mendapatkan nilai hasil pengujian *confusion matrix* sebagai berikut:

   a. True Positive = 11557
   b. True Negative = 1500087
   c. False Positif = 2117
   d. False Negative = 3097

   Nilai *confusion matrix* di atas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut: :

   - $Recall\ (Re) = \frac{\text{TP}}{\text{TP+FN}} x\ 100\% = 78{,}92\%$
   - $Precision(\text{Pr}) = \frac{\text{TP}}{\text{TP+FP}} = \frac{170189}{170189+25734}\ x\ 100\% = 84{,}30\%$

- $Specifity\ (Sp) = \frac{\text{TN}}{\text{TN+FP}}\ x\ 100\%\ =\ 99{,}86\%$
- $Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} x100\% =\ 99{,}66\%$
- $F-Score = \frac{2\times Pr\ \times Re}{Pr+Re}\ x\ 100\% =\ 81{,}52\%$

Berdasarkan percobaan menggunakan model algoritma *Lu-Net* Rai & Chatterjee (2020) pada fold 1, menghasilkan nilai *confusion matrix* pada *Recall* sebesar 78,92%, *Precision* sebesar 84,30%, *Specificity* sebesar 99,86%, *Accuracy* sebesar 99,66% dan *F-Score* sebesar 81,52%.

2. Fold 2 *Testing Lu-Net* Rai & Chatterjee (2020)

Proses *testing* Fold ke 2 dengan menggunakan arsitektur *Lu-Net* Rai & Chatterjee (2020) mendapatkan nilai hasil pengujian *confusion matrix* sebagai berikut:

a. True Positive = 13440
b. True Negative = 1575995
c. False Positif = 1959
d. False Negative = 3525

Nilai *confusion matrix* di atas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut: :

- $Recall\ (Re) = \frac{\text{TP}}{\text{TP+FN}} x\ 100\% = 77{,}27\%$
- $Precision(\text{Pr}) = \frac{\text{TP}}{\text{TP+FP}} = \frac{170189}{170189+25734}\ x\ 100\% = 84{,}27\%$
- $Specifity\ (Sp) = \frac{\text{TN}}{\text{TN+FP}}\ x\ 100\%\ =\ 99{,}87\%$

- $Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} x100\% =\ 99{,}65\%$
- $F-Score = \frac{2 \times Pr \times Re}{Pr+Re}\ x\ 100\% =\ 80{,}62\%$

Berdasarkan percobaan menggunakan model algoritma *Lu-Net* Rai & Chatterjee (2020) pada fold 2, menghasilkan nilai *confusion matrix* pada *Recall* sebesar 77,27%, *Precision* sebesar 84,27%, *Specificity* sebesar 99,87%, *Accuracy* sebesar 99,65% dan *F-Score* sebesar 80,62%.

3. Fold 3 *Testing Lu-Net* Rai & Chatterjee (2020)

Proses *testing* Fold ke 3 dengan menggunakan arsitektur *Lu-Net* Rai & Chatterjee (2020) mendapatkan nilai hasil pengujian *confusion matrix* sebagai berikut:

a. True Positive = 11537
b. True Negative = 1500552
c. False Positif = 1653
d. False Negative = 3116

Nilai *confusion matrix* diatas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut: :

- $Recall\ (Re) = \frac{\mathrm{TP}}{\mathrm{TP+FN}} x\ 100\% = 78{,}56\%$
- $Precision(\mathrm{Pr}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{170189}{170189+25734}\ x\ 100\% = 87{,}23\%$
- $Specifity\ (Sp) = \frac{\mathrm{TN}}{\mathrm{TN+FP}}\ x\ 100\% =\ 99{,}89\%$
- $Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} x100\% =\ 99{,}69\%$

- $F - Score = \frac{2 \times Pr \times Re}{Pr+Re}\ x\ 100\% =\ 82{,}67\%$

Berdasarkan percobaan menggunakan model algoritma *Lu-Net* Rai & Chatterjee (2020) pada fold 3, menghasilkan nilai *confusion matrix* pada *Recall* sebesar 78,56%, *Precision* sebesar 87,23%, *Specificity* sebesar 99,89%, *Accuracy* sebesar 99,69% dan *F-Score* sebesar 82,67%.

4. Fold 4 *Testing Lu-Net* Rai & Chatterjee (2020)

Proses *testing* Fold ke 4 dengan menggunakan arsitektur *Lu-Net* Rai & Chatterjee (2020) mendapatkan nilai hasil pengujian *confusion matrix* sebagai berikut:

a. True Positive = 11735
b. True Negative = 1500313
c. False Positif = 1891
d. False Negative = 2918

Nilai *confusion matrix* diatas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut: :

- $Recall\ (Re) = \frac{\mathrm{TP}}{\mathrm{TP+FN}} x\ 100\% = 79{,}92\%$
- $Precision(\mathrm{Pr}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{170189}{170189+25734}\ x\ 100\% = 85{,}75\%$
- $Specifity\ (Sp) = \frac{\mathrm{TN}}{\mathrm{TN+FP}}\ x\ 100\%\ =\ 99{,}88\%$
- $Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} x100\% =\ 99{,}68\%$
- $F - Score = \frac{2 \times Pr \times Re}{Pr+Re}\ x\ 100\% =\ 82{,}73\%$

Berdasarkan percobaan menggunakan model algoritma *Lu-Net* Rai & Chatterjee (2020) pada fold 4, menghasilkan nilai *confusion matrix* pada *Recall* sebesar 79,92%, *Precision* sebesar 85,75%, *Specificity* sebesar 99,88%, *Accuracy* sebesar 99,68% dan *F-Score* sebesar 82,73%.

5. Fold 5 *Testing Lu-Net* Rai & Chatterjee (2020)

Proses *testing* Fold ke 5 dengan menggunakan arsitektur *Lu-Net* Rai & Chatterjee (2020) mendapatkan nilai hasil pengujian *confusion matrix* sebagai berikut:

a. True Positive = 11447
b. True Negative = 1499917
c. False Positif = 2287
d. False Negative = 3207

Nilai *confusion matrix* diatas digunakan sebagai dasar untuk mengetahui hasil kinerja sistem dengan hasil sebagai berikut: :

- $Recall\ (Re) = \frac{\mathrm{TP}}{\mathrm{TP+FN}} x\ 100\% = 78{,}43\%$
- $Precision(\mathrm{Pr}) = \frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{170189}{170189+25734}\ x\ 100\% = 83{,}30\%$
- $Specifity\ (Sp) = \frac{\mathrm{TN}}{\mathrm{TN+FP}}\ x\ 100\% = 99{,}85\%$
- $Accuracy\ (Acc) = \frac{TP+TN}{TP+TN+FP+FN} x100\% = 99{,}64\%$
- $F - Score = \frac{2 \times Pr \times Re}{Pr+Re}\ x\ 100\% = 80{,}79\%$

Berdasarkan percobaan menggunakan model algoritma *Lu-Net* Rai & Chatterjee (2020) pada fold 5, menghasilkan nilai *confusion matrix* pada *Recall* sebesar 78,43%, *Precission* sebesar 83,30%, *Specificity* sebesar 99,85%, *Accuracy* sebesar 99,64% dan *F-Score* sebesar 80,79%.

Dari pengujian model arsitektur *Lu-Net* Rai & Chatterjee (2020) dengan menggunakan *K-Fold Cross Validation* didapatkan hasil *confusion matrix* seperti table berikut.

