KLASIFIKASI PENYAKIT ALZHEIMER PADA CITRA MRI OTAK MENGGUNAKAN TENSORFLOW

SKRIPSI

Diajukan untuk Memenuhi Tugas Akhir dan Melengkapi Syarat Guna Memperoleh Gelar Sarjana Strata (S-1) dalam ilmu Teknologi Informasi

Diajukan oleh :

Miftakhul Huda
2008096031

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI WALISONGO SEMARANG

TAHUN 2024

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama : Miftakhul Huda
NIM : 2008096031
Jurusan : Teknologi Informasi

Menyatakan bahwa skripsi yang berjudul:

Klasifikasi Penyakit Alzheimer Pada Citra MRI Otak Menggunakan Tensorflow

Secara keseluruhan adalah penelitian/karya saya sendiri, kecuali bagian tertentu yang dirujuk sumbernya.

Semarang, 2 Mei 2024
Pembuat pernyataan

Miftakhul Huda
NIM. 2008096031

PENGESAHAN

KEMENTRIAN AGAMA
UNIVERSITAS ISLAM NEGERI WALISONGO
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Prof. Dr. Hamka Ngaliyan Semarang
Telp.024-7601295 Fax.7615387

PENGESAHAN

Naskah skripsi berikut ini:

Judul : KLASIFIKASI PENYAKIT ALZHEIMER PADA CITRA MRI OTAK MENGGUNAKAN TENSORFLOW
Penulis : **Miftakhul Huda**
NIM : 2008096031
Jurusan : Teknologi Informasi

Telah diujikan dalam sidang tugas akhir oleh Dewan Penguji Fakultas Sains dan Teknologi UIN Walisongo dan dapat diterima sebagai salah satu syarat memperoleh gelar sarjana dalam Teknologi Informasi.

Semarang, 2 Mei 2024

DEPARTEMEN AGAMA REPUBLIK INDONESIA UIN WALISONGO SEMARANG

Penguji I	Penguji II
Nur Cahyo Hendro Wibowo, S.T., M.Kom.	Dr. Masy Ari Ulinuha, ST., M.T
NIP. 19731222 200604 1 001	NIP. 19810812 201101 1007
Penguji III	**Penguji IV**
Hery Mustofa, M.Kom	Siti Nur Aini, S.Kom., M.Kom
NIP. 19870317 201903 1 007	NIP. 19840131 201801 2 001
Pembimbing I	**Pembimbing II**
Dr. Khotibul Umam, M.Kom	Dr. Masy Ari Ulinuha, ST., M.T
NIP. 19790827 201101 1007	NIP. 19810812 201101 1007

NOTA DINAS

NOTA DINAS

Semarang, 1 April 2024

Yth. Ketua Program Studi Teknologi Informasi
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum. Wr. Wb.

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan :

Judul : KLASIFIKASI PENYAKIT ALZHEIMER PADA CITRA MRI OTAK MENGGUNAKAN TENSORFLOW
Nama : **Miftakhul Huda**
NIM : 2008096031
Jurusan : Teknologi Informasi

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo Semarang untuk diujikan dalam Sidang Munaqosyah.

Wassalamu'alaikum. Wr. Wb.

Pembimbing I,

Dr. Khotibul Umam, ST., M.Kom
NIP. 19790827 201101 1007

NOTA DINAS

NOTA DINAS

Semarang, 1 April 2024

Yth. Ketua Program Studi Teknologi Informasi
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum. Wr. Wb.

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan :

Judul : KLASIFIKASI PENYAKIT ALZHEIMER PADA CITRA MRI OTAK MENGGUNAKAN TENSORFLOW
Nama : **Miftakhul Huda**
NIM : 2008096031
Jurusan : Teknologi Informasi

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo Semarang untuk diujikan dalam Sidang Munaqosyah.

Wassalamu'alaikum. Wr. Wb.

Pembimbing II,

Dr. Masy Ari Ulinuha, M.T.
NIP. 198108122011011007

LEMBAR PERSEMBAHAN

Puji dan Syukur penulis ucapkan kepada Allah SWT, laporan tugas akhir ini dapat terselesaikan dengan baik. Karya kecil ini penulis persembahkan untuk:

1. Bapak Ngadiono dan Ibu Muzaroah sebagai orangtua penulis.
2. Faizatul Akla sebagai saudari penulis.
3. Segenap dosen Jurusan Teknologi Informasi.
4. Teman – teman Teknologi Informasi 2020.
5. Almamater Universitas Islam Negeri Walisongo Semarang.

MOTTO

"Barangsiapa yang mengerjakan kebaikan sekecil apapun,
niscaya dia akan melihat (balasan)nya."
(Q.S Al-Zalzalah: 7)

ABSTRAK

Penyakit *Alzheimer* adalah salah satu jenis penyakit yang paling umum terjadi pada lansia, yang ditandai oleh kerusakan pada otak dan mempengaruhi kapasitas memori, bicara, dan tingkah laku. Salah satu cara untuk mendeteksi penyakit Alzheimer adalah dengan menggunakan Citra MRI. Untuk mendeteksi penyakit *Alzheimer* diperlukan suatu metode yang dapat membantu mempermudah dan mempercepat proses deteksi penyakit *Alzheimer*. *Machine learning*, khususnya pada metode *Convolutional Neural Network* (CNN), merupakan metode yang memiliki kemampuan untuk mengelola dan menganalisis data yang besar dan kompleks serta mengambil bentuk gambar 2D sebagai masukan. Pada penilitian ini digunakan metode CNN dengan model arsitektur *ResNet50*. Hasil dari penelitian ini berhasil mencapai tingkat *accuracy* yang tinggi, yaitu sebesar 87%. Dengan nilai rata-rata *recall* sebesar 87%, *precision* sebesar 87%, serta *f1-score* sebesar 87%. Selain itu, model pada penilitian ini mampu memberikan hasil yang konsisten dan stabil. Penelitian ini menunjukkan bahwa penggunaan CNN dengan model *ResNet50* sangat efektif dalam mendeteksi penyakit *Alzheimer* pada Citra MRI. Hasil akurasi yang tinggi ini menunjukkan potensi penting dari pendekatan *machine learning* dalam diagnosis dini penyakit *Alzheimer*.

Kata Kunci : Penyakit Alzheimer, Citra MRI, *Convolutional Neural Network*

KATA PENGANTAR

Dengan memanjatkan puja dan puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik, dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi ini yang berjudul **Klasifikasi Penyakit *Alzheimer* Pada Citra MRI Otak Menggunakan *Tensorflow*** sebagai salah satu syarat untuk menyelesaikan program sarjana (S1) Jurusan Teknologi Informasi Fakultas Sains Dan Teknologi Universitas Islam Negeri Walisongo Semarang.

Penulis menyadari bahwa skripsi ini tidak mungkin terselesaikan tanpa adanya dukungan, bantuan, bimbingan, dan nasehat dari berbagai pihak selama penyusunan skripsi ini. Pada kesempatan ini, izinkan penulis mengucapkan terima kasih kepada :

1. Kedua orang tua penulis, Bapak Ngadiono dan Ibu Muzaroah, yang senantiasa selalu memberikan dukungan baik berupa do'a, moral dan materi sampai saat ini.
2. Bapak Nur Cahyo Hendro Wibowo, S.T., M.Kom selaku Ketua Program Studi Teknologi Informasi Universitas Islam Negeri Walisongo Semarang
3. Bapak Dr. Khotibul Umam, ST., M.Kom dan Bapak Dr. Masy Ari Ulinuha, ST., M.T selaku dosen pembimbing skripsi, yag telah memberikan arahan, kesabaran dan bimbingan dalam mengerjakan skripsi.

4. Seluruh dosen Teknologi Informasi, staf, karyawan dan dosen di lingkungan Universitas Islam Negeri Walisongo Semarang yang telah memberikan ilmu pengetahuan yang tidak ternilai selama menempuh pendidikan.
5. Semua pihak yang telah membantu hingga terselesaikan pembuatan skripsi yang tidak dapat disebutkan satu persatu.

Penulis sadar dalam penulisan skripsi ini masih jauh dari kata sempurna, masih banyak kekurangan dan kesalahan, maka dari itu segala kritik dan saran yang membangun akan menyempurnakan penulisan skripsi ini serta bermanfaat bagi penulis dan para pembaca.

Semarang, 5 April 2024

Miftakhul Huda
NIM. 2008096031

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

A. Latar Belakang

Alzheimer adalah jenis penyakit *neurodegeneratif* yang paling umum dan sering kali mengenai orang lanjut usia. Penyakit ini secara khusus mempengaruhi fungsi kognitif, termasuk daya ingat, pemikiran, dan perilaku seseorang. *Alzheimer* mengakibatkan kerusakan progresif pada otak, yang memengaruhi kemampuan seseorang untuk berpikir, berkomunikasi, dan menjalani kehidupan sehari-hari dengan mandiri (Hidayatul & Sinuraya, 2016).

Penyebab pasti *Alzheimer* masih belum sepenuhnya dipahami, tetapi ditemukan bahwa dalam otak penderita Alzheimer, terdapat penumpukan *plak amyloid-beta* dan degenerasi *neurofibrilar* yang mengganggu fungsi sel saraf (Nisa & Lisiswanti, 2016). Penyakit ini merupakan penyakit kronis yang memburuk seiring waktu dan pada akhirnya dapat mengakibatkan hilangnya kemampuan untuk hidup mandiri.

Berdasarkan informasi dari data Laporan *Alzheimer's Disease Internasional* (ADI) menyebutkan bahwa terdapat 55 juta orang diyakini hidup dengan penyakit Alzheimer atau

demensia lainnya. Jika tidak ada terobosan yang ditemukan, jumlah ini akan meningkat hampir dua kali lipat setiap 20 tahun dan mencapai 78 juta pada tahun 2030 dan 139 juta pada tahun 2050 (Clare, 2022).

Dalam beberapa tahun terakhir, perkembangan dalam bidang pembelajaran mesin dan kecerdasan buatan telah membuka peluang baru untuk mendukung diagnosis penyakit Alzheimer melalui analisis Citra *Magnetic resonance imaging* (MRI). Salah satu pendekatan yang menjanjikan adalah penerapan model jaringan saraf tiruan, khususnya model jaringan saraf konvolusi, untuk klasifikasi Citra *Magnetic resonance imaging* (MRI) (Wildah *et al*., 2020).

MRI-Scan sangat sensitif dan sukses memberikan informasi citra yang baik, sehingga MRI mampu memberikan gambaran yang jelas di dalam otak. Dalam konteks ini, Q.S Al-Mu'Minun(23): 78 menyajikan inspirasi yang menarik. Ayat ini berbunyi :

وَهُوَ الَّذِيْٓ اَنْشَاَ لَكُمُ السَّمْعَ وَالْاَبْصَارَ وَالْاَفْـِٕدَةَۗ قَلِيْلًا مَّا تَشْكُرُوْنَ

Terjemahan :

> *"Allah telah mengeluarkan kamu dari perut ibumu dalam keadaan tidak mengetahui sesuatu pun, dan Dia memberi kamu pendengaran, penglihatan, dan hati, agar kamu bersyukur."*

Penggunaan istilah "penglihatan" dalam ayat ini bisa diartikan secara harfiah atau sebagai representasi pengetahuan yang diberikan oleh Allah kepada manusia (Bisriyah, 2022). Dengan memadukan konsep "penglihatan" dari ayat ini dengan teknologi Citra *Magnetic resonance imaging* (MRI), dapat dicoba untuk mengaplikasikan prinsip-prinsip ilmu pengetahuan modern untuk menerapkan penglihatan (citra warna) secara lebih dalam dan bermakna. Dalam konteks ini, *Tensorflow*, sebagai kerangka kerja *Machine Learning* yang populer dan kuat, memiliki potensi untuk menjadi alat yang sangat efektif dalam mengembangkan model klasifikasi tersebut (Chelghoum *et al*., 2020).

Sejumlah penelitian sebelumnya telah memanfaatkan teknik *Convolutional Neural Network* (CNN) dalam upaya mengklasifikasikan serta mendeteksi penyakit *Alzheimer* melalui analisis citra otak. Misalnya pada penelitian yang dilakukan oleh Rubaiyat Alim Hridhee, dkk, yang berjudul *Alzheimer's Disease Classification From 2D MRI Brain Scans Using Convolutional Neural Networks.* Pada penelitian tersebut mendapatkan hasil akurasi sebesar 94,77% (Hridhee *et al*., 2023). Berbeda hal nya dengan penelitian yang dilakukan oleh Suriya Murugan, dkk, yang berjudul *DEMNET: A Deep Learning Model for Early Diagnosis of Alzheimer Diseases and Dementia*

From MR Images mendapatkan nilai akurasi sebesar 95,23% (Murugan *et al*., 2021).

Tujuan penelitian ini adalah untuk merancang dan membuat model *machine learning* yang akan digunakan dalam pengklasifikasian penyakit *Alzheimer* pada citra MRI otak menggunakan model arsitektur *ResNet50* dan diharapkan hasil penelitian ini dapat membantu dalam deteksi dini penyakit *Alzheimer* sehingga dapat membantu kualitas hidup pasien.

B. Identifikasi Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, dapat diidentifikasi beberapa masalah yang menjadi fokus dari penelitian ini:

1. Penyakit Alzheimer sulit didiagnosis dengan akurat pada tahap awal dan penggunaan model *Machine Learning* untuk klasifikasi citra MRI otak memiliki potensi untuk meningkatkan akurasi diagnosis.
2. Membantu dalam deteksi dini penyakit Alzheimer, yang dapat memberikan manfaat besar bagi kualitas hidup pasien. Namun, penting untuk memahami secara lebih mendalam bagaimana hasil deteksi dini ini dapat secara langsung mempengaruhi pengobatan dan perawatan pasien Alzheimer, serta bagaimana implementasi teknologi ini dapat diintegrasikan ke dalam praktik klinis secara efektif.