Tabel 4.3 Nilai Hasil Confusion Matrix K Fold Cross Validation Model Arsitektur *Lu-Net* Rai & Chatterjee (2020)

K-Fold	Recall	Precision	Specificity	Accuracy	F-Score
Fold 1	78,92%	84,3%	99,86%	99,66%	81,52%
Fold 2	77,27%	84,27%	99,87%	99,65%	80,62%
Fold 3	78,56%	87,23%	99,89%	99,69%	82,67%
Fold 4	79,92%	85,75%	99,88%	99,68%	82,73%
Fold 5	78,43%	83,3%	99,85%	99,64%	80,79%
Rata-Rata	**78,62**%	**84,97**%	**99,87**%	**99,66**%	**81,67**%

Berdasarkan data table di atas maka didapatkan nilai Recall tertinggi pada fold ke 4 dengan nilai 79,92%, nilai terendah pada fold ke 2 dengan nilai 77,27%, dan nilai rata-rata 78,62%. Pada Precision nilai tertinggi didapatkan pada Fold 3 dengan nilai 87,23%, nilai terendah pada Fold 5 dengan nilai 83,30% dan nilai rata-rata 84,97%. Nilai specificity tertinggi didapatkan pada fold ke 3 dengan nilai 99,89%, nilai terendah pada fold ke 5 dengan nilai 99,85% dan nilai rata-rata

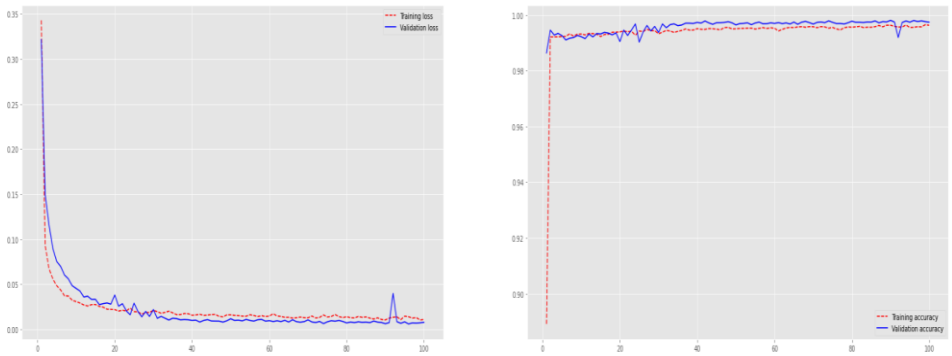
99,87%. Pada Accuracy nilai tertinggi didapatkan pada fold ke 3 dengan nilai 99,69%, nilai terendah pada fold ke 5 dengan nilai 99,64% dan nilai rata-rata 99,66%. Selanjutnya pada F-Score nilai tertinggi didapatkan pada fold ke 4 dengan nilai 82,73%, nilai terendah pada fold ke 2 dengan nilai 80,62% dan nilai rata-rata 81,67%.

4.10 Uji Coba Pengaruh Data *Training* Terhadap Hasil *Accuracy*

Pada tahap ini uji coba dilakukan dengan menggunakan data training yang berbeda, dimana dari data keseluruhan diambil sebanyak 20% untuk data testing, kemudian sisa data dibagi menjadi 5, adapun pembagian data dapat dilihat pada tabel 4.1. Hasil uji coba pengaruh data *training* terhadap *accuracy* adalah seabgai berikut:

1. Hasil *Training* Dengan Data *Training* 20%

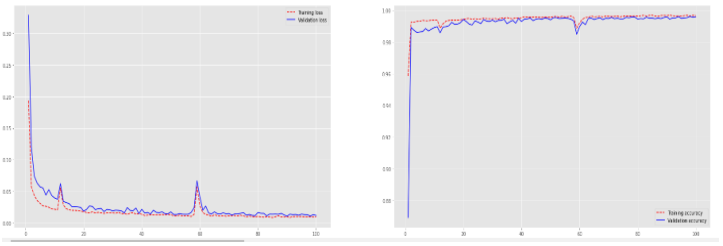
   Proses *training* dengan menggunakan data 20% dan menggunakan arsitektur *Modification Lu-Net*, pada epoch pertama mendapatkan nilai loss sebesar 0.3428 dan pada epoch terakhir nilai loss menurun menjadi 0.0110. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.17.



Gambar 4.17 Grafik *Training* Dengan Data *Training* 20%

2. Hasil *Training* Dengan Data *Training* 40%

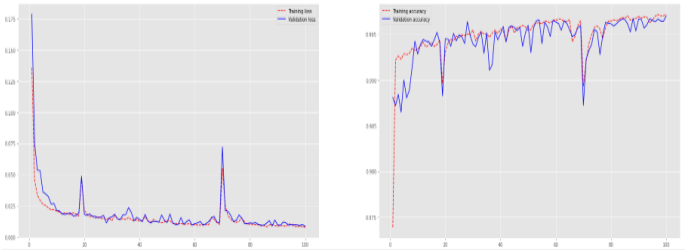
Proses *training* dengan menggunakan data 40% dan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0.1940 dan pada epoch terakhir nilai loss menurun menjadi 0.0099. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.18.



Gambar 4.18 Grafik *Training* Dengan Data *Training* 40%

3. Hasil *Training* Dengan Data *Training* 60%

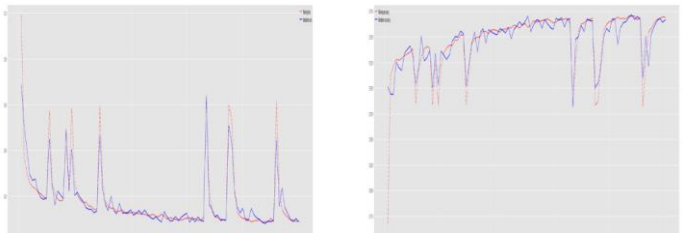
Proses *training* dengan menggunakan data 60% dan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0.1357 dan pada epoch terakhir nilai loss menurun menjadi 0.0078. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.19.



Gambar 4.19 Grafik *Training* Dengan Data *Training* 60%

4. Hasil *Training* Dengan Data *Training* 80%

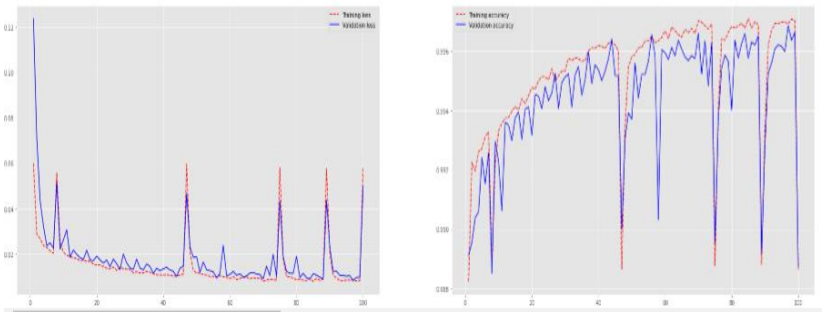
Proses *training* dengan menggunakan data 80% dan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0.1601 dan pada epoch terakhir nilai loss menurun menjadi 0.0191. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.20.



Gambar 4.20 Grafik *Training* Dengan Data *Training* 80%

5. Hasil *Training* Dengan Data *Training* 100%

Proses *training* dengan menggunakan data 100% dan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0.0600 dan pada epoch terakhir nilai loss menurun menjadi 0.0083. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.21.



Gambar 4.21 Grafik *Training* Dengan Data *Training* 100%

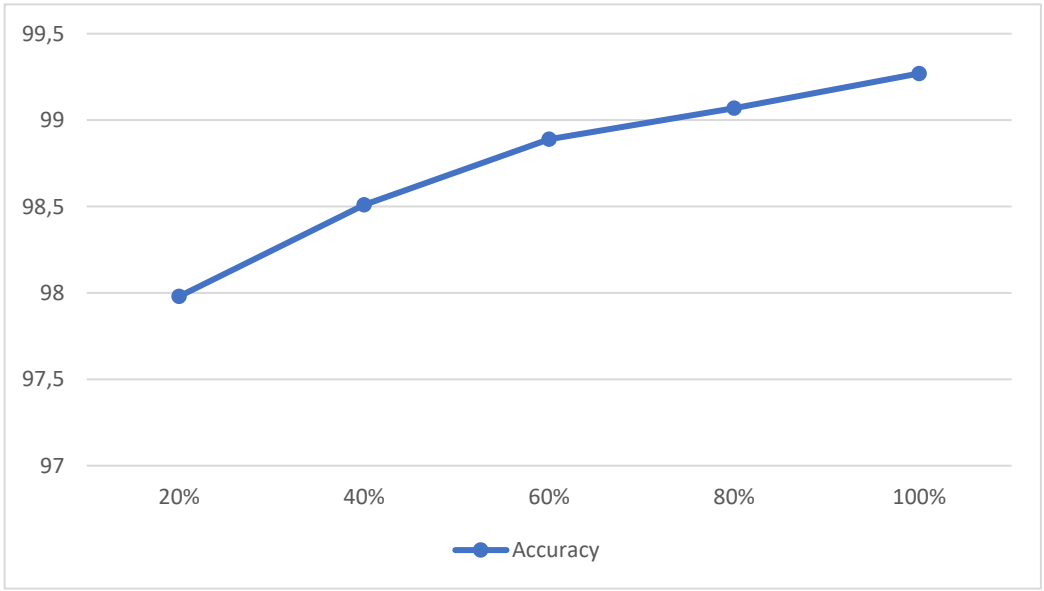
1.10.1 Hasil *Testing* Pengaruh Data *Training* Terhadap Hasil *Accuracy*

Pada uji coba kali ini skenario *testing* menggunakan data testing yang sama pada tiap model hasil *training*. Hasil proses *testing* pada pengaruh data *training* terhadap hasil akurasi akan ditampilkan pada tabel 4.4.

Tabel 4.4 Hasil *Testing* Pengaruh Data *Training* Terhadap Hasil *Accuracy*

Data *Training*	**TP**	**TN**	**FP**	**FN**	***Accuracy***
20%	10113	1533586	28873	2952	97,98%
40%	9371	1542731	19728	3695	98,51%
60%	9551	1548532	13927	3514	98,89%
80%	9117	1551752	10708	3948	99,07%
100%	8635	1555341	7119	4430	99,27%

Dari percobaan ini *Accuracy* pada 20% data training mendapatkan nilai 97.98%, pada 40% data training mendapatkan nilai 98.51%, pada 60% data training mendapatkan nilai 98.89%, pada 80% data training mendapatkan nilai 99.07%, dan pada 100% data training mendapatkan nilai 99.27%. Adapun grafik data hasil *testing* dapat dilihat pada gambar 4.22.



Gambar 4.22 Grafik *Accuracy* Terhadap Data *Training*

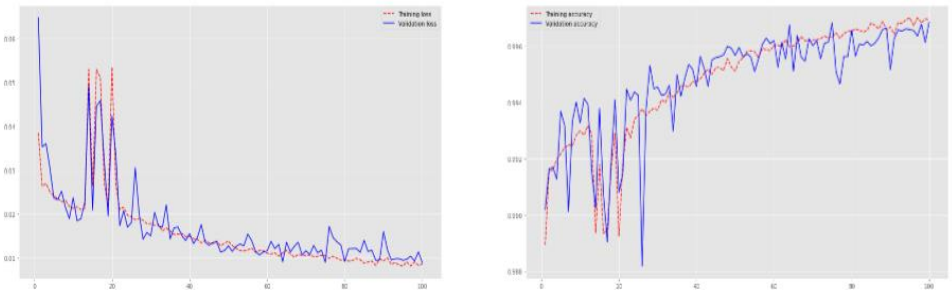
Dari grafik dan hasil perolehan uji coba diatas maka dapat diambil kesimpulan bahwa penggunaan data training yang lebih besar mampu menghasilkan nilai *Accuracy* yang lebih tinggi.

4.11 Hasil Uji Coba Pengaruh *Learning Rate* Terhadap Hasil *Accuracy*

Pada tahap ini uji coba ini dilakukan dengan menggunakan pembagian data sama dengan uji coba tahap pertama, dataset dibagi menjadi 3 bagian, dimana 80% data *training*, 10% data *validation*, dan 10% data *testing*. Kemudian optimizer menggunakan Adam dengan beberapa *learning rate* yang berbeda. *Learning rate* yang digunakan anatara lain 0.01, 0.05, 0.10, 0.15, dan 0,20. Adapun hasil uji coba pengaruh *learning rate* terhadap hasil *accuracy* adalah sebagai berikut:

1. Hasil *Training Learning Rate* 0,01

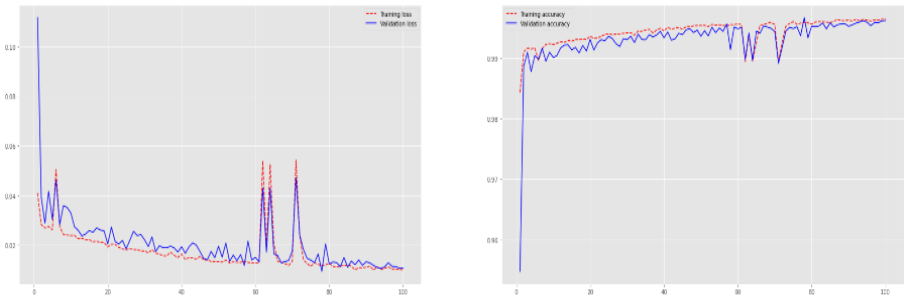
   Proses *training* dengan menggunakan *learning rate* 0,01 dan menggunakan arsitektur *Modification Lu-Net*, pada epoch pertama mendapatkan nilai loss sebesar 0.0386 dan pada epoch terakhir nilai loss menurun menjadi 0.0086. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.23.



Gambar 4.23 Grafik *Training Learning Rate* 0,01

2. Hasil *Training Learning Rate* 0,05

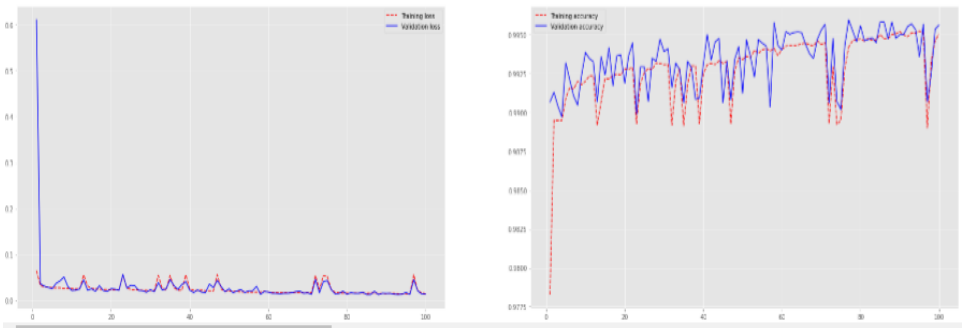
Proses *training* dengan menggunakan *learning rate* 0,05 dan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0.0409 dan pada epoch terakhir nilai loss menurun menjadi 0.0099. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.24.