C. Rumusan Masalah

1. Bagaimana merancang dan membuat model *machine learning* yang akan digunakan serta hasil performa model dalam pengklasifikasian penyakit *Alzheimer* pada citra MRI otak menggunakan *framework TensorFlow*?
2. Bagaimana performa model *machine learning* dalam pengklasifikasian penyakit *Alzheimer* pada citra MRI menggunakan *framework Tensorflow*?

D. Batasan Masalah

Batasan masalah pada penelitian ini ditentukan agar cakupan tidak meluas atau menyimpang dari yang telah direncanakan. Adapun batasan masalah pada penelitian ini adalah:

1. Dataset yang digunakan dalam penelitian ini adalah *public dataset* yang berjumlah 3000 data berformat *.nii* yang terbagi dalam 3 kelas.
2. Penelitian ini akan menggunakan *TensorFlow* dengan model arsitektur *ResNet50.*
3. Model pada penelitian ini dibuat dengan bahasa pemrograman Python**.**

E. Tujuan Penelitian

1. Merancang dan membuat model *machine learning* yang akan digunakan serta menganalisis performa model dalam

pengklasifikasian penyakit *Alzheimer* pada citra MRI menggunakan *framework TensorFlow*.

2. Menganalisis performa model dalam pengklasifikasian penyakit *Alzheimer* pada citra MRI menggunakan *framework tensorflow.*

F. Manfaat Penelitian

1. Manfaat Teoritis

   Penelitian ini membuktikan bagaimana teknologi kecerdasan buatan, khususnya *TensorFlow*, dapat digunakan dalam diagnosis medis. Dengan kata lain, penelitian ini dapat memberikan manfaat teoritis menggabungkan teknologi kecerdasan buatan dalam bidang medis dan membuka peluang penelitian lebih lanjut serta perbaikan dalam diagnosis dan perawatan penyakit *Alzheimer*.

2. Manfaat Praktis

   Dengan menggunakan algoritma klasifikasi (*Tensorflow*) yang dikembangkan dalam penelitian ini, dapat membantu mendeteksi penyakit *Alzheimer* pada pasien lebih dini. Hal ini dapat mengurangi kesalahan diagnosis dan membantu pasien menerima perawatan yang sesuai.

G. Sistematika Penulisan

Sistematika penulisan ditujukan kepada pembaca agar lebih mudah dalam memahami isi laporan penelitian. Secara garis besar sistematika penulisan proposal ini terdiri dari :

BAB I PENDAHULUAN

Dalam bagian pendahuluan membahas latar belakang penelitian ini dimana untuk mengatahui sebab penelitian ini dilakukan dan selanjutnya membahas rumusan masalah, batasan masalah, tujuan penelitian dan manfaat penelitian.

BAB II LANDASAN PUSTAKA

Pada landasan pustaka membahas hal-hal dasar yang berisi teori-teori yang berkaitan dengan klasifikasi penyakit Alzheimer pada citra MRI otak menggunakan *framework TensorFlow* dan hal-hal yang mendukung dalam hal melakukan klasifikasi ini.

BAB III METODOLOGI PENELITIAN

Bagian ini membahas tentang sumber data dan tahapan yang dilakukan dalam pembuatan penelitian serta gambaran umum sistem yang akan dikerjakan.

BAB IV HASIL DAN PEMBAHASAN

Bagian ini menjelaskan tentang hasil pengolahan data, pengujian, serta menjelaskan kelebihan dan kekurangan dari hasil olahan data.

BAB V KESIMPULAN DAN SARAN

Berisi kesimpulan yang merupakan rangkuman dari hasil penelitian ini dan berisi saran-saran yang membangun untuk pengembangan yang lebih baik di masa depan.

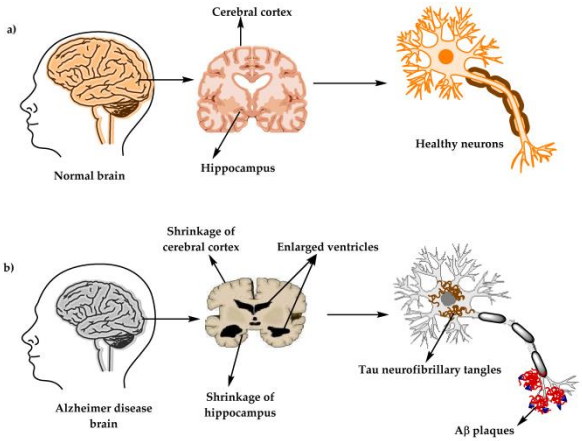
BAB II
LANDASAN PUSTAKA

A. Kajian Teori

1. Penyakit *Alzheimer*

Demensia adalah istilah medis yang mengacu pada sekelompok gejala *neurologis* yang terkait dengan penurunan fungsi kognitif dan gangguan kemampuan berpikir, ingatan, serta kemampuan menjalani kehidupan sehari-hari (Gan *et al*., 2021). Salah satu penyebab umum demensia, yaitu penyakit *Alzheimer*.

Penyakit *Alzheimer* adalah jenis demensia yang paling umum dan dapat didefinisikan sebagai penyakit *neurodegeneratif* progresif lambat. Penyakit Alzheimer disebabkan oleh terbentuknya protein abnormal di otak yang di sebut *β-Amiloid* dan *Tau*, protein ini menyebabkan kerusakan pada neuron-neuron otak. Hilangnya fungsi kognitif secara progresif dapat disebabkan oleh kerusakan otak. Kerusakan otak ini dimulai dari *Hippocampus,* yang dimana area ini beratnggung jawab dalam membentuk ingatan (Mavroudis *et al.*, 2019).



Gambar 2. 1 Struktur fisiologis otak dan neuron pada (a) otak sehat dan (b) otak penyakit *Alzheimer*

Gejala awal penyakit *Alzheimer* bisa sangat ringan dan seringkali tidak terdiagnosis dengan mudah. Salah satu gejala awal yang paling umum adalah gangguan ingatan jangka pendek, di mana seseorang mungkin kesulitan mengingat informasi baru atau peristiwa yang baru saja terjadi. Selain itu, ada juga penurunan kemampuan untuk melakukan tugas-tugas sehari-hari yang sebelumnya dilakukan dengan mudah, seperti kesulitan mengatur jadwal atau menemukan kata-kata yang tepat dalam percakapan (Benoit *et al*., 2020).

Penderita *Alzheimer* juga mungkin mengalami perubahan mood dan perilaku, seperti kebingungan, kecemasan, atau depresi. Gejala awal ini seringkali

diperparah seiring berjalannya waktu, sehingga penting untuk segera mencari bantuan medis jika ada kekhawatiran mengenai penyakit Alzheimer.

Alzheimer mengalami perkembangan secara progresif melalui beberapa tahapan yang berbeda. Berikut adalah tiga tahapan utama dari *Alzheimer*:

a. Tahap Awal (*Mild Alzheimer's Disease*)

Tahap awal penyakit Alzheimer sering dimulai dengan gejala ringan yang mungkin sulit diidentifikasi. Pada tahap ini, penderita mungkin mengalami kesulitan dalam mengingat informasi baru, seperti nama orang atau tempat, serta mengalami penurunan kemampuan untuk melakukan tugas-tugas sehari-hari seperti mengatur keuangan atau merencanakan jadwal. Meskipun gejala ini mungkin masih bisa diatasi, mereka dapat menjadi semakin mengganggu dalam kehidupan sehari-hari.

b. Tahap Menengah (*Moderate Alzheimer's Disease*)

Pada tahap menengah, gejala Alzheimer menjadi lebih nyata dan mempengaruhi fungsi kognitif secara signifikan. Penderita akan mengalami penurunan ingatan yang lebih parah, sering kali tidak dapat mengenali anggota keluarga mereka atau tempat-

tempat yang biasanya dikenali. Perubahan perilaku juga lebih mencolok, termasuk kebingungan, kecemasan, atau bahkan agresi. Kemampuan berbicara dan berkomunikasi juga akan terganggu, dan penderita mungkin kesulitan mencari kata-kata yang tepat atau memahami percakapan.

c. Tahap Lanjut (*Severe Alzheimer's Disease*)

Tahap lanjut Alzheimer adalah tahap paling parah, di mana gejala menjadi sangat menghancurkan dan membatasi kemampuan penderita untuk melakukan tugas-tugas dasar sehari-hari. Pada tahap ini, penderita mungkin memerlukan bantuan penuh dalam makan, berpakaian, mandi, dan perawatan pribadi lainnya. Mereka bisa kehilangan kemampuan berbicara sama sekali dan menjadi semakin terisolasi. Meskipun ada periode *luciditas* di mana mereka bisa lebih sadar, tahap lanjut Alzheimer akhirnya mengarah pada penurunan fisik yang serius dan berpotensi mematikan.

2. *Magnetic Resonance Imaging*

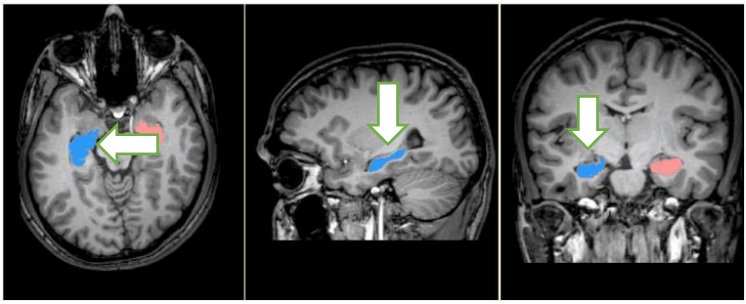
Magnetic Resonance Imaging (MRI) merupakan suatu teknologi diagnostik medis yang memanfaatkan kombinasi medan magnet dan gelombang radio frekuensi

untuk menciptakan representasi visual yang sangat rinci serta tiga dimensi dari struktur internal dalam tubuh manusia (Kang, 2019). Metode ini memungkinkan pencitraan internal yang mendalam, memperlihatkan organ-organ dan jaringan tubuh secara detail, dan memberikan dasar yang kuat untuk diagnosis yang akurat serta pemilihan pengobatan yang tepat (Marques *et al*., 2019).

Pertama, saat seseorang menjalani MRI, tubuh mereka ditempatkan di dalam tabung panjang yang berada di dalam medan magnet yang sangat kuat. Magnet ini mengarahkan proton-proton dalam inti atom tubuh ke arah yang sama. Proton-proton ini pada dasarnya adalah kecil, seperti bola magnet yang bergerak dan berputar. Kemudian, sinyal radio yang dikirimkan ke dalam tubuh meresap ke dalam jaringan dan menyebabkan proton-proton ini berpindah ke arah yang berbeda (Liu, 2020). Kemudian, ketika sinyal radio dimatikan, proton-proton kembali ke posisi semula dalam medan magnet. Ketika mereka kembali ke posisi semula, mereka menghasilkan sinyal kembali yang ditangkap oleh detektor MRI.

Perbedaan dalam sinyal-sinyal ini diubah menjadi gambar-gambar berwarna yang sangat rinci yang

menggambarkan berbagai struktur dalam tubuh manusia (Chandarana *et al*., 2018). Oleh karena itu, MRI adalah alat yang sangat berguna dalam mendiagnosis berbagai kondisi medis, termasuk cedera otak, penyakit jantung, dan kanker, karena dapat memberikan informasi yang sangat rinci tentang organ dan jaringan dalam tubuh manusia.



Gambar 2. 2 MRI otak dengan anak panah yang menunjukkan bagian *Hippocampus*

Pada Gambar 2.2, terlihat dengan jelas anak panah yang menunjuk ke area *hippocampus* pada otak. *Hippocampus* merupakan bagian yang sangat sensitif dan cenderung mengalami kerusakan awal ketika seseorang terkena penyakit *Alzheimer*. Gambar ini menggambarkan pentingnya pengamatan terhadap area tersebut dalam konteks diagnosa penyakit *Alzheimer*, karena kerusakan

pada *hippocampus* sering kali menjadi tanda pertama atau indikator utama perkembangan penyakit ini.

3. *Neuroimaging Informatics Technology Initiative*

NIfTI adalah format file khusus neuroimaging yang dikembangkan oleh *National Institutes of Health* (NIH) pada awal tahun 2000-an untuk menyempurnakan format *Analyze* yang digunakan sebelumnya. Format ini menggunakan beberapa bidang yang tidak dimanfaatkan sepenuhnya dalam *header Analyze 7.5* untuk merekam data baru, seperti orientasi gambar, untuk menghilangkan ambiguitas kiri-kanan dalam studi otak. Selain itu, NIfTI juga dapat mempertahankan jenis data yang berbeda dari format Analyze, seperti unsigned 16-bit (Elhadad et al., 2024).

Meskipun file, *header*, dan piksel data berbeda yang dihasilkan oleh format tersebut, hasilnya adalah satu file gambar yang disimpan dengan ekstensi '.nii'. File-file tersebut berukuran 348 byte untuk header dan piksel data dengan ekstensi '.hdr' dan '.img,' dan 352 byte untuk satu file '.nii' karena adanya empat byte tambahan di bagian akhir, terutama untuk membuat ukurannya habis dibagi 16, dan menjamin mekanisme penyimpanan metadata lebih banyak, jika empat byte ini bukan nol. Dalam

praktiknya, format NIfTI yang disempurnakan untuk pemrosesan data resonansi magnetik berbobot difusi telah dikembangkan (Elhadad et al., 2024).