Gambar 4.24 Grafik *Training Learning Rate* 0,05

3. Hasil *Training Learning Rate* 0,10

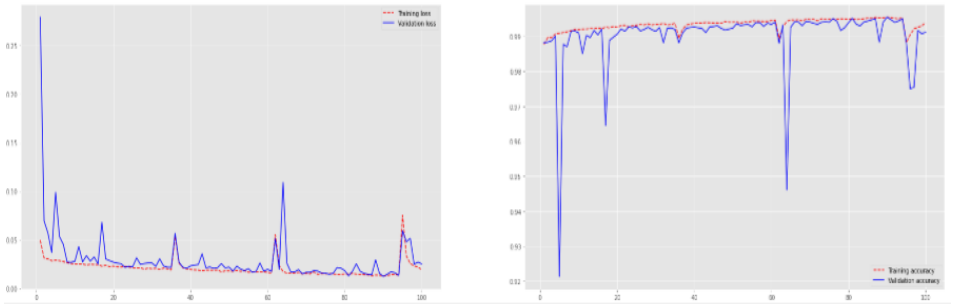
Proses *training* dengan menggunakan *learning rate* 0,10 dan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0.0644 dan pada epoch terakhir nilai loss menurun menjadi 0.0141. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.25.



Gambar 4.25 Grafik *Training Learning Rate* 0,10

4. Hasil *Training Learning Rate* 0,15

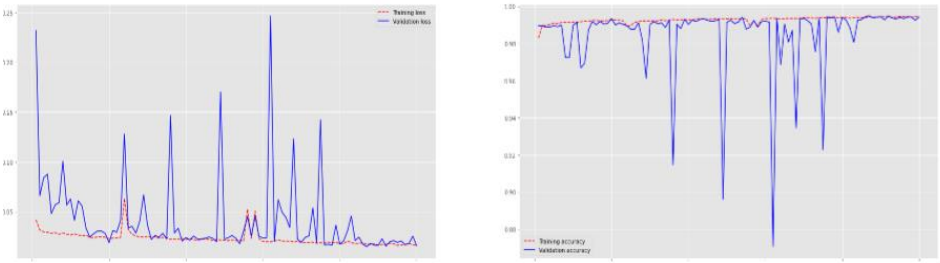
Proses *training* dengan menggunakan *learning rate* 0,15 dan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0.0495 dan pada epoch terakhir nilai loss menurun menjadi 0.0190. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.26.



Gambar 4.26 Grafik *Training Learning Rate* 0,15

5. Hasil *Training Learning Rate* 0,20

Proses *training* dengan menggunakan *learning rate* 0,10 dan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0.0418 dan pada epoch terakhir nilai loss menurun menjadi 0.0164. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.27.



Gambar 4.27 Grafik *Training Learning Rate* 0,20

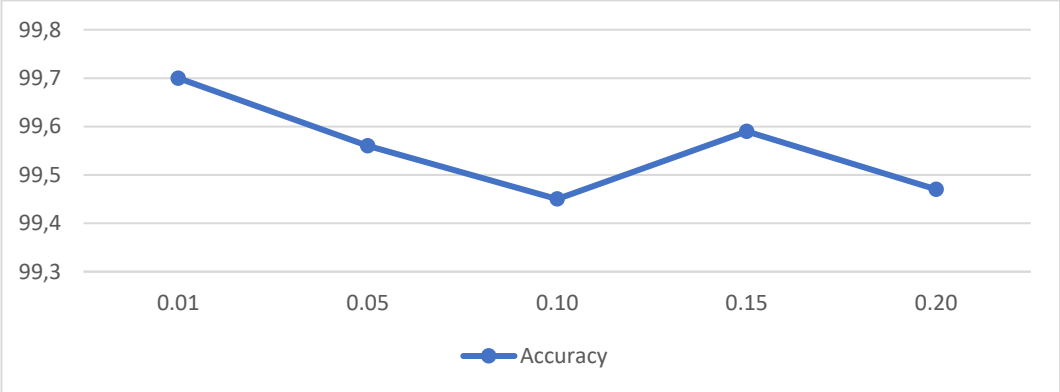
4.11.1 Hasil *Testing* Pengaruh *Learning Rate* Terhadap Hasil *Accuracy*

Pada uji coba kali ini skenario *testing* menggunakan data *testing* sebesar 20% dari data keseluruhan pada tiap model. Hasil proses *testing* pada pengaruh *learning rate* terhadap hasil akurasi akan ditampilkan pada tabel 4.5.

Tabel 4.5 Hasil *Testing* Pengaruh *Learning Rate* Terhadap Hasil *Accuracy*

Learning Rate	**TP**	**TN**	**FP**	**FN**	***Accuracy***
0,01	12366	1499924	1542	3025	99,70%
0,05	12479	1497701	2313	4365	99,56%
0,10	10217	1498311	2649	5680	99,45%
0,15	10094	1500499	1460	4805	99,59%
0,20	10221	1498630	2025	5981	99,47%

Dari percobaan ini *Accuracy* pada *learning rate* 0.01 mendapatkan nilai 99.70%, pada *learning rate* 0.05 mendapatkan nilai 99.56%, pada *learning rate* 0.10 mendapatkan nilai 99.45%, pada *learning rate* 0.15 mendapatkan nilai 99.59%, dan pada *learning rate* 0.20 mendapatkan nilai 99.47%. Adapun grafik hasil uji coba kali ini dapat dilihat pada gambar 4.28.



Gambar 4.28 Grafik *Learning Rate* Terhadap *Accuracy*

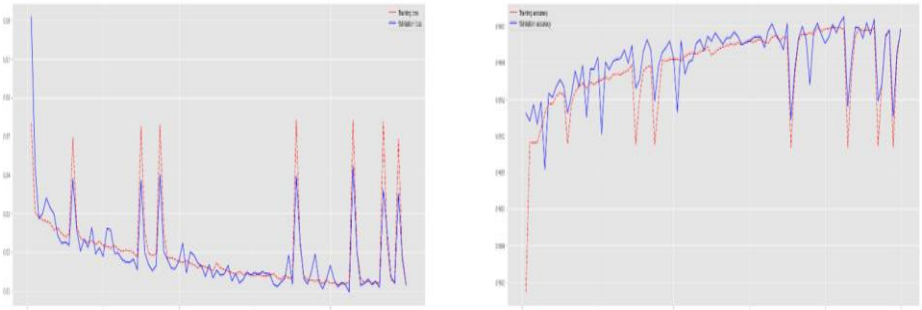
Dari grafik dan hasil perolehan uji coba diatas maka didapatkan *learning rate* terbaik pada 0.01, dan grafik *learning rate* terus turun sampai ke 0.10, kemudian kembali naik pada 0.15 dan turun lagi pada 0.20.