4. *Machine Learning*

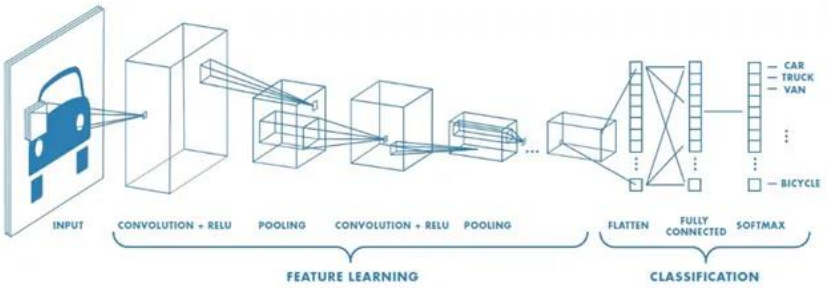
Machine Learning adalah cabang dari kecerdasan buatan yang fokus pada pengembangan algoritma yang memungkinkan komputer untuk belajar dari data dan melakukan tugas tertentu tanpa harus secara eksplisit diprogram. Secara teoritis, *Machine Learning* mengambil inspirasi dari konsep pembelajaran manusia, di mana sistem komputer berusaha untuk meniru kemampuan manusia dalam mengenali pola dan mengambil keputusan berdasarkan pengalaman (Mahadevkar *et al*., 2022).

Ada berbagai jenis algoritma *Machine Learning* yang digunakan tergantung pada tugas yang ingin diselesaikan. Beberapa jenis algoritma populer dalam *Machine Learning* meliputi *Regresi Linier, K-Nearest Neighbors, Decision Trees, Random Forest, Support Vector Machines,* dan *Neural Networks* (Shinde & Shah, 2018). Masing-masing algoritma memiliki kegunaan dan kompleksitas yang berbeda-beda. Misalnya, *Neural Networks* adalah algoritma yang digunakan dalam *Deep Learning*, dan mereka mampu memodelkan data yang sangat kompleks, seperti dalam

pengenalan gambar dan bahasa alami (Shinde & Shah, 2018).

5. *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu jenis arsitektur jaringan saraf tiruan yang dirancang khusus untuk tugas-tugas pemrosesan gambar dan pengenalan pola visual (Lu *et al.*, 2019). *Algortima* ini adalah sebuah bentuk *Multi Layer Perceptron* yang mengambil inspirasi dari struktur jaringan saraf biologis manusia. Dalam arsitektur *Convolutional Neural Network* (CNN), setiap layer dibedakan oleh dimensinya, yang terdiri dari tinggi (*height*), lebar (*width*), dan kedalaman (*depth*) (Cho & Kang, 2019).



Gambar 2. 3 Arsitektur CNN

Dalam arsitektur CNN, langkah pertama adalah tahap konvolusi, yang merujuk pada Gambar 2.3 sebagai panduan. Tahap ini melibatkan penggunaan *kernel* dengan

ukuran tertentu, dan jumlah *kernel* yang diterapkan biasanya bergantung pada jumlah fitur yang ingin diekstraksi. Proses konvolusi ini bertujuan untuk mengekstraksi fitur-fitur relevan dari citra input. Setelah konvolusi, langkah selanjutnya adalah menerapkan fungsi aktivasi, sering kali menggunakan *ReLU* (*Rectified Linear Unit*), yang membantu dalam memperkenalkan unsur *non-linearitas* ke dalam model (Daanouni *et al*., 2022).

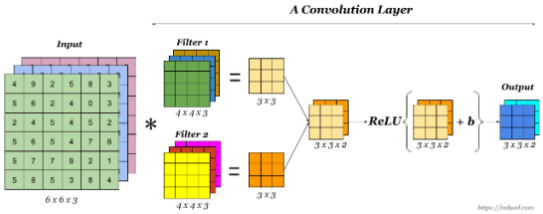
Selanjutnya, hasil dari tahap konvolusi dan aktivasi ini dijalani melalui tahap *pooling*, yang dapat berupa *max-pooling* atau *average-pooling*, dan bertujuan untuk mereduksi dimensi peta fitur serta menjaga fitur-fitur utama. Tahap konvolusi dan *pooling* ini dapat diulang beberapa kali, dengan setiap iterasi menambah kedalaman peta fitur, sehingga model dapat memahami fitur-fitur semakin kompleks (Daanouni *et al*., 2022).

Setelah tahap konvolusi dan *pooling* selesai dan peta fitur yang cukup telah diperoleh, langkah berikutnya adalah menghubungkan hasilnya ke dalam jaringan saraf terhubung penuh (*fully connected neural network*) untuk mendapatkan *output* kelas yang sesuai (Daanouni *et al*., 2022).

a. *Convolution Layer*

Lapisan pertama dalam *Convolutional Neural Network* (CNN) adalah lapisan konvolusi, yang merupakan komponen kunci dalam proses ekstraksi fitur penting dari gambar. Dalam operasi konvolusi, suatu matriks filter diterapkan pada matriks input ini untuk mengekstraksi fitur-fitur yang relevan (Pang *et al*., 2018).

Konsep dasar di balik proses ini adalah melakukan pergeseran matriks filter ke seluruh matriks input, sambil melakukan operasi perkalian dan penjumlahan pada setiap langkah. Hasil dari operasi ini adalah menciptakan matriks baru yang disebut sebagai *feature map*, yang secara grafis menggambarkan fitur-fitur yang telah berhasil diekstraki dari gambar asli (Mao *et al*., 2019).



Gambar 2. 4 *A Convolution Layer*

b. *Activation Function Layer*

Fungsi aktivasi adalah salah satu komponen kunci dalam jaringan saraf (*Neural Networks*) yang memegang peran yang sangat penting. Fungsinya adalah untuk menentukan apakah *neuron* (juga disebut sebagai *node*) dalam jaringan tersebut harus aktif atau tidak. Perlu ditekankan bahwa fungsi aktivasi tidak diterapkan pada *neuron-neuron* dalam lapisan input, tetapi hanya pada lapisan tersembunyi (*hidden layer*) dan lapisan *output* (Ravichandiran, 2019).

Pentingnya fungsi aktivasi terletak pada kemampuannya untuk memperkenalkan unsur *non-linear* dalam jaringan saraf. Tanpa fungsi aktivasi, *neuron-neuron* dalam jaringan akan berperilaku secara *linear*, mirip dengan *regresi linear*. Namun, dengan menerapkan fungsi aktivasi, kita memberikan elemen *non-linear* yang penting untuk memungkinkan jaringan saraf memahami dan memodelkan pola-pola yang kompleks dalam data. Salah satu fungsi aktivasi yang paling umum digunakan dalam CNN adalah *Rectified Linear Unit* (ReLU). Selain *ReLU*, ada juga beberapa fungsi

aktivasi lain yang digunakan dalam jaringan saraf, seperti *Sigmoid*, *Hyperbolic Tangent* (Tanh) dan *Softmax* (Rasel *et al*., 2022).

1) *Rectified Linear Unit* (ReLU)

   ReLU adalah jenis fungsi aktivasi yang memiliki ciri khas nilai outputnya berkisar dari 0 hingga tak terhingga. Ini berarti bahwa ketika nilai input (x) kurang dari 0, fungsi aktivasi ReLU menghasilkan nilai 0 sebagai output, sedangkan jika nilai input (x) lebih besar atau sama dengan 0, fungsi ini akan mengembalikan nilai input tersebut sebagai outputnya (Firmansyah & Hayadi, 2022).

2) *Sigmoid*

   Fungsi aktivasi *Sigmoid* adalah salah satu yang paling umum digunakan dalam konteks klasifikasi dua kelas. *Sigmoid* berfungsi untuk mengubah rentang nilai dari inputnya ke dalam rentang antara 0 hingga 1 (Esrayanti Simanjuntak *et al*., 2023).

3) *Hyperbolic Tangent* (Tanh)

   Fungsi aktivasi Tanh memiliki ciri khas rentang nilai yang berkisar antara -1 hingga 1.

Ini berarti bahwa ketika input yang diberikan ke fungsi Tanh adalah positif, output yang dihasilkan akan mendekati 1, sementara jika inputnya negatif, outputnya akan mendekati -1 (Firmansyah & Hayadi, 2022).

4) *Softmax*

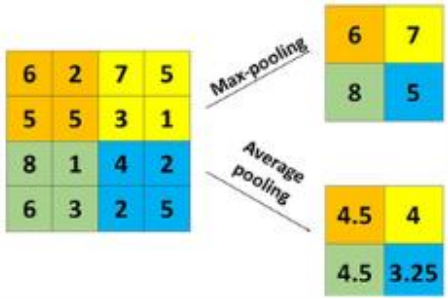
Fungsi aktivasi *softmax* pada dasarnya dapat dianggap sebagai generalisasi dari fungsi aktivasi sigmoid yang sering digunakan dalam konteks *multiclass classification* di jaringan saraf (*neural network*). Perbedaan utamanya terletak pada output yang dihasilkan oleh fungsi *softmax*, yang memberikan probabilitas atau peluang untuk setiap kelas sebagai nilai output. Dengan kata lain, *softmax* menghasilkan distribusi probabilitas di antara berbagai kelas, yang berarti bahwa jumlah probabilitas semua kelas akan selalu sama dengan 1 (Falakhi *et al*., 2022).

c. *Polling Layer*

Dalam operasi konvolusi, kita menghasilkan matriks baru yang dikenal sebagai *feature map*, yang mencerminkan fitur-fitur yang diekstraksi dari

gambar. Namun, seringkali *feature map* yang dihasilkan memiliki dimensi yang cukup besar, dan untuk mengatasi hal ini, kita memerlukan proses yang dikenal sebagai *pooling operation*. Tujuan utama dari *pooling operation* adalah untuk mengurangi dimensi dari *feature map* sambil tetap mempertahankan informasi penting yang relevan. Proses *pooling* ini juga dikenal sebagai *subsampling* atau *downsampling*. Dengan demikian, lapisan *pooling* menjadi alat yang sangat efektif dalam mengurangi kompleksitas data dengan cara mengurangi jumlah dimensi dan secara selektif mempertahankan fitur-fitur yang esensial yang diperlukan dalam proses pelatihan jaringan saraf (Christlein *et al*., 2019).

Terdapat beberapa jenis pooling yang umum digunakan, seperti *max-pooling* dan *average-pooling*. *Max-pooling* mengambil nilai maksimum dari area yang dipilih, sementara *average-pooling* mengambil rata-rata nilai di dalam area tersebut.



Gambar 2. 5 Proses *Pooling*

Seperti yang terlihat pada ilustrasi Gambar 2.5, saat kita memiliki sebuah peta fitur dengan dimensi 4x4 dan menerapkan operasi *max pooling* dengan *kernel* berukuran 2x2, hasilnya adalah peta fitur baru dengan ukuran yang lebih kecil, yaitu 2x2. Prinsipnya adalah setiap *kernel* berfungsi untuk menangani area berukuran 2x2 pada peta fitur asli dan kemudian mengambil nilai maksimum dari keempat piksel dalam area tersebut. Dengan kata lain, *kernel* bergerak melintasi peta fitur dengan langkah tertentu, dan di setiap langkahnya, nilai maksimum diambil. Hal ini mengakibatkan pengurangan dimensi karena setiap kali kernel bergerak, hanya nilai maksimum yang dijaga, sehingga menghasilkan peta fitur yang lebih kecil

namun tetap mencerminkan informasi yang signifikan dari peta fitur asli.

d. *Optimization*

Optimization adalah salah satu aspek kunci dalam pembelajaran *neural network*. Dalam konteks ini, belajar mengacu pada proses yang pada dasarnya adalah upaya untuk mengoptimalkan jaringan, yaitu meminimalkan kesalahan (*error*) atau biaya (*cost*) serta mencari titik di mana kesalahan paling sedikit terjadi. Hal ini mencakup penyesuaian secara bertahap terhadap koefisien *neural network* (N. Zhang *et al*., 2020).

Salah satu pendekatan optimasi yang sangat mendasar adalah metode *gradient descent*, yang bekerja dengan mengikuti *gradien* (perubahan) kesalahan untuk menemukan titik minimum. Namun, ada beberapa variasi optimasi yang memperkaya pendekatan ini dengan peningkatan tambahan. Sebagai contoh, *TensorFlow*, salah satu kerangka kerja *deep learning*, menyediakan beberapa opsi pengoptimalan model seperti *Gradient Descent Optimizer, Adam Optimizer, RMSProp Optimizer, Adagrad Optimizer, Ftrl*

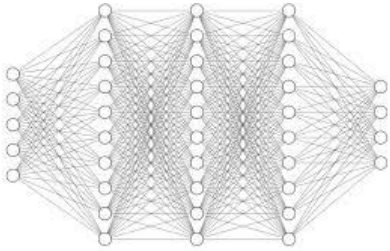
Optimizer dan *Momentum Optimizer* (Dubey *et al*., 2020).

e. *Fully Connected Layer*

Dalam metode *Convolutional Neural Network* (CNN), terdapat dua jenis lapisan utama, yaitu *convolution layer* dan *pooling layer*. Fungsi utama dari lapisan ini adalah untuk mengekstrak fitur-fitur yang relevan dari gambar input, yang pada dasarnya menjadikan CNN sebagai sebuah alat ekstraktor fitur yang sangat kuat. Namun, untuk mengambil langkah berikutnya dalam proses klasifikasi, diperlukan suatu algoritma yang mampu menginterpretasikan dan mengklasifikasikan fitur-fitur yang telah diekstrak (Basha *et al*., 2020).

Di sinilah *Fully Connected Layer* menjadi peran penting. *Fully Connected Layer* adalah bagian dari jaringan saraf biasa yang mampu mengambil fitur-fitur yang telah diekstrak oleh lapisan sebelumnya dan menggunakannya untuk melakukan klasifikasi. Biasanya, *Fully Connected Layer* digunakan dalam dua lapisan terakhir dari jaringan saraf, sering kali diikuti oleh fungsi *softmax*. Fungsi *softmax* digunakan untuk menghasilkan probabilitas

keluaran, yang memungkinkan kita untuk menginterpretasikan hasil prediksi sebagai probabilitas kelas tertentu. Tujuan dari Fully Connected Layer adalah untuk melakukan transformasi pada dimensi data sehingga data dapat diklasifikasikan secara linear (Xu *et al*., 2019).



Gambar 2. 6 Proses *Fully Conected*

6. *Transfer Learning*

Transfer Learning adalah salah satu teknik dalam *machine learning* dan *deep learning* yang memungkinkan model untuk memanfaatkan pengetahuan yang telah dipelajari dari suatu tugas untuk diterapkan dalam tugas lain yang serupa atau berbeda. Dalam *Transfer Learning*, model yang telah dilatih sebelumnya (*pre-trained model*) menggunakan data yang besar dan beragam, seperti model-model yang telah terlatih pada dataset gambar besar seperti *ImageNet*, dapat digunakan sebagai titik

awal atau dasar untuk mengembangkan model baru (Wang *et al*., 2021).

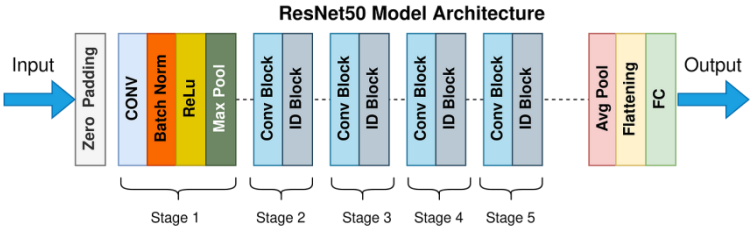
Ada beberapa pendekatan dalam Transfer Learning, termasuk *fine-tuning* dan *feature extraction*. Dalam *fine-tuning*, model *pre-trained* diubah sedikit demi sedikit pada lapisan-lapisan atasnya untuk menyesuaikan dengan tugas baru, sementara lapisan-lapisan bawah yang lebih umum tetap tidak berubah (Ribani & Marengoni, 2019). Di sisi lain, dalam *feature extraction*, model *pre-trained* hanya digunakan sebagai ekstraktor fitur, di mana lapisan-lapisan atasnya dihapus, dan model tersebut digunakan untuk menghasilkan fitur-fitur yang kemudian dijadikan input untuk model klasifikasi atau regresi yang baru (Imoto *et al*., 2019). *Transfer Learning* telah terbukti sangat efektif dalam berbagai tugas seperti pengenalan gambar, pemrosesan bahasa alami dan menjadi alat yang penting dalam mempercepat perkembangan aplikasi kecerdasan buatan.

7. *ResNet-50*

ResNet-50 adalah salah satu arsitektur neural network yang telah terbukti sangat efektif dalam tugas pengenalan gambar dan pemrosesan visual. Salah satu karakteristik utama dari arsitektur *ResNet-50* adalah penggunaan blok

residu (*residual block*). Blok residu ini memungkinkan model untuk melatih jaringan yang lebih dalam (*deep*) tanpa mengalami masalah degradasi kinerja. Dalam blok residu, ada *shortcut* yang mengizinkan aliran informasi langsung dari lapisan masukan (*input layer*) ke lapisan-lapisan di dalam blok tersebut (Kulsum & Cherid, 2023).

ResNet-50 memiliki arsitektur yang cukup mendalam dengan total 50 lapisan. Model ini terdiri dari beberapa blok residu dengan konvolusi 3x3 dan jumlah filter yang meningkat secara bertahap. Blok-blok tersebut dilapis dengan lapisan-lapisan konvolusi, normalisasi batch, dan fungsi aktivasi ReLU. Di bagian akhir arsitektur, terdapat lapisan-lapisan fully connected untuk klasifikasi (Asma-Ull *et al*., 2023).



Gambar 2. 7 Arsitekur ResNet50

8. *TensorFlow*

TensorFlow adalah sebuah perangkat lunak sumber terbuka yang digunakan untuk mengembangkan dan

melatih model *Machine learning* dan *neural netowrk*. Dikembangkan oleh Google Brain Team, *TensorFlow* telah menjadi salah satu *framework* yang paling populer dalam bidang kecerdasan buatan (*Artificial Intelligence*) dan pembelajaran mesin (*Machine Learning*) (Sanchez *et al*., 2020).

Graf komputasi dalam *TensorFlow* terdiri dari *node* yang mewakili operasi matematika dan *edge* yang mewakili aliran data antara *node-node* tersebut. Ini memungkinkan para pengguna untuk dengan mudah mendefinisikan, melatih, dan menerapkan model *Machine learning* dengan lebih fleksibel. Selain itu, *TensorFlow* memiliki berbagai alat dan pustaka tambahan yang mendukung tugas-tugas seperti pengolahan gambar, pemrosesan bahasa alami, dan analisis data (Grattarola & Alippi, 2021).

9. Evaluasi Model

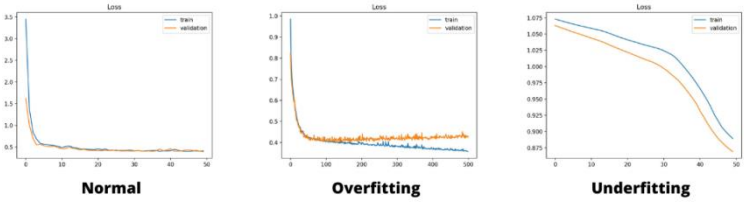
a. *Learning Curve*

Learning Curve adalah grafik yang menunjukkan bagaimana kinerja model berubah seiring dengan meningkatnya jumlah data yang digunakan untuk melatih model tersebut (Viering & Loog, 2023). *Learning Curve* berguna untuk mengidentifikasi dan

menganalisis data yang mengalami *overfitting* ataupun *underfitting*.

Overfitting adalah suatu kondisi di mana model machine learning tidak dapat mempelajari pola data pelatihan yang digunakan untuk melatihnya. Akibatnya, model tersebut cenderung tampil dengan sangat baik pada data pelatihan, tetapi kinerjanya buruk pada data baru yang tidak pernah dilihat sebelumnya (Mutasa *et al*., 2020).

Underfitting adalah kondisi yang terjadi ketika model *machine learning* terlalu sederhana untuk memahami kompleksitas data yang digunakan untuk pelatihan. Dalam *underfitting*, model tidak mampu menangkap pola yang ada dalam data pelatihan dengan baik, sehingga kinerjanya buruk baik pada data pelatihan maupun data uji atau validasi (H. Zhang *et al*., 2019).



Gambar 2. 8 *Learning Curve*

Dalam Gambar 2.8, *Overfitting* biasanya terdeteksi ketika kita melihat bahwa kurva *loss* pada data pelatihan terus menurun seiring berjalannya waktu, tetapi pada saat yang sama, kurva *loss* pada data uji mulai datar atau bahkan meningkat setelah beberapa epoch. Hal ini mengindikasikan bahwa model telah terlalu keras dalam mempelajari pola yang ada dalam data pelatihan, sehingga kesulitan dalam beradaptasi dengan data uji atau melakukan generalisasi dengan baik.

Kemudian, kita dapat mengidentifikasi *underfitting* ketika melihat bahwa kurva *loss* pada data pelatihan tidak menunjukkan penurunan yang signifikan setelah beberapa iterasi atau epoch, atau bahkan ketika kurva *loss* pada data uji justru berada pada tingkat yang lebih rendah daripada kurva *loss* pada data pelatihan. Hal ini mengindikasikan bahwa model tidak mampu menangkap dengan baik pola yang ada dalam data, sehingga tidak dapat melakukan prediksi yang akurat baik pada data pelatihan maupun data uji.

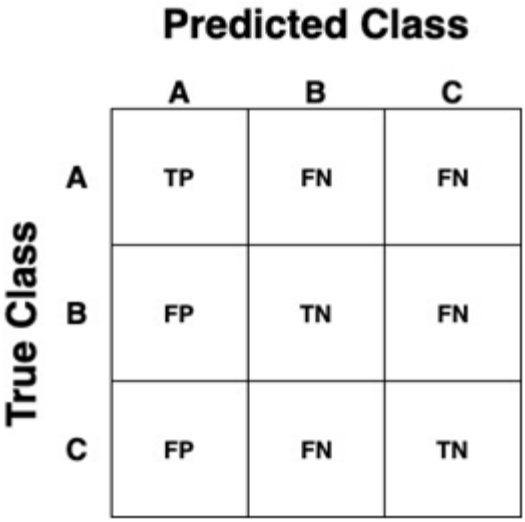
Sedangkan dalam situasi normal, kurva *loss* pada data uji seharusnya menunjukkan kestabilan

atau penurunan seiring berjalannya iterasi atau epoch, mengindikasikan bahwa model tidak terlalu rumit atau terlalu cocok dengan data pelatihan. Ini menjadi tanda bahwa model memiliki kemampuan untuk melakukan prediksi yang akurat pada data baru yang belum pernah dilihat sebelumnya.

b. *Confusion Matrix*

Confusion Matrix sering disebut juga dengan *error matrix* adalah alat penting untuk mengukur performa sebuah model klasifikasi. Dalam analisis performa model, kita sering mengandalkan parameter seperti akurasi, *recall*, presisi dan *F1-score* (Ruuska *et al*., 2018).

Konsep ini memungkinkan kita untuk mengevaluasi secara lebih rinci seberapa baik atau buruk model tersebut berkinerja. Dalam *confusion matrix* terdapat 4 istilah sebagai representasi hasil proses klasifikasi dengan 4 kombinasi nilai prediksi dan aktual yang berbeda. Keempat istilah tersebut adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN) (Mahfudh & Mustofa, 2019).



Gambar 2. 9 *Confusion Matrix*

Pada Gambar 2.9 , *True Positive* (TP) adalah jumlah data yang benar-benar positif dan telah diprediksi dengan benar sebagai positif oleh model, *False Positive* (FP) adalah jumlah data yang sebenarnya negatif tetapi telah salah diprediksi sebagai positif oleh model, *False Negative* (FN) adalah jumlah data yang sebenarnya positif tetapi telah salah diprediksi sebagai negatif oleh model dan *True Negative* (TN) adalah jumlah data yang benar-benar negatif dan telah diprediksi dengan benar sebagai negatif oleh model (Kulkarni *et al*., 2020).

Dengan menggunakan informasi ini, *confusion matrix* memungkinkan kita untuk menghitung berbagai metrik evaluasi performa model seperti akurasi, presisi, *recall* dan *f1-score* (Luque *et al.*, 2019).

Akurasi (*Accuracy*) adalah rasio dari jumlah prediksi yang benar (TP dan TN) terhadap jumlah total sampel. Akurasi mengukur sejauh mana model mampu mengklasifikasikan dengan benar (Kulkarni *et al.*, 2020). Berikut ini rumus menetukan nilai akurasi berdasarkan persamaan 2.1.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \qquad (2.1)$$

Presisi (*Precision*) adalah rasio dari TP terhadap total sampel yang diprediksi positif (TP + FP). Presisi mengukur sejauh mana prediksi positif model adalah benar (Kulkarni *et al.*, 2020). Berikut ini rumus menetukan nilai *precision* berdasarkan persamaan 2.2.

$$Precision = \frac{TP}{TP + FP} \qquad (2.2)$$

Recall adalah rasio dari TP terhadap total sampel positif (TP + FN). *Recall* mengukur kemampuan model dalam mengidentifikasi seluruh sampel yang sebenarnya positif (Kulkarni *et al.*, 2020). Berikut ini rumus menetukan nilai *recall* berdasarkan persamaan 2.3.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

F1-score adalah ukuran gabungan yang mempertimbangkan baik *recall* maupun presisi. Ini berguna ketika ingin mencari keseimbangan antara kedua metrik ini (Kulkarni *et al*., 2020). Berikut ini rumus menetukan nilai *f1-score* berdasarkan persamaan 2.4.

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.4}$$

B. Kajian Penelitian yang Relevan

Berikut ini adalah tabel penelitian terkait yang berisi kumpulan jurnal tentang penelitian terdahulu yang berkaitan dengan penelitian pada tugas akhir kali ini.