4.12 Uji Coba Pengaruh *Optimizer* Terhadap Hasil *Accuracy*

Pada tahap ini uji coba ini dilakukan dengan menggunakan pembagian data sama dengan uji coba tahap pertama, dataset dibagi menjadi 3 bagian, dimana 80% data *training*, 10% data *validation*, dan 10% data *testing*. Kemudian menggunakan beberapa optimizer yaitu *optimizer Adam, optimizer RMSprop, dan optimizer SGD.* Adapun hasil uji coba pengaruh *optmizer* terhadap hasil *accuracy* adalah sebagai berikut:

1. Hasil Uji Coba *Optimizer Adam*

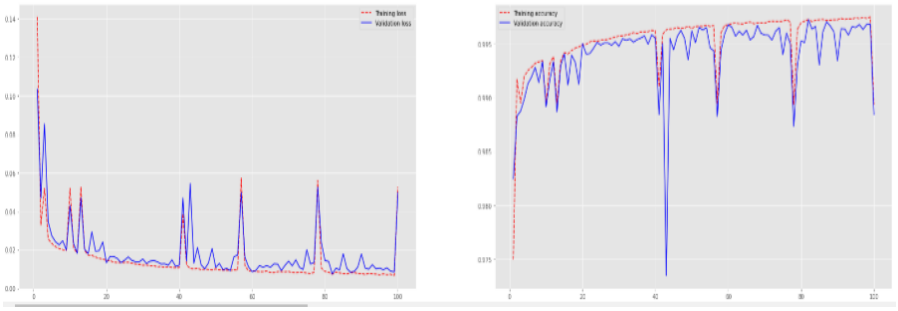
   Proses *training* dengan menggunakan *optimizer Adam* dan menggunakan arsitektur *Modification Lu-Net*, pada epoch pertama mendapatkan nilai loss sebesar 0.0621 dan pada epoch terakhir nilai loss menurun menjadi 0.0073. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.29.



Gambar 4.29 Grafik *Training Optimizer Adam*

2. Hasil Uji *Optimizer RMSprop*

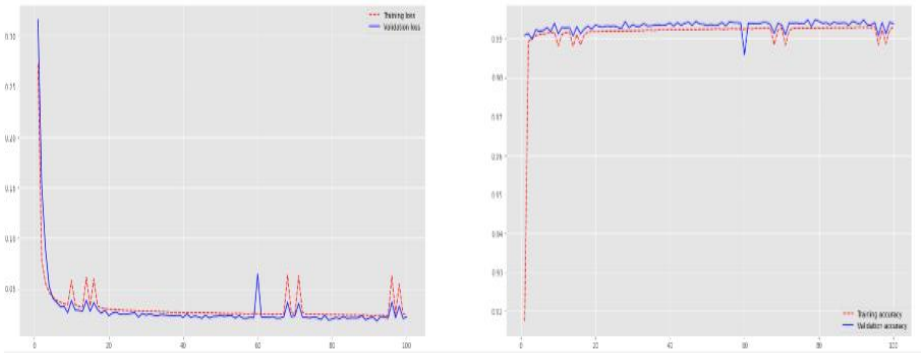
Proses *training* dengan menggunakan *optimizer RMSprop* dan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0.1408 dan pada epoch terakhir nilai loss menurun menjadi 0.0527. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.30.



Gambar 4.30 Grafik *Training Optimizer RMSprop*

3. Hasil Uji *Optimizer SGD*

Proses *training* dengan menggunakan *optimizer SGD* dan menggunakan arsitektur *Modification Lu-Net,* pada epoch pertama mendapatkan nilai loss sebesar 0.2719 dan pada epoch terakhir nilai loss menurun menjadi 0.0236. Grafik loss dan accuracy pada uji coba kali ini dapat dilihat pada gambar 4.31.



Gambar 4.31 Grafik *Training Optimizer SGD*

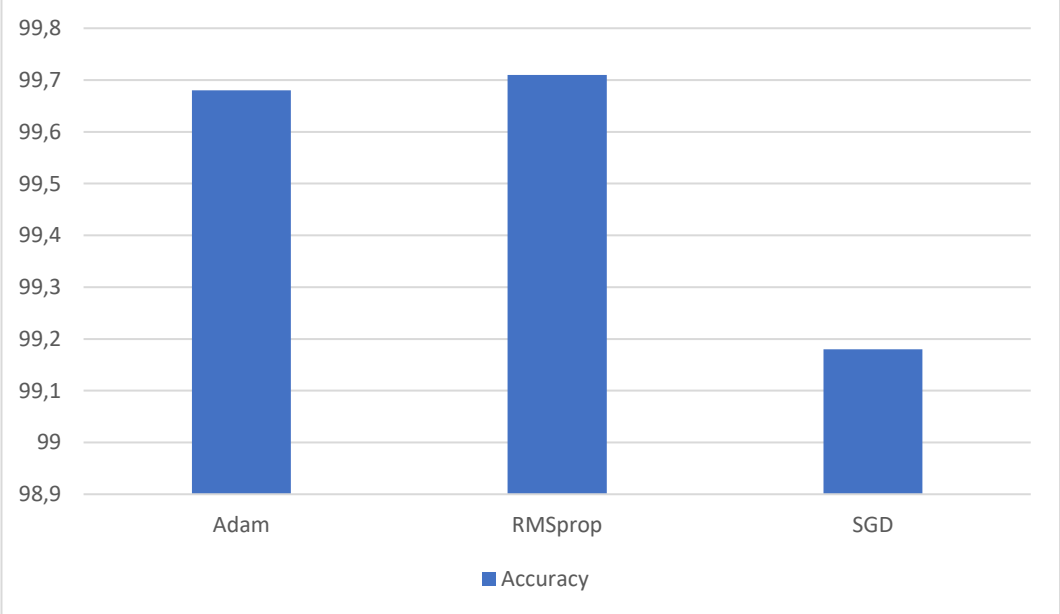
4.12.1 Hasil *Testing* Pengaruh *Optimizer* Terhadap Hasil *Accuracy*

Pada uji coba kali ini skenario *testing* menggunakan data *testing* sebesar 20% dari data keseluruhan pada tiap model. Hasil proses *testing* pada pengaruh *learning rate* terhadap hasil akurasi akan ditampilkan pada tabel 4.6.

Tabel 4.6 Hasil *Testing* Pengaruh *Optimizer* Terhadap Hasil *Accuracy*

Optimizer	**TP**	**TN**	**FP**	**FN**	***Accuracy***
Adam	16404	1495542	2703	2208	99,68%
RMSprop	11675	1500716	1549	2917	99,71%
SGD	5068	1503790	542	7456	99,47%

Dari percobaan ini *Accuracy* pada *optimizer Adam* mendapatkan nilai 99.68%, pada *optimizer RMSprop* mendapatkan nilai 99.71%, dan pada *optimizer SGD* mendapatkan nilai 99.47%. Dari gambar 4.32 grafik dan hasil perolehan uji coba maka didapatkan *optimizer* terbaik pada *RMSprop,* dan terendah pada *optimizer SGD.*



Gambar 4.32 Grafik *Optimizer* Terhadap *Accuracy*

4.13 Pembahasan

Setelah uji coba dilakukan pada model arsitektur *Modification Lu-net* dengan data yang telah dibagi menjadi 80% data *training*, 10% data *validation*, dan 10% data *testing*. Setelah uji coba dilakukan kemudian pengujian lanjutan dilakukan dengan menggunakan metode *K Fold Cross Validation.* Dalam metode pengujian menggunakan fold sebanyak 5 kali.

Selanjutnya dilakukan uji coba dengan arsitektur *Lu-net* Rai & Chatterjee (2020), dengan pembagian data yang sama, yaitu 80% data *training*, 10% data *validation*, dan 10% data *testing*. Kemudian pengujian dilakukan dengan menggunakan metode *K-Fold Cross Validation* dengan fold sebanyak 5 kali.