Tabel 2. 1 Penelitian Terkait

No	Judul	Nama & Tahun Penelitian	Metode	Hasil
1	*Diagnosis and Detection of Alzheimer's Disease Using Learning Algorithm*	Gargi Pant Shukla, Santosh Kumar, Saroj Kumar Pandey, Rohit Agarwal, Neeraj Varshney, Ankit Kumar (2023)	Random forest, XGBoost, dan Convolution Neural Networks (CNN)	Berdasarkan hasil penlitian didapatkan tingkat akurasi yang mengesankan sebesar 97,57% dan sensitivitas 97,60%
2	*Classification Of Alzheimer's Disease from MRI Data Using an Ensemble of Hybrid Deep Convolutional Neural Networks*	Emimal Jabason, M. Omair Ahmad, M.N.S. Swamy (2019)	*Convolutional Neural Network (CNN)* menggunakan arsitektur *DenseNet121, DenseNet169, DenseNet201*	Hasil terbaik untuk *10 - fold cross validation* akurasi untuk arsitektur *DenseNet121* 88.32%, *DenseNet169* 92.23%, *DenseNet201*

				95.23%
3	*Alzheimer's Disease Classification From 2D MRI Brain Scans Using Convolutional Neural Networks*	Rubaiyat Alim Hridhee, Biddut Bhowmik, Quazi Delwar Hossain (2023)	*Convolutional Neural Network (CNN)* menggunakan arsitektur *VGG16 dan Xception*	Berdasarkan hasil penelitian didapatkan akurasi sebesar 94,77% dan *F1-score* 94,81%
4	*DEMNET: A Deep Learning Model for Early Diagnosis of Alzheimer Diseases and Dementia From MR Images*	Suriya Murugan, Chandran Venkatesan, M. G. Sumithra, Xiao-Zhi Gao, B. Elakkiya, M. Akila, S. Manoharan (2021)	*Convolutional Neural Network (CNN)* menggunakan arsitektur DEMNET	Berdasarkan hasil penelitian didapatkan akurasi sebesar 95,23%

Penelitian mengenai klasifikasi penyakit Alzheimer pada citra MRI otak dengan memanfaatkan model arsitektur *ResNet50* dan menggunakan *framework TensorFlow* membawa inovasi dibandingkan dengan studi-

studi sebelumnya. Penelitian ini memanfaatkan model arsitektur *ResNet50* yang dikenal sebagai salah satu arsitektur jaringan saraf tiruan yang memiliki kemampuan dalam memahami fitu-fitur yang kompleks pada citra. Dibandingkan dengan metode sebelumnya, penggunaan framework *Tensorflow* memberikan fleksibilitas dalam pengembangan dan penyesuaian model, memungkinkan penelitian untuk lebih terfokus pada pengoptimalan model yang disesuaikan dengan data citra MRI otak yang spesifik.

BAB III
METODOLOGI PENELITIAN

A. Jenis Penelitian

Penelitian ini merupakan penelitian eksperimental dengan pendekatan kuantitatif. Penelitian ini bertujuan untuk mengembangkan dan menguji model *Machine Learning* berbasis *Deep Learning* menggunakan *Tensorflow* untuk mengklasifikasikan citra MRI otak sebagai pasien dengan *Alzheimer* atau pasien yang sehat.

B. Jenis dan Sumber Data

Pada tahap ini, langkah pertama adalah menghimpun data gambar MRI dari *Alzheimer's Disease Neuroimaging Initiative (ADNI)*. Data yang digunakan bersifat data sekunder karena dapat diakses melalui internet (https://adni.loni.usc.edu/). *Alzheimer's Disease Neuroimaging Initiative (ADNI)* menyediakan data yang terorganisir dan sudah diolah dengan baik. Dataset ADNI mencakup gambar MRI T1 dan T2 dari pemindai MRI 1.5T dan 3T. Dataset yang digunakan adalah ADNI-1 berasal dari pemindai Tesla 1.5T, yang telah diproses sebelumnya

dengan teknologi *Magnetization Prepared-RApid Gradient Echo* (MP-RAGE).

Dalam pengumpulan ini, berhasil diperoleh 3000 gambar yang dikelompokkan menjadi 3 kelas utama. Rinciannya dapat dilihat pada Tabel 3. 1.

Tabel 3. 1 Jumlah Dataset

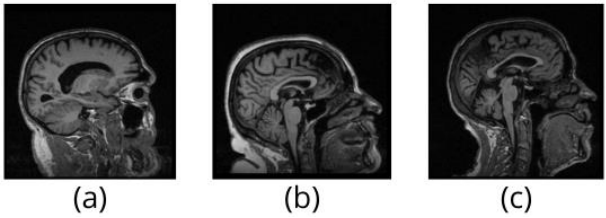
Data	***Alzheimer Disease* (AD)**	***Mild Cognitive Impairment* (MCI)**	***Cognitive Normal* (CN)**
Jumlah Data	**1000**	**1000**	**1000**
Jenis Kelamin	**510 L/490 P**	**475 L/525 P**	**500 L/500 P**

Pada tabel 3.1, menyajikan distribusi data untuk tiga kelompok subjek dalam konteks penelitian *Alzheimer Disease* (AD), *Mild Cognitive Impairment* (MCI), dan *Cognitive Normal* (CN). Setiap kelompok subjek terdiri dari 1000 individu. Informasi tambahan disediakan mengenai jenis kelamin subjek dalam masing-masing kelompok.

Dalam kelas AD, terdapat 1000 subjek dengan proporsi 51% laki-laki (510 individu) dan 49% perempuan (490 individu). Sebaliknya, pada kelas MCI, distribusi jenis kelamin adalah 47.5% laki-laki (475

individu) dan 52.5% perempuan (525 individu). Sedangkan pada kelas CN, terdapat 50% laki-laki (500 individu) dan 50% perempuan (500 individu). Gambar-gambar ini memiliki ukuran yang beragam dengan format *Neuroimaging Informatics Technology Initiative* (NiFTI).

Berikut contoh sampel gambar yang diambil pada setiap kelas.



Gambar 3. 1 Sampel Gambar dari dataset: (a) normal (CN), (b) MCI, dan (c) AD

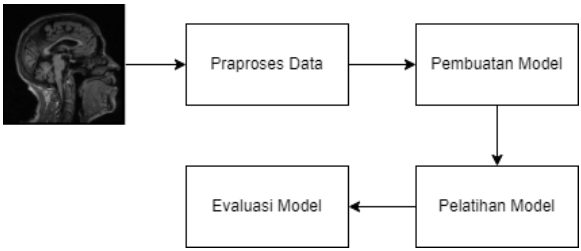
C. Metode

Dalam penelitian ini, pemilihan *Convolutional Neural Network (CNN)* sebagai metode utama dalam analisis data penelitian ini didasarkan pada keunggulan CNN dalam menangani data gambar, terutama dalam konteks analisis citra medis seperti gambar MRI

Model arsitektur *ResNet50* dipilih karena keunggulannya dalam menangani masalah *vanishing gradient* dan *exploding gradient*, yang sering terjadi pada jaringan yang lebih dalam. Oleh karena itu, *ResNet50*

merupakan pilihan yang baik untuk analisis data yang melibatkan dataset kompleks seperti gambar MRI dengan banyak lapisan informasi.

Berikut adalah tahapan-tahapan yang akan dijalani selama penelitian ini:



Gambar 3. 2 Diagram Blok Metodologi Penelitian

Di bawah ini adalah penjelasan yang terkandung dalam Gambar 3.2 yang menggambarkan tahapan metodologi penelitian:

1. Praproses Data

a. Perubahan Format dari Nifti ke JPG

Proses perubahan file NIfTI (*Neuroimaging Informatics Technology Initiative*) ke format JPEG (*Joint Photographic Experts Group*) melibatkan beberapa tahap yang penting. Pertama, file NIfTI, yang biasanya digunakan untuk menyimpan data citra medis seperti MRI, harus dibaca dan diinterpretasi dengan menggunakan pustaka seperti

nibabel di *python*. Selanjutnya, nilai piksel dalam citra NIfTI, yang sering kali direpresentasikan dengan kedalaman bit yang tinggi untuk mempertahankan detail yang diperlukan untuk analisis medis, perlu dinormalisasi atau diubah agar sesuai dengan format yang diterima oleh JPEG, yang biasanya menggunakan 8-bit per kanal untuk warna (24-bit total untuk warna RGB). Proses normalisasi ini dapat melibatkan penyesuaian skala nilai piksel atau konversi ke ruang warna yang sesuai.

Setelah normalisasi, citra dapat dikompresi ke format JPEG untuk mengurangi ukuran file dan memudahkan penyimpanan dan pengiriman. Proses kompresi ini dapat menyebabkan hilangnya beberapa detail dalam citra. Oleh karena itu, perlu mempertimbangkan tingkat kompresi yang sesuai untuk memastikan bahwa informasi di dalam citra tidak terlalu terpengaruh. Selain itu, metadata yang mungkin terdapat dalam file NIfTI, seperti informasi tentang pasien atau parameter pencitraan, mungkin hilang atau tidak disertakan dalam file JPEG. Kehilangan metadata ini dapat mengurangi konteks atau informasi yang tersedia untuk interpretasi citra.

Oleh karena itu, saat mengonversi file NIfTI ke format JPEG, perlu dipertimbangkan pengaruhnya terhadap kualitas, ukuran, dan konten informasi yang terkandung dalam citra.

b. Proses Pemilihan Slice Tengah

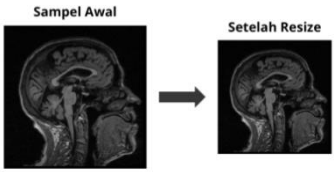
Proses pemilihan slice tengah dalam citra medis, seperti MRI, merupakan tahap penting dalam analisis dan interpretasi data. Citra medis sering kali terdiri dari serangkaian slice atau irisan yang merepresentasikan bagian-bagian yang berbeda dari organ atau jaringan yang dipelajari. Memilih slice tengah dari kumpulan slice ini memungkinkan untuk fokus pada area yang paling representatif atau relevan dari organ atau struktur tertentu. Biasanya, slice tengah dipilih berdasarkan pada sumbu tertentu dari citra, seperti sumbu z atau sumbu slice. Proses ini melibatkan perhitungan jumlah slice dan menentukan posisi slice tengah dalam setiap sumbu.

Pemilihan slice tengah ini dapat membantu mengurangi kemungkinan distorsi atau perubahan posisi yang tidak diinginkan yang mungkin terjadi pada slice di tepi. Selain itu, slice tengah sering digunakan untuk menghasilkan representasi visual

yang paling akurat dan informatif dari organ atau struktur yang diamati.

c. *Resizing* Ukuran Citra MRI

Saat melakukan *input* citra, langkah yang perlu dilakukan adalah mengubah ukuran citra-citra tersebut untuk membuat ukurannya seragam pada semua sampel citra. Hal ini dilakukan untuk memastikan bahwa semua citra memiliki dimensi yang konsisten, sehingga model dapat memproses data dengan benar dan efisien.



Gambar 3. 3 Resizing Ukuran Citra

Pada Gambar 3.3, terjadi proses penyesuaian ukuran citra yang bertujuan untuk menciptakan konsistensi dalam representasi citra. Ketika menghadapi dataset yang memiliki variasi ukuran citra yang signifikan, proses *resizing* menjadi penting untuk menormalkan ukuran citra sehingga menjadi seragam. Hal ini membantu model *Convolutional Neural Network* (CNN) untuk mempelajari pola dan

fitur yang relevan tanpa terpengaruh oleh perbedaan ukuran citra.

Setelah dilakukan proses *resizing*, data citra diubah menjadi *array NumPy* menggunakan fungsi *np.array()*. Dengan mengubahnya menjadi *array NumPy*, data pelatihan dan label dapat digunakan secara efisien dalam proses pelatihan model.

d. Pembagian Data

Setelah melalui tahap praproses data, langkah selanjutnya adalah membagi dataset menjadi dua bagian dengan perbandingan 80:20. Pembagian ini bertujuan untuk mengevaluasi kinerja model secara objektif, memastikan bahwa model yang dibangun dapat menggeneralisasi dengan baik pada data yang belum pernah dilihat sebelumnya. Perbandingan 80:20 memberikan keseimbangan yang baik antara melatih model dengan data yang cukup banyak dan menguji model pada data yang cukup representatif.

Meskipun demikian, untuk melihat dampak dari penambahan data pelatihan, percobaan dengan perbandingan 90:10 juga dilakukan. Penggunaan perbandingan 90:10 dapat memberikan lebih banyak data untuk pelatihan, yang potensial

menghasilkan model yang lebih baik dalam beberapa kasus. Namun, evaluasi pada set pengujian dengan perbandingan 90:10 dapat menjadi kurang andal karena ukurannya yang lebih kecil, sehingga *trade-off* antara pelatihan model yang lebih baik dan evaluasi yang lebih akurat harus dipertimbangkan secara cermat.

Meskipun demikian, yang tetap menjadi fokus utama adalah penggunaan perbandingan 80:20 untuk penelitian ini karena memberikan keseimbangan yang baik antara jumlah data pelatihan dan pengujian.

2. Pembuatan Model

Pada tahap pembuatan model, pendekatan yang akan digunakan adalah *Transfer Learning*, di mana kita akan memanfaatkan model yang telah dilatih sebelumnya, yakni *ResNet50*. Model *ResNet50* ini sudah memiliki kemampuan untuk memahami banyak fitur pada gambar-gambar yang kompleks. Namun, agar model ini sesuai dengan tujuan khusus kita, beberapa *layer* tambahan akan ditambahkan untuk menyesuaikan hasil output yang diinginkan.

3. Pelatihan Model

Setelah pembuatan model dengan memanfaatkan metode *Transfer Learning*, tahap selanjutnya adalah melakukan proses pelatihan (*Training*) pada model tersebut. Proses pelatihan melibatkan dua tahap utama, yaitu tahap *compile* dan *fit*.

a. *Compile*

```
model.compile(loss='categorical_crossentr  py',
        optimizer='Adam',
        metrics = ['accuracy'])
```

Penggunaan *model.compile()* adalah langkah dalam konfigurasi model, yang bertujuan untuk menentukan bagaimana model akan belajar dan dinilai selama proses pelatihan. Pada metode *model.compile()*, terdapat beberapa argumen yang memiliki fungsi masing-masing. Pertama, *loss='categorical_crossentropy'* berperan dalam menetapkan fungsi loss yang akan digunakan selama pelatihan. Dalam konteks ini, digunakan *categorical_crossentropy* karena model ditujukan untuk melakukan klasifikasi pada data dengan lebih dari dua kelas.

Selanjutnya, argumen *optimizer='Adam'* bertanggung jawab untuk memilih algoritma optimisasi yang akan mengatur bagaimana bobot model diperbarui berdasarkan perhitungan *loss*. Di sini, dipilih algoritma yang cukup populer yaitu Adam, yang sering digunakan dalam optimisasi *neural network*. Dalam konteks *metrics=['accuracy']*, argumen ini berperan dalam menentukan metrik evaluasi yang akan digunakan untuk menilai performa model selama proses pelatihan dan saat dilakukan pengujian. Dalam situasi ini, kita telah memilih akurasi sebagai metrik evaluasi yang relevan. Akurasi mengukur persentase klasifikasi yang benar yang dilakukan oleh model, menggambarkan sejauh mana model berhasil dalam mengidentifikasi dan memprediksi data dengan benar.

b. *Fit*

```
history = model.fit(X_train,y_train,validation_split=0.2,
                    epochs =20,
                    verbose=1,
                    batch_size=32,
                    callbacks=[tensorboard, reduce_lr])
```

Dalam proses pengaturan model *machine learning*, terdapat sekumpulan *hyperparameter* yang memiliki peran kunci dalam mengontrol cara model tersebut belajar serta bagaimana parameter internalnya diperbarui selama proses pembelajaran. Salah satu *hyperparameter* yang penting adalah *epoch*, yang memiliki peran menentukan seberapa banyak algoritma pembelajaran akan melalui seluruh dataset pelatihan dalam satu *iterasi*. Nilai *epoch* yang lebih tinggi biasanya dapat meningkatkan tingkat akurasi model, karena model memiliki lebih banyak kesempatan untuk mempelajari pola-pola dalam data. Namun, perlu diingat bahwa peningkatan nilai epoch juga memiliki konsekuensi dalam hal waktu yang dibutuhkan. Dengan kata lain, semakin tinggi nilai *epoch*, semakin lama waktu yang dibutuhkan untuk menyelesaikan pelatihan model.

Selain *epoch*, terdapat *hyperparameter* lain yaitu *learning rate*. *Learning rate* adalah *hyperparameter* yang menentukan seberapa cepat atau lambat model belajar dari data selama proses pelatihan. Ketika *learning rate* rendah, proses pembelajaran akan

berjalan lebih lambat, tetapi model cenderung mencapai hasil yang lebih baik karena perbaruan parameter dilakukan secara perlahan-lahan dan stabil. Di sisi lain, jika *learning rate* terlalu tinggi, proses pembelajaran dapat berjalan lebih cepat, tetapi model dapat mengalami kesulitan dalam konvergensi atau bahkan melompati nilai minimum yang diinginkan dalam fungsi *loss*.

Selanjutnya, terdapat *hyperparameter* yang disebut *batch size* dalam model *machine learning*, yang memainkan peran penting dalam proses pelatihan. *Batch size* menentukan jumlah sampel data yang akan diserahkan ke model pada setiap *iterasi* pembelajaran.

Dalam pelatihan model ini, telah ditetapkan nilai *epoch* sebanyak 20, yang akan menentukan berapa kali algoritma pembelajaran akan melalui keseluruhan dataset pelatihan dalam satu *iterasi*. Nilai *learning rate* dan *batch size* diatur menggunakan nilai default, yaitu 0.001 dan 32. Selain itu, dalam metode *fit*, juga telah diintegrasikan penggunaan *callbacks*. *Callbacks* memungkinkan kita untuk memantau, mengontrol, atau melakukan

tindakan khusus pada model selama berlangsungnya proses pelatihan.

Setelah proses pelatihan model selesai, kita akan memperoleh nilai *loss* dan akurasi yang menggambarkan kinerja model terhadap data yang telah digunakan selama pelatihan. Dengan menggunakan hasil ini, model dapat dievaluasi secara menyeluruh untuk mengukur sejauh mana kinerja dan kemampuannya dalam melakukan tugasnya.

4. Evaluasi Model

Tahapan ini berisi evaluasi model pada tugas klasifikasi multikelas dengan tiga kelas, yaitu *Alzheimer Disease* (AD), *Cognitive Normal* (CN) dan *Mild Cognitive Impairment* (MCI), melibatkan penggunaan *classification report* dan *confusion matrix*. Dalam konteks tiga kelas AD, CN, dan MCI, *classification report* akan memberikan ringkasan yang jelas tentang seberapa baik model dapat membedakan dan mengklasifikasikan masing-masing kelas. Sedangkan *confusion matrix* akan menggambarkan seberapa baik model dapat mengklasifikasikan setiap kelas dan sejauh mana terdapat kebingungan antara kelas-kelas tersebut.

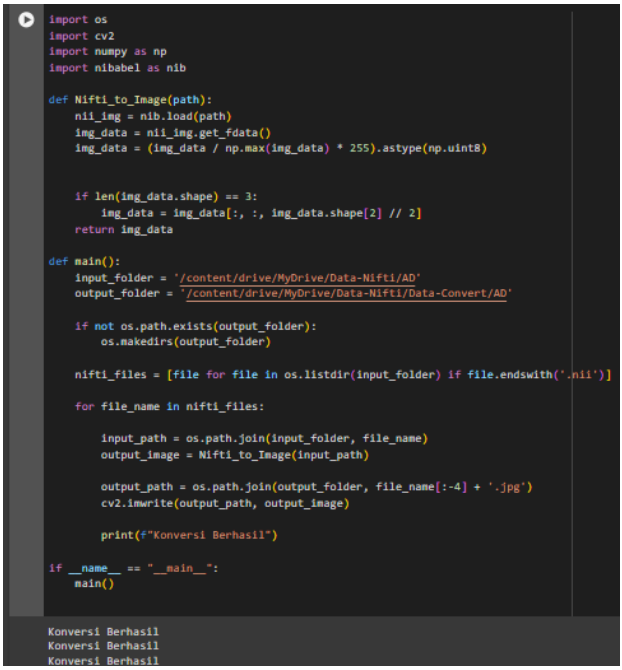
BAB IV
HASIL DAN PEMBAHASAN

Dalam penelitian ini, peneliti mengaplikasikan CNN untuk mengklasifikasikan gambar-gambar terkait Alzheimer menggunakan TensorFlow. Proses utama dalam pembuatan model klasifikasi metode CNN adalah perancangan klasifikasi. Prinsip utamanya adalah melalui pelatihan pada CNN untuk menciptakan model yang optimal. Proses pelatihan CNN melibatkan dua set data, yaitu data latih dan data uji. Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk menguji dan memvalidasi performa model.

A. Praproses Data

1. Mengkonversi Citra MRI

Sebelum data gambar dari file NIfTI diproses, langkah konversi perlu dilakukan untuk mempersiapkan data tersebut agar sesuai dengan kebutuhan proses selanjutnya. Proses konversi ini terutama bertujuan untuk mengubah skala intensitas nilai piksel dalam data gambar sehingga dapat diinterpretasikan dengan lebih mudah.



```
import os
import cv2
import numpy as np
import nibabel as nib

def Nifti_to_Image(path):
    nii_img = nib.load(path)
    img_data = nii_img.get_fdata()
    img_data = (img_data / np.max(img_data) * 255).astype(np.uint8)


    if len(img_data.shape) == 3:
        img_data = img_data[:, :, img_data.shape[2] // 2]
    return img_data

def main():
    input_folder = '/content/drive/MyDrive/Data-Nifti/AD'
    output_folder = '/content/drive/MyDrive/Data-Nifti/Data-Convert/AD'

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

    nifti_files = [file for file in os.listdir(input_folder) if file.endswith('.nii')]

    for file_name in nifti_files:

        input_path = os.path.join(input_folder, file_name)
        output_image = Nifti_to_Image(input_path)

        output_path = os.path.join(output_folder, file_name[:-4] + '.jpg')
        cv2.imwrite(output_path, output_image)

        print(f"Konversi Berhasil")

if __name__ == "__main__":
    main()
```

```
Konversi Berhasil
Konversi Berhasil
Konversi Berhasil
```

Gambar 4. 1 Proses konversi data ke format .jpg

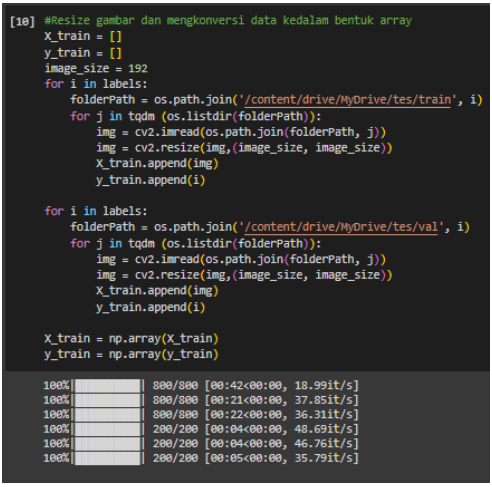
Dalam Gambar 4.1 diatas, langkah konversi melibatkan transformasi data ke dalam tipe data *uint8*. Data gambar awalnya memiliki tipe data *float 32-bit big-endian*(>f4) dengan rentang nilai yang luas. Untuk tujuan visualisasi atau analisis lebih lanjut, data tersebut perlu dikonversi ke tipe data *uint8*, yang biasanya digunakan dalam representasi gambar digital. Tipe data *uint8* memiliki rentang

nilai antara 0 hingga 255, yang sesuai dengan skala intensitas piksel dalam gambar digital.

Selain itu, jika data gambar memiliki dimensi yang lebih dari dua, seperti dalam kasus gambar medis 3D, langkah tambahan perlu dilakukan untuk memilih *slice* yang sesuai untuk diproses. Dalam Gambar 4.1, dilakukan pemilihan *slice* bagian tengah dari data gambar 3D untuk diproses lebih lanjut.

2. *Resizing* Ukuran Citra MRI

Dalam proses ini, memungkinkan peneliti untuk menyusun dataset pelatihan dengan ukuran yang seragam dan format yang sesuai.



```
[10] #Resize gambar dan mengkonversi data kedalam bentuk array
     X_train = []
     y_train = []
     image_size = 192
     for i in labels:
         folderPath = os.path.join('/content/drive/MyDrive/tes/train', i)
         for j in tqdm (os.listdir(folderPath)):
             img = cv2.imread(os.path.join(folderPath, j))
             img = cv2.resize(img,(image_size, image_size))
             X_train.append(img)
             y_train.append(i)

     for i in labels:
         folderPath = os.path.join('/content/drive/MyDrive/tes/val', i)
         for j in tqdm (os.listdir(folderPath)):
             img = cv2.imread(os.path.join(folderPath, j))
             img = cv2.resize(img,(image_size, image_size))
             X_train.append(img)
             y_train.append(i)

     X_train = np.array(X_train)
     y_train = np.array(y_train)

     100%|██████████| 800/800 [00:42<00:00, 18.99it/s]
     100%|██████████| 800/800 [00:21<00:00, 37.85it/s]
     100%|██████████| 800/800 [00:22<00:00, 36.31it/s]
     100%|██████████| 200/200 [00:04<00:00, 48.69it/s]
     100%|██████████| 200/200 [00:04<00:00, 46.76it/s]
     100%|██████████| 200/200 [00:05<00:00, 35.79it/s]
```

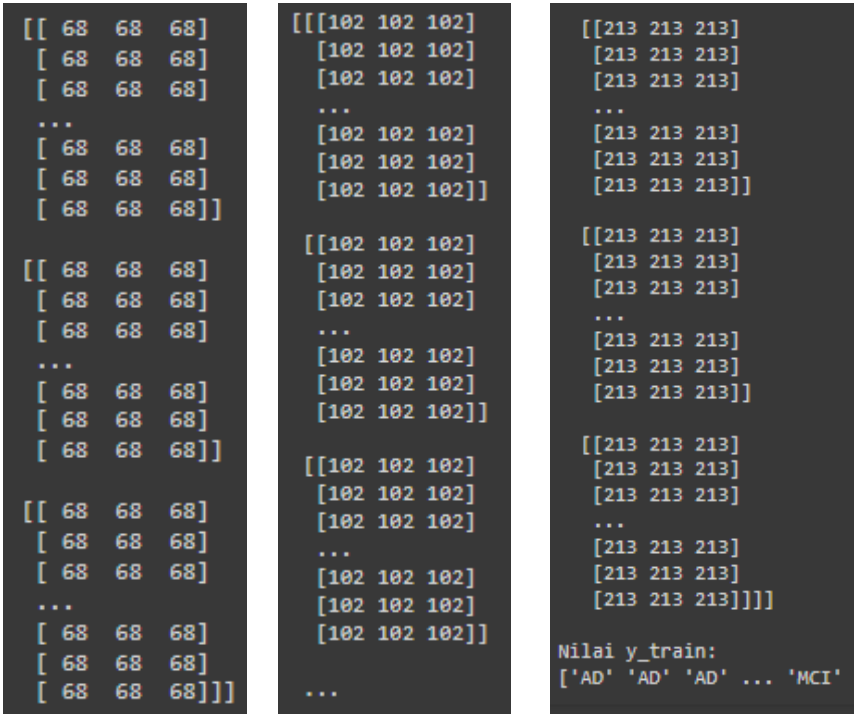
Gambar 4. 2 Proses *resizing* dan mengubah data ke bentuk array

Pada Gambar 4.2, langkah-langkah ini dilakukan untuk menyiapkan dataset pelatihan (*X_train*) dan labelnya (*y_train*) dalam bentuk yang dapat digunakan untuk melatih model. Proses dimulai dengan iterasi melalui setiap label dalam dataset, di mana setiap label mewakili kelas atau kategori dari gambar yang akan diproses.

Kemudian, untuk setiap gambar dalam setiap kelas, dilakukan proses pembacaan gambar menggunakan *OpenCV* (*cv2.imread*) dari lokasi folder yang sesuai. Setelah gambar dibaca, dilakukan *resizing* menggunakan metode *cv2.resize* untuk menyesuaikan ukuran gambar menjadi ukuran yang ditentukan, dalam hal ini, ukuran gambar menjadi 192x192 piksel. Setelah gambar diubah, gambar tersebut dimasukkan ke dalam list *X_train* sebagai elemen *array* gambar. Label dari gambar yang sesuai juga dimasukkan ke dalam list *y_train*. Proses ini diulang untuk setiap gambar dalam setiap kelas pada dataset pelatihan dan dataset validasi.