Data hasil pengujian model arsitektur *Modification Lu-Net,* model arsitektur *Lu-Net* yang dikembangkan oleh Rai & Chatterjee (2020) dan hasil pengujian yang dilakukan oleh Rai & Chatterjee (2020) dapat dilihat pada tabel 4.7.

Tabel 4.7 Perbandingan Confusion Matrix Penelitian

Confusion Matrix	*Modification Lu-Net*		*Lu-Net* Rai & Chatterjee (2020)		Pengujian oleh Rai & Chatterjee (2020)
	Hasil Testing	Rata-Rata *K Fold Cross Validation*	Hasil Testing	Rata-Rata *K Fold Cross Validation*	
Recall	89,45%	81,96%	81,83%	78,62%	100%
Precission	80,11%	86,03%	85,15%	84,97%	97,00%
Specificity	99,77%	99,83%	99,85%	99,87%	95,00%
Accuracy	99,64%	99,64%	99,64%	99,66%	98,00%
F-Score	84,52%	83,91%	83,46%	81,67%	98,00%

Dari data tabel hasil uji diketahui bahwa hasil uji coba dengan menggunakan model arsitektur *Modification Lu-Net* mendapatkan akurasi yang lebih stabil, hal dapat dilihat dalam hasil uji coba model arsitektur *Modification Lu-Net* yang sama dengan rata-rata pengujian *K Fold Cross Validation*. Berbeda dengan model arsitektur *Lu-Net* Rai & Chatterjee (2020) yang hasil uji coba berbeda dengan rata-rata pengujian *K Fold Cross Validation*. Hal ini juga tergambar pada tabel 4.4, dimana peningkatan model modifikasi arsitektur yang diterapkan dapat meningkatkan stabilitas dan nilai performa arsitektur Rai & Chatterjee (2020).

Penggunaan data *training* yang lebih banyak juga mampu menghasilkan nilai akurasi yang lebih tinggi, dimana hal ini tergambar pada tabel 4.8, dimana hasil *accuracy* tertinggi didapatkan pada data *training* terbanyak. Penggunaan *learning rate* dan *optimizer* yang tepat juga sangat berpengaruh terhadap *accuracy*, dimana pada gambar 4.31, *learning rate* terbaik didapatkan pada 0.01, dan pada gambar 4.32, *optimizer* terbaik didapatkan dari *optimizer RMSprop*.

Tabel 4.8 Tabel Pengaruh Data Training Terhadap *Accuracy*

No	**Jumlah Data Training**	**Jumlah Data Testing**	***Accuracy***
1	629 Citra MRI (20%)	785 Citra MRI	97,98%
2	1258 Citra MRI (40%)	785 Citra MRI	98,51%
3	1887 Citra MRI (60%)	785 Citra MRI	98,89%
4	2516 Citra MRI (80%)	785 Citra MRI	99,07%
5	3144 Citra MRI (100%)	785 Citra MRI	99,27%

Selain kelebihan di atas, penggunaan dataset yang besar pada kedua model juga memberikan nilai false positif dan false negative yang cukup besar, dimana

hasil true positif, true negative, false positif dan false negative dapat dilihat pada tabel 4.9.

Tabel 4.9 Perbandingan Parameter Dasar Confusion Matrix

Model		**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Rata-Rata**
Modification Lu-Net	TP	14.288	14.256	15.368	14.895	14.820	**14.725**
	TN	1.496.804	1.497.058	1.496.106	1.496.432	1.496.859	**1.496.652**
	FP	2.105	1.851	2.803	2.477	2.050	**2.257**
	FN	3.661	3.692	2.581	3.053	3.128	**3.223**
Lu-Net Rai & Chatterjee (2020)	TP	11.557	13.440	11.537	11.735	11.447	**11.943**
	TN	1.500.087	1.575.995	1.500.552	1.500.313	1.499.917	**1.515.373**
	FP	2.117	1.959	1.653	1.891	2.287	**1.981**
	FN	3.097	3.525	3.116	2.918	3.207	**3.173**

Berdasarkan hasil pembahasan pada sub bab ini maka dapat disimpulkan bahwa sebagai manusia jangan merasa cukup dengan ilmu yang sudah dimiliki, dikarenakan orang yang tidak memiliki ilmu tidak akan sama dengan orang yang berilmu. Hal ini selaras dengan firman Allah dalam Al-Qur'an surah al-Mujadalah ayat 11.

يٰٓاَيُّهَا الَّذِيْنَ اٰمَنُوْٓا اِذَا قِيْلَ لَكُمْ تَفَسَّحُوْا فِى الْمَجٰلِسِ فَافْسَحُوْا يَفْسَحِ اللّٰهُ لَكُمْۚ وَاِذَا قِيْلَ انْشُزُوْا فَانْشُزُوْا يَرْفَعِ اللّٰهُ الَّذِيْنَ اٰمَنُوْا مِنْكُمْۙ وَالَّذِيْنَ اُوْتُوا الْعِلْمَ دَرَجٰتٍۗ وَاللّٰهُ بِمَا تَعْمَلُوْنَ خَبِيْرٌ

”Niscaya Allah akan mengangkat (derajat) orang-orang yang beriman di antaramu dan orang-orang yang diberi ilmu beberapa derajat. Dan Allah Maha mengetahui apa yang kamu kerjakan.” (AlMujadalah/58: 11).

Ibnu Abbas menafsirkan: “Allah mengangkat derajat orang yang berilmu di atas orang yang tidak berpendidikan ke derajat yang sangat tinggi”. Dari

penelitian ini dapat diambil pelajaran bahwa betapa kompleksnya Allah menciptakan sebuah sistem sebagaimana sistem syaraf manusia dalam mengelola informasi. Dalam penelitian ini tergambar bahwasanya sistem syaraf manusia yang Allah SWT buat untuk mengelola informasi sangatlah kompleks. System syaraf tiruan untuk mendeteksi tumor ini dibuang dengan cara mempelajari sistem kerja syaraf manusia.

د فة ف ما آي سنى فول هاا فعن نغ فسل ب

"Sampaikanlah dariku walau hanya satu ayat" (HR. Bukhari).

Dari hadist diatas juga menjelaskan bahwasanya sebagai manusia dianjurkan untuk menyampaikan ilmu yang telah dipelajari walaupun hanya sedikit. Hal ini dikarenakan dalam hadis nabi yang lain juga menjelaskan bahwasanya ilmu yang bermanfaat tidak akan terputus pahalanya walaupun sudah meninggal dunia.

إِذَا مَاتَ الْإِنْسَانُ انْقَطَعَ عَمَلُهُ إِلَّا مِنْ ثَلَاثَةٍ مِنْ صَدَقَةٍ جَارِيَةٍ وَعِلْمٍ يُنْتَفَعُ بِهِ وَوَلَدٍ صَالِحٍ يَدْعُو لَهُ

"Jika seseorang meninggal dunia, maka terputuslah amalannya kecuali tiga perkara (yaitu): sedekah jariyah, ilmu yang dimanfaatkan, atau do'a anak yang sholeh" (HR. Muslim)

Dalam hadist diatas menjelaskan bahwa ilmu yang bermanfaat merupakan salah satu dari tiga amalan yang pahalanya akan selalu mengalir walaupun sudah meninggal dunia.

BAB V

PENUTUP

5.1 Kesimpulan

Setelah dilakukan penelitian pada deteksi tumor otak menggunakan algoritma *Modification Lu-Net* dengan metode *K-Fold Cross Validation* maka hasil rata-rata yang diperoleh antara lain *recall* 81,96%, *precision* 86,03%, *specificity* 99,83%, *accuracy* 99,64%, dan *f-score* 83,91%.

Selain uji coba untuk mengetahui hasil *confusion matrix,* uji coba juga dilakukan dengan menggunakan jumlah data *training* yang berbeda, beberapa *learning rate* yang berbeda, dan beberapa *optimizer* yang berbeda. Hal ini bertujuan untuk mengetahui faktor-faktor yang mempengaruhi hasil akurasi algoritma *Modification Lu-Net*.

Hasil uji coba deteksi tumor otak menggunakan *Modification Lu-Net* dengan beberapa data *training* yang berbeda menghasilkan kesimpulan bahwa jumlah data sangat berpengaruh terhadap hasil akurasi, dimana semakin banyak data *training* yang digunakan maka algoritma akan semakin baik dalam mendeteksi tumor otak. Sedangkan pemilihan *learning rate* dan *optimizer* yang tepat memiliki peran yang tak kalah penting, dikarenakan dari percobaan pengaruh *learning rate* dan *optimizer* terhadap hasil akurasi menunjukkan bahwa pemilihan *learning rate* dan *optimizer* yang tepat akan mampu menghasilkan akurasi yang lebih baik.

5.2 Saran

Berdasarkan hasil percobaan pada penelitian ini diharapkan bagi peneliti selanjutnya dapat meningkatkan hasil deteksi yang lebih akurat. Maka peneliti memberikan saran terhadap penelitian di masa mendatang yaitu sebagai berikut:

1. Menambah jumlah data yang digunakan, hal ini akan mempengaruhi kinerja sistem agar lebih baik
2. Melanjutkan penelitian terkait suatu algoritma deteksi tumor otak sehingga diharapkan nantinya akan menghasilkan algoritma baru yang jauh lebih akurat.

DAFTAR PUSTAKA

Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2015). Learning activation functions to improve deep neural networks. *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, *2013*, 1–9.

Aman, R. A., Soernarya, M. F., Andriani, R., Munandar, A., Tadjoedin, H., Susanto, E., Nuhonni, S. A., & Nasional, K. P. K. (2016). Panduan Penatalaksanaan Tumor Otak. *Komite Penanggulangan Kanker Nasional*, 1–79. http://kanker.kemkes.go.id/guidelines.php?id=5

Brown, M. J., Hutchinson, L. A., Rainbow, M. J., Deluzio, K. J., & De Asha, A. R. (2017). A comparison of self-selected walking speeds and walking speed variability when data are collected during repeated discrete trials and during continuous walking. *Journal of Applied Biomechanics*, *33*(5), 384–387. https://doi.org/10.1123/jab.2016-0355

Carmack, P. S., Spence, J. S., & Schucany, W. R. (2012). Generalised correlated cross-validation. *Journal of Nonparametric Statistics*, *24*(2), 269–282. https://doi.org/10.1080/10485252.2012.655733

Cireşan, D. C., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in Neural Information Processing Systems*, *4*, 2843–2851.

Deng, L., & Yu, D. (2013). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, *7*(3–4), 197–387. https://doi.org/10.1561/2000000039

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. *45*(8), 1951–1954. https://doi.org/10.1002/chin.200650130

Jung, Y. (2018). Multiple predicting K-fold cross-validation for model selection. *Journal of Nonparametric Statistics*, *30*(1), 197–215. https://doi.org/10.1080/10485252.2017.1404598

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, *53*(8), 5455–5516. https://doi.org/10.1007/s10462-020-09825-6

Kurmi, Y., & Chaurasia, V. (2020). Classification of magnetic resonance images for brain tumour detection. *IET Image Processing*, *14*(12), 2808–2818. https://doi.org/10.1049/iet-ipr.2019.1631

Liu, T., Fang, S., Zhao, Y., Wang, P., & Zhang, J. (2015). *Implementation of Training Convolutional Neural Networks*. http://arxiv.org/abs/1506.01195

Lou, A., Guan, S., & Loew, M. H. (2021). *DC-UNet: rethinking the U-Net*

architecture with dual channel efficient CNN for medical image segmentation. 98. https://doi.org/10.1117/12.2582338

McFaline-Figueroa, J. R., & Lee, E. Q. (2018). Brain Tumors. *American Journal of Medicine*, *131*(8), 874–882. https://doi.org/10.1016/j.amjmed.2017.12.039

Patel, A. P., Fisher, J. L., Nichols, E., Abd-Allah, F., Abdela, J., Abdelalim, A., Abraha, H. N., Agius, D., Alahdab, F., Alam, T., Allen, C. A., Anber, N. H., Awasthi, A., Badali, H., Belachew, A. B., Bijani, A., Bjørge, T., Carvalho, F., Catalá-López, F., … Fitzmaurice, C. (2019). Global, regional, and national burden of brain and other CNS cancer, 1990–2016: a systematic analysis for the Global Burden of Disease Study 2016. *The Lancet Neurology*, *18*(4), 376–393. https://doi.org/10.1016/S1474-4422(18)30468-X

Rai, H. M., & Chatterjee, K. (2020). Detection of brain abnormality by a novel Lu-Net deep neural CNN model from MR images. *Machine Learning with Applications*, *2*(June), 100004. https://doi.org/10.1016/j.mlwa.2020.100004

Rehman, M. U., Cho, S., Kim, J. H., & Chong, K. T. (2020). Bu-net: Brain tumor segmentation using modified u-net architecture. *Electronics (Switzerland)*, *9*(12), 1–12. https://doi.org/10.3390/electronics9122203

Sarkar, S., Kumar, A., Chakraborty, S., Aich, S., Sim, J., & Kim, H. (2020). *A CNN based Approach for the Detection of Brain Tumor Using MRI Scans*. *16580*, 16580–16586.

Wen, P. Y., Macdonald, D. R., Reardon, D. A., Cloughesy, T. F., Sorensen, A. G., Galanis, E., DeGroot, J., Wick, W., Gilbert, M. R., Lassman, A. B., Tsien, C., Mikkelsen, T., Wong, E. T., Chamberlain, M. C., Stupp, R., Lamborn, K. R., Vogelbaum, M. A., Van Den Bent, M. J., & Chang, S. M. (2010). Updated response assessment criteria for high-grade gliomas: Response assessment in neuro-oncology working group. *Journal of Clinical Oncology*, *28*(11), 1963–1972. https://doi.org/10.1200/JCO.2009.26.3541

Wu, J. (2017). Introduction to Convolutional Neural Networks. *Introduction to Convolutional Neural Networks*, 1–31. https://web.archive.org/web/20180928011532/https://cs.nju.edu.cn/wujx/teaching/15_CNN.pdf