Setelah semua gambar telah diubah dan labelnya ditambahkan, list *X_train* dan *y_train* diubah menjadi *array numpy* menggunakan *np.array()*.

Dengan mengubah data ke dalam bentuk *array numpy*, data menjadi lebih mudah diolah dan dapat digunakan secara langsung dalam proses pelatihan model.



Gambar 4. 3 Nilai array pada gambar

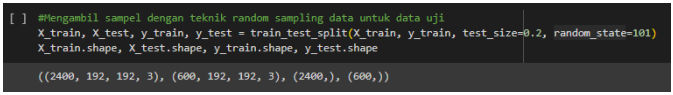
Dalam keluaran tersebut, *X_train* adalah variabel yang berisi data gambar yang akan digunakan untuk pelatihan model, sementara *y_train* berisi label kelas yang sesuai dengan setiap data

gambar dalam X_train. Variabel X_train berisi array multi-dimensi yang mewakili intensitas piksel dari setiap gambar. Setiap gambar direpresentasikan sebagai matriks 2D atau 3D, tergantung pada apakah gambar adalah citra grayscale atau citra berwarna.

Di sisi lain, *y_train* berisi array satu dimensi yang berisi label kelas untuk setiap gambar dalam *X_train*. Label kelas ini menunjukkan kategori atau kondisi yang sesuai dengan setiap gambar, dalam hal ini, label kelas berupa AD, CN, dan MCI.

3. Pembagian Data

Dalam tahap ini, dataset di bagi menjadi data latih dan data uji. Bisa dilihat pada Gambar 4.4.



Gambar 4. 4 Proses Pembagian Data

Pada Gambar 4.4, terlihat bahwa fungsi *train_test_split* dari pustaka *sklearn.model_selection* digunakan untuk membagi dataset menjadi data latih dan data uji. Dalam proses ini, dataset awal yang terdiri dari *X_train* dan *y_train* akan dibagi menjadi dua bagian, yaitu data latih baru (*X_train*

dan *y_train*) dan data uji (*X_test* dan *y_test*). Proporsi pembagian antara data latih dan data uji ditentukan oleh parameter *test_size*, yang disetel ke 0.2, menunjukkan bahwa 20% dari dataset akan dialokasikan sebagai data uji. Selain itu, parameter *random_state* digunakan untuk mengontrol pengacakan data saat pembagian. Penggunaan nilai *random_state* yang sama akan menghasilkan pembagian yang sama setiap kali kode dieksekusi.

Hasil cetak *shape* dari *X_train*, *X_test*, *y_train*, dan *y_test* menunjukkan bahwa proses pembagian dataset telah berhasil dilakukan sesuai dengan proporsi yang diinginkan. *X_train* dan *y_train* memiliki jumlah sampel yang lebih besar daripada *X_test* dan *y_test*, sesuai dengan proporsi pembagian yang telah ditetapkan sebelumnya. Dengan demikian, proses ini memastikan bahwa dataset telah dibagi dengan benar untuk digunakan dalam proses pelatihan dan evaluasi model secara efektif.

B. Pembuatan Model

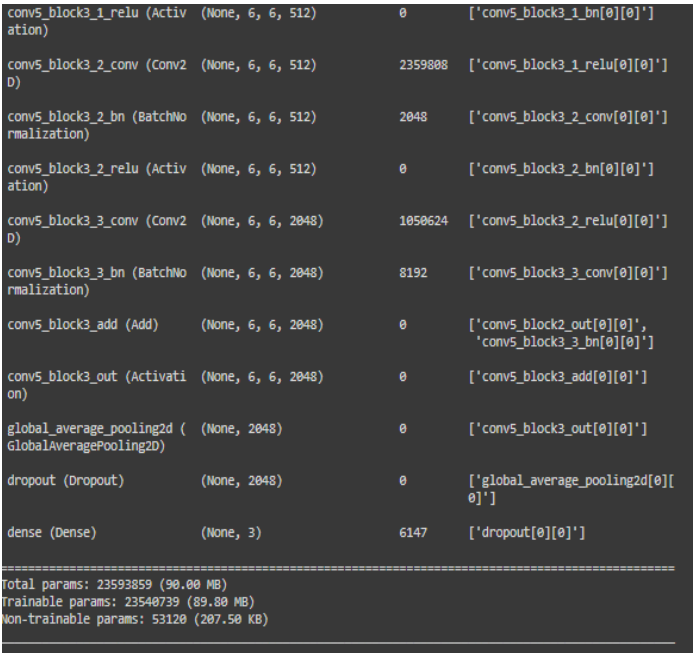
Dalam pembuatan model, peneliti menerapkan metode *transfer learning* dengan memanfaatkan model yang sudah terlatih menggunakan dataset *ImageNet*

untuk diterapkan pada dataset baru. Seperti yang telah diuraikan sebelumnya, salah satu model yang akan kami gunakan dalam penelitian ini adalah *ResNet50*. Dalam pembuatan model ini akan ditambahkan beberapa layer untuk menyesuaikan data yang telah disiapkan. Ringkasan model dapat dilihat pada Gambar 4. 5.

```
#Proses inisialisasi model ResNet50
base_model = ResNet50(weights='imagenet',
                      include_top=False,
                      input_shape=(image_size,image_size,3)
                      )

# Membuat top layers baru di atas basis model
model = base_model.output
model = tf.keras.layers.GlobalAveragePooling2D()(model)
model = tf.keras.layers.Dropout(rate=0.5)(model)
model = tf.keras.layers.Dense(3, activation='softmax')(model)
model = tf.keras.models.Model(inputs=base_model.input, outputs=model)
model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_2 (InputLayer)        [(None, 192, 192, 3)]     0         []

 conv1_pad (ZeroPadding2D)   (None, 198, 198, 3)       0         ['input_2[0][0]']

 conv1_conv (Conv2D)         (None, 96, 96, 64)        9472      ['conv1_pad[0][0]']

 conv1_bn (BatchNormalizati  (None, 96, 96, 64)        256       ['conv1_conv[0][0]']
 on)

 conv1_relu (Activation)     (None, 96, 96, 64)        0         ['conv1_bn[0][0]']

 pool1_pad (ZeroPadding2D)   (None, 98, 98, 64)        0         ['conv1_relu[0][0]']

 pool1_pool (MaxPooling2D)   (None, 48, 48, 64)        0         ['pool1_pad[0][0]']
```

Gambar 4. 5 Proses Pembuatan Model



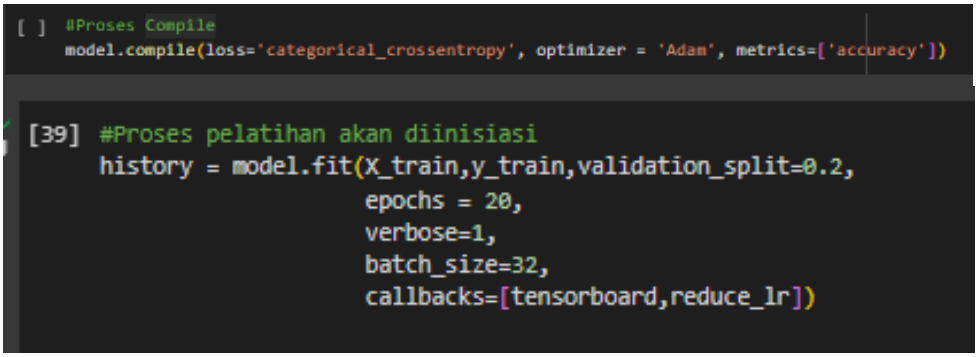
Gambar 4. 6 Lanjutan Proses Pembuatan Model

Pada Gambar 4.5 dan 4.6, terlihat bahwa setelah *layer input*, model menggunakan *ResNet50* sebagai dasar model, yang kemudian ditambahkan dengan 3 layer tambahan, yaitu *GlobalAveragePooling2D*, *Dropout*, dan *Dense*. Total parameter yang digunakan dalam model ini adalah 23.593.859, di mana sebanyak 23.540.739 dianggap sebagai parameter yang dapat dilatih (*trainable parameters*), sementara 53.120 parameter sisanya adalah *non-trainable parameters*.

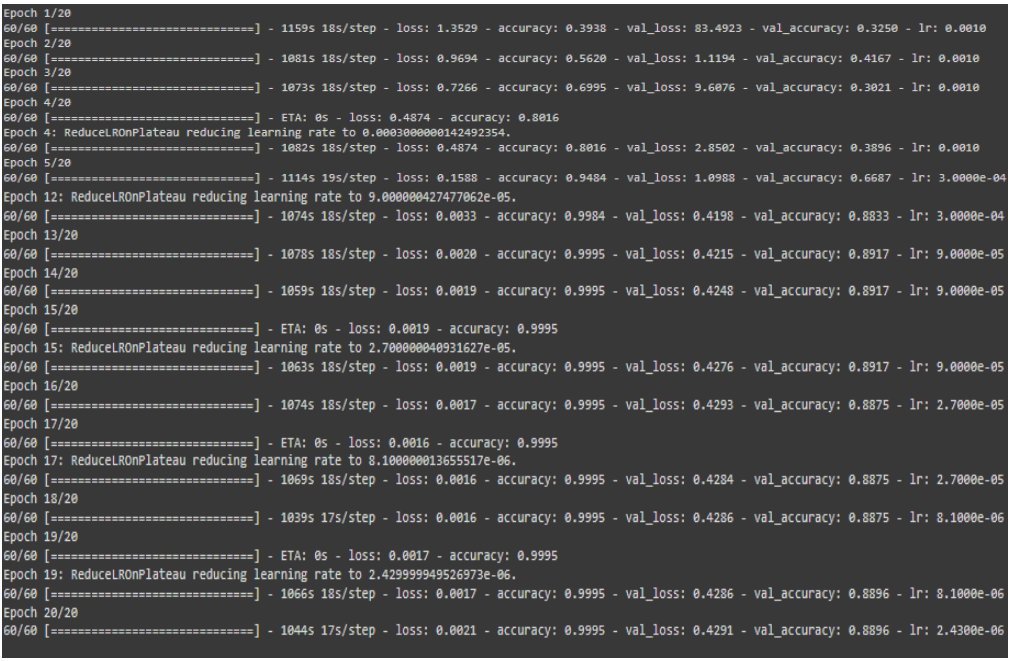
Layer output Dense memiliki bentuk *output* (None, 3), yang menghasilkan output dengan 3 kelas yang diaktifkan menggunakan fungsi *softmax*. Dalam struktur ketiga layer tersebut, output dari *ResNet50* pertama-tama diproses melalui *GlobalAveragePooling2D* untuk mengurangi jumlah parameter dalam model. Kemudian, hasil dari *GlobalAveragePooling2D* dikenai *Dropout* untuk mencegah terjadinya *overfitting*. Terakhir, hasil dari *Dropout* disambungkan dengan layer *Dense* untuk menghasilkan output dengan 3 kelas.

C. Pelatihan Model

Dalam pelatihan model, Proses pelatihan melibatkan dua tahap utama, yaitu tahap *compile* dan *fit*. Bisa dilihat pada Gambar 4.7.



Gambar 4. 7 Proses Pelatihan Model



Gambar 4. 8 Hasil Proses Pelatihan Model

Pada Gambar 4.8, terdapat 20 *epoch* yang dilakukan dengan menggunakan dataset yang telah disiapkan. Pada *epoch* pertama, waktu yang diperlukan untuk menyelesaikan satu *epoch* adalah sekitar 1159 detik atau sekitar 19 menit 19 detik. Pada akhir epoch pertama, nilai *val_loss* tercatat sebesar 83.4923 dan *val_accuracy* sebesar 0.3250.

Selanjutnya, pada *epoch* kedua, waktu yang diperlukan untuk menyelesaikan satu epoch pada epoch

kedua adalah sekitar 1081 detik atau sekitar 18 menit 1 detik. Pada akhir *epoch* kedua, terjadi peningkatan dalam performa model, dengan nilai *val_loss* menurun menjadi 1.1194 dan akurasi meningkat menjadi 0.4167.

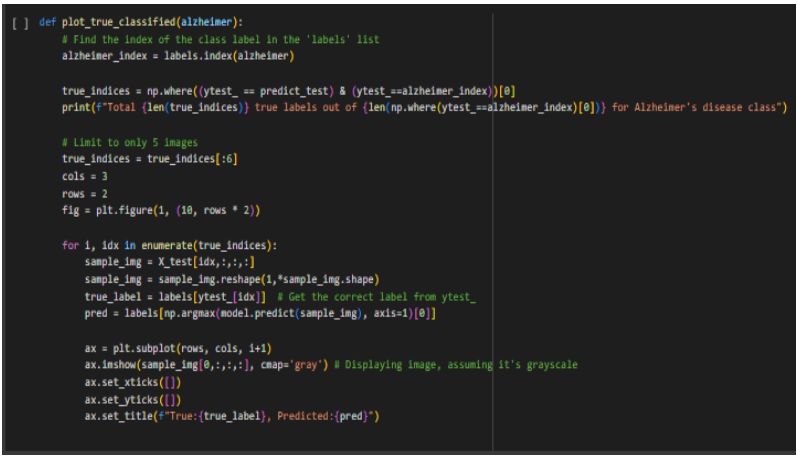
Pada *epoch* ketiga, akurasi turun menjadi 0.3021, *val_loss* melonjak tajam menjadi 9.6076. Namun, pada *epoch* keempat, setelah penyesuaian *learning rate*, terjadi penurunan yang signifikan dalam *val_loss*, dan nilai *val_accuracy* mengalami peningkatan sekitar 0.3896. Proses ini menandai fase adaptasi model terhadap perubahan dalam struktur dan pola data.