LAMPIRAN

Lampiran 1 Kode Program

```
# Set Ukuran
im_width = 224
im_height = 224

# Load Data
train_files = []
mask_files = glob('../input/lunet/lgg-mri-segmentation/kaggle_3m/*/*_mask*')

for i in mask_files:
    train_files.append(i.replace('_mask',''))

print(train_files[:10])
print(mask_files[:10])

#ResBlock Definition
def resblock(X, f):
    X_copy = X  #copy of input

    # main path
    X = Conv2D(f, kernel_size=(1,1), kernel_initializer='he_normal')(X)
    X = BatchNormalization()(X)
    X = Activation('relu')(X)

    X = Conv2D(f, kernel_size=(3,3), padding='same', kernel_initializer='he_normal')(X)
    X = BatchNormalization()(X)

    # shortcut path
    X_copy = Conv2D(f, kernel_size=(1,1), kernel_initializer='he_normal')(X_copy)
    X_copy = BatchNormalization()(X_copy)

    # Adding the output from main path and short path together
    X = Add()([X, X_copy])
    X = Activation('relu')(X)

    return X

def upsample_concat(x, skip):
    X = UpSampling2D((2,2))(x)
    merge = Concatenate()([X, skip])

    return merge

#Modification_Lunet
def unet(input_size=(224, 224, 3)):
    inputs = Input(input_size)

    conv_1 = Conv2D(32, 3, activation='relu', padding='same', kernel_initializer='he_normal')(inputs)
    conv_1 = BatchNormalization()(conv_1)
    conv_1 = Conv2D(32, 3, activation='relu', padding='same', kernel_initializer='he_normal')(conv_1)
    conv_1 = BatchNormalization()(conv_1)
    pool_1 = MaxPool2D((2,2))(conv_1)

    conv_2 = resblock(pool_1, 64)
    pool_2 = MaxPool2D((2,2))(conv_2)

    conv_3 = resblock(pool_2, 128)
```

```
    up_3 = upsample_concat(conv_3, conv_2)
    up_3 = resblock(up_3, 64)

    up_4 = upsample_concat(up_3, conv_1)
    up_4 = resblock(up_4, 32)

    # final output
    out = Conv2D(1, (1,1), kernel_initializer='he_normal', padding='same',
activation='sigmoid')(up_4)

    return Model(inputs=[inputs], outputs=[out])

#Model
model = unet(input_size=(im_height, im_width, 3))
model.summary()

#Data_generator
def train_generator(data_frame, batch_size, aug_dict,
        image_color_mode="rgb",
        mask_color_mode="grayscale",
        image_save_prefix="image",
        mask_save_prefix="mask",
        save_to_dir=None,
        target_size=(224,224),
        seed=1):

    image_datagen = ImageDataGenerator(**aug_dict)
    mask_datagen = ImageDataGenerator(**aug_dict)

    image_generator = image_datagen.flow_from_dataframe(
        data_frame,
        x_col = "filename",
        class_mode = None,
        color_mode = image_color_mode,
        target_size = target_size,
        batch_size = batch_size,
        save_to_dir = save_to_dir,
        save_prefix  = image_save_prefix,
        seed = seed)

    mask_generator = mask_datagen.flow_from_dataframe(
        data_frame,
        x_col = "mask",
        class_mode = None,
        color_mode = mask_color_mode,
        target_size = target_size,
        batch_size = batch_size,
        save_to_dir = save_to_dir,
        save_prefix  = mask_save_prefix,
        seed = seed)

    train_gen = zip(image_generator, mask_generator)

    for (img, mask) in train_gen:
        img, mask = adjust_data(img, mask)
        yield (img,mask)

def adjust_data(img,mask):
    img = img / 223
    mask = mask / 223
    mask[mask > 0.5] = 1
    mask[mask <= 0.5] = 0

    return (img, mask)
```

```
#Split_data
kf = KFold(n_splits = 5, shuffle=False)
df = pandas.DataFrame(data={"filename": train_files, 'mask' : mask_files})
df1 , test = train_test_split(df, test_size=0.1)
df2 = df1.sample(frac=1).reset_index(drop=True)

#Augmented_data and definition variable
train_generator_args = dict(rotation_range=0.2,
                            width_shift_range=0.05,
                            height_shift_range=0.05,
                            shear_range=0.05,
                            zoom_range=0.05,
                            horizontal_flip=True,
                            fill_mode='nearest')
histories = []
loss = []
recall = []
precision = []
specificity = []
accuracy = []
tp = []
tn = []
fp = []
fn = []

EPOCHS = 100
BATCH_SIZE = 32
fold_no = 1

#Metrics
def tp(ytrue, ypred):
    tp = K.sum(K.round(K.clip(ytrue * ypred, 0, 1)))
    tp = tp + K.epsilon()
    return tp

def tn(ytrue, ypred):
    tn = K.sum(K.round(K.clip((1 - ytrue) * (1 - ypred), 0, 1)))
    tn = tn + K.epsilon()
    return tn

def fp(ytrue, ypred):
    fp = K.sum(K.round(K.clip((1 - ytrue) * ypred, 0, 1)))
    fp = fp + K.epsilon()
    return fp

def fn(ytrue, ypred):
    fn = K.sum(K.round(K.clip(ytrue * (1 - ypred), 0, 1)))
    fn = fn + K.epsilon()
    return fn

def recall(ytrue, ypred):
    tp = K.sum(K.round(K.clip(ytrue * ypred, 0, 1)))
    fn = K.sum(K.round(K.clip(ytrue * (1 - ypred), 0, 1)))
    recall_keras = tp / (tp + fn + K.epsilon())
    return recall_keras

def precision(ytrue, ypred):
    tp = K.sum(K.round(K.clip(ytrue * ypred, 0, 1)))
    fp = K.sum(K.round(K.clip((1 - ytrue) * ypred, 0, 1)))
    precision_keras = tp / (tp + fp + K.epsilon())
    return precision_keras

def specificity(ytrue, ypred):
    tn = K.sum(K.round(K.clip((1 - ytrue) * (1 - ypred), 0, 1)))
    fp = K.sum(K.round(K.clip((1 - ytrue) * ypred, 0, 1)))
    return tn / (tn + fp + K.epsilon())
```

```
#Train & Test
for k, (train_index, test_index) in enumerate(kf.split(df2)):
    train_data_frame = df2.iloc[train_index]
    test_data_frame = df2.iloc[test_index]

    train_gen = train_generator(train_data_frame, BATCH_SIZE,
                                train_generator_args,
                                target_size=(im_height, im_width))

    test_gener = train_generator(test_data_frame, BATCH_SIZE,
                                dict(),
                                target_size=(im_height, im_width))

    model = unet(input_size=(im_height, im_width, 3))
    print(f'Training for fold {fold_no} ...')

    model.compile(optimizer='adam',
                  loss="binary_crossentropy",
                  metrics=[["accuracy"], tp,tn,fp, fn, recall, precision, specificity]
                  )
    callbacks = [ModelCheckpoint(str(k+1) + '_resunet_brain_mri_seg.hdf5', verbose=1,
save_best_only=True)]
    history = model.fit(train_gen,
                          steps_per_epoch=len(train_data_frame) / BATCH_SIZE,
                          epochs=EPOCHS,
                          callbacks=callbacks,
                          validation_data = test_gener,
                          validation_steps=len(test_data_frame) / BATCH_SIZE)
    fold_no = fold_no + 1

    model = load_model(str(k+1) + '_resunet_brain_mri_seg.hdf5',  custom_objects={'tp': tp, 'tn': tn,
'fp': fp, 'fn': fn, 'recall': recall, 'precision': precision, 'specificity': specificity,})

    test_gen = train_generator(test, BATCH_SIZE,
                                dict(),
                                target_size=(im_height, im_width))
    results = model.evaluate(test_gen, steps=len(test) / BATCH_SIZE)
    results = dict(zip(model.metrics_names,results))

    histories.append(history)
    loss.append(results['loss'])
    accuracy.append(results['accuracy'])
```