Selanjutnya, pada *epoch* kelima, nilai *val_accuracy* naik menjadi sekitar 0.6687, dengan *val_loss* yang menurun signifikan menjadi 1.0988. Demikianlah seterusnya, setiap *epoch* dilakukan dengan iterasi yang sama, di mana model mempelajari pola dari dataset yang diberikan. Proses ini berlanjut hingga *epoch* ke-20, di mana pada akhirnya, waktu yang diperlukan untuk satu epoch adalah sekitar 1044 detik atau sekitar 17 menit 24 detik. Pada akhir *epoch* ke-20, model mencapai nilai *val_loss* sebesar 0.4291 dan nilai *val_accuracy* sebesar 0.8896. Dengan demikian, proses pelatihan ini menunjukkan peningkatan yang signifikan dalam

performa model dari *epoch* ke *epoch*, dengan waktu yang berbeda-beda untuk menyelesaikan setiap siklus pelatihan.

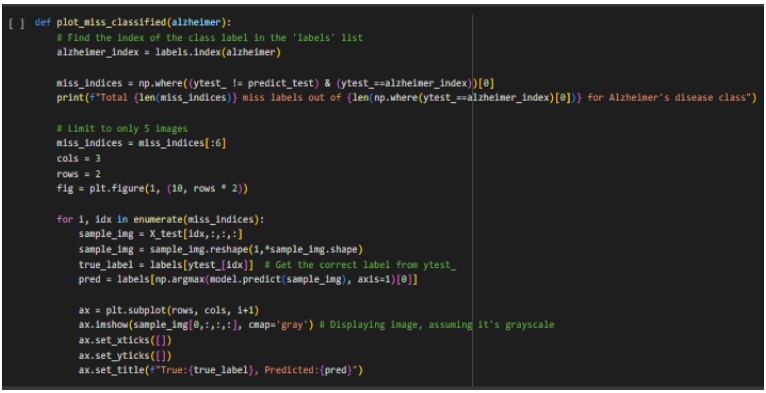
D. Pengujian/*Testing* Model

Setelah dilakukannya pelatihan model, tahap selanjutnya adalah melakukan pengujian pada model. Melalui pengujian ini, peneliti akan mengevaluasi sejauh mana model mampu mengidentifikasi kelas dengan tepat. Pengujian ini dilakukan dengan menetukan nilai *true* dan nilai *false* dari sebuah prediksi yang direpresentasikan pada Gambar 4.9 dan Gambar 4.10.



```
[ ] def plot_true_classified(alzheimer):
        # Find the index of the class label in the 'labels' list
        alzheimer_index = labels.index(alzheimer)

        true_indices = np.where((ytest_ == predict_test) & (ytest_==alzheimer_index))[0]
        print(f"Total {len(true_indices)} true labels out of {len(np.where(ytest_==alzheimer_index)[0])} for Alzheimer's disease class")

        # Limit to only 5 images
        true_indices = true_indices[:6]
        cols = 3
        rows = 2
        fig = plt.figure(1, (10, rows * 2))

        for i, idx in enumerate(true_indices):
            sample_img = X_test[idx,:,:,:]
            sample_img = sample_img.reshape(1,*sample_img.shape)
            true_label = labels[ytest_[idx]]  # Get the correct label from ytest_
            pred = labels[np.argmax(model.predict(sample_img), axis=1)[0]]

            ax = plt.subplot(rows, cols, i+1)
            ax.imshow(sample_img[0,:,:,:], cmap='gray') # Displaying image, assuming it's grayscale
            ax.set_xticks([])
            ax.set_yticks([])
            ax.set_title(f"True:{true_label}, Predicted:{pred}")
```

Gambar 4. 9 Proses Menetukan Nilai *True*



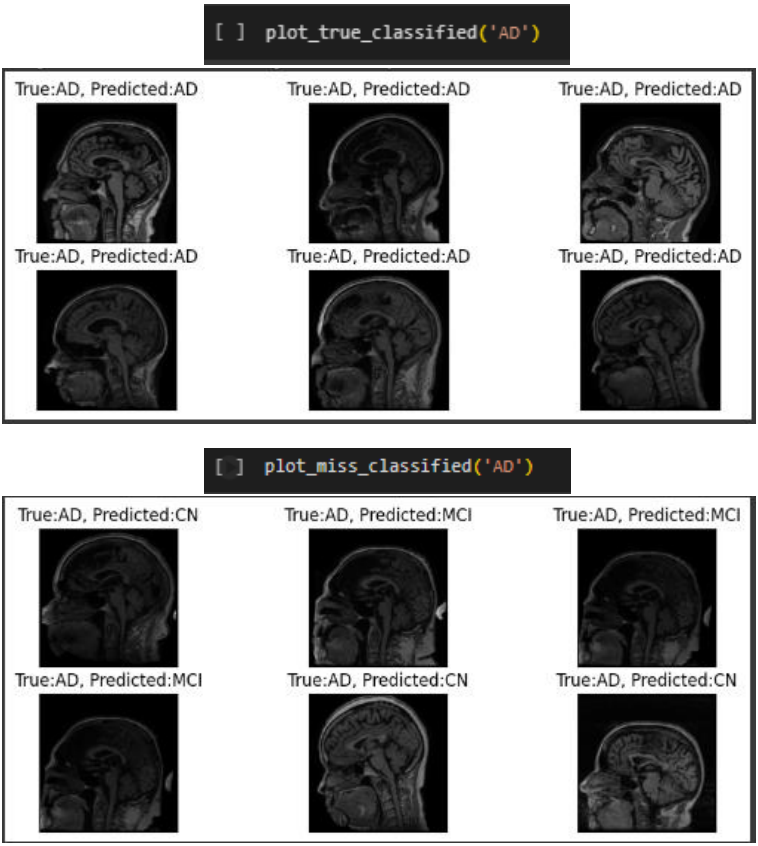
```
[ ] def plot_miss_classified(alzheimer):
        # Find the index of the class label in the 'labels' list
        alzheimer_index = labels.index(alzheimer)

        miss_indices = np.where((ytest_ != predict_test) & (ytest_==alzheimer_index))[0]
        print(f"Total {len(miss_indices)} miss labels out of {len(np.where(ytest_==alzheimer_index)[0])} for Alzheimer's disease class")

        # Limit to only 5 images
        miss_indices = miss_indices[:6]
        cols = 3
        rows = 2
        fig = plt.figure(1, (10, rows * 2))

        for i, idx in enumerate(miss_indices):
            sample_img = X_test[idx,:,:,:]
            sample_img = sample_img.reshape(1,*sample_img.shape)
            true_label = labels[ytest_[idx]]  # Get the correct label from ytest_
            pred = labels[np.argmax(model.predict(sample_img), axis=1)[0]]

            ax = plt.subplot(rows, cols, i+1)
            ax.imshow(sample_img[0,:,:,:], cmap='gray') # Displaying image, assuming it's grayscale
            ax.set_xticks([])
            ax.set_yticks([])
            ax.set_title(f"True:{true_label}, Predicted:{pred}")
```

Gambar 4. 10 Proses Menetukan Nilai *False*

Dalam Gambar 4.9 dan gambar 4.10, fungsi *plot_true_classified* dan *plot_miss_classified* adalah bagian dari proses evaluasi performa model dalam memprediksi kelas pada data pengujian. Fungsi *plot_true_classified* bertujuan untuk menampilkan sampel gambar yang telah benar-benar diklasifikasikan dengan tepat oleh model untuk kelas tertentu, dalam hal ini kelas Alzheimer. Sedangkan fungsi *plot_miss_classified* bertujuan untuk menampilkan sampel gambar diprediksi salah oleh model. Prosesnya dimulai dengan mencari indeks label kelas Alzheimer dalam daftar label. Kemudian, indeks yang sesuai dengan kelas tersebut dari data pengujian dipilih untuk dianalisis. Sampel-sampel yang dipilih

ditampilkan bersama dengan label yang sebenarnya dan label yang diprediksi oleh model.

1. Proses pengujian Pada Kelas AD



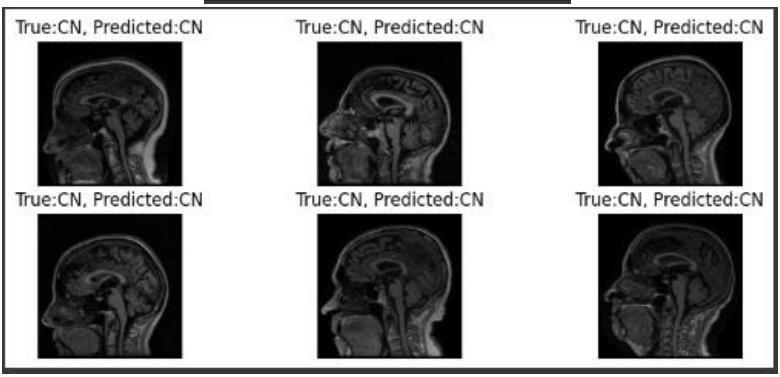
Gambar 4. 11 Hasil Proses Pengujian Kelas AD

Pada gambar 4.11, dapat dilihat fungsi *plot_true_classified('AD')* menampilkan 6 sampel

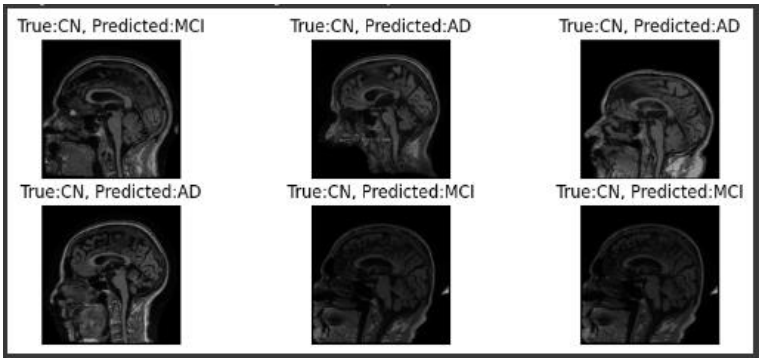
gambar yang telah diklasifikasikan dengan benar oleh model sebagai kelas AD. Sedangkan fungsi *plot_miss_classified('AD')* menampilkan 6 sampel gambar yang salah diklasifikasikan oleh model. Yang dimana gambar tersebut merupakan kelas AD tapi diprediksi sebagai kelas CN dan kelas MCI.

2. Proses Pengujian Pada Kelas CN

```
plot_true_classified('CN')
```




```
[ ] plot_miss_classified('CN')
```

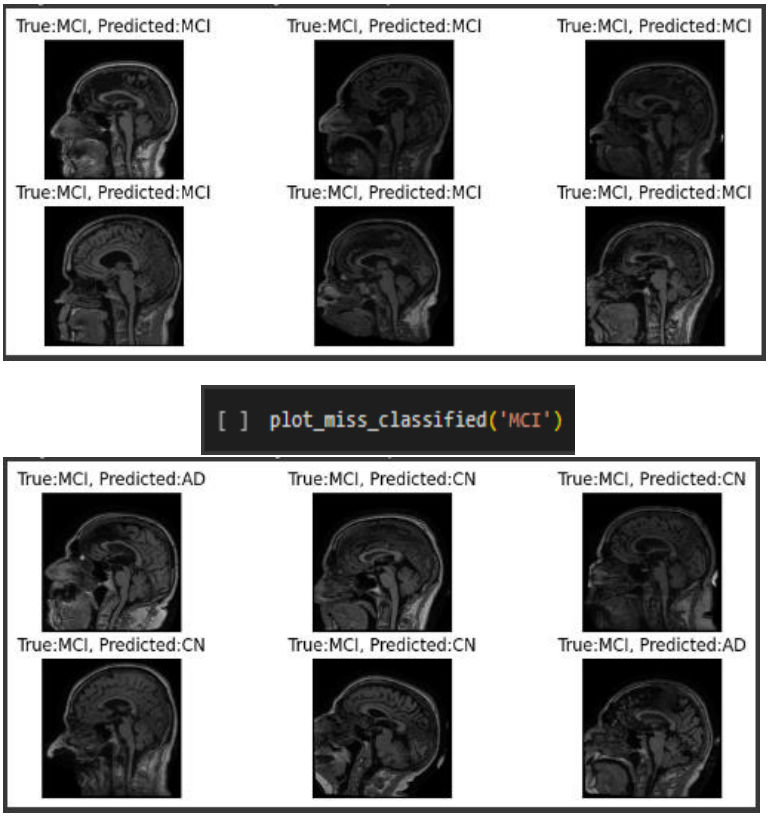


Gambar 4. 12 Hasil Proses Pengujian Kelas CN

Pada gambar 4.12, dapat dilihat fungsi *plot_true_classified('CN')* menampilkan 6 sampel gambar yang telah diklasifikasikan dengan benar oleh model sebagai kelas CN. Sedangkan fungsi *plot_miss_classified('CN')* menampilkan 6 sampel gambar yang salah diklasifikasikan oleh model. Yang dimana gambar tersebut merupakan kelas CN tapi diprediksi sebagai kelas AD dan kelas MCI.

3. Proses Pengujian Pada Kelas MCI

```
[ ] plot_true_classified('MCI')
```



Gambar 4. 13 Hasil Proses Pengujian Kelas MCI

Pada gambar 4.13, dapat dilihat fungsi *plot_true_classified('MCI')* menampilkan 6 sampel gambar yang telah diklasifikasikan dengan benar oleh model sebagai kelas MCI. Sedangkan fungsi *plot_miss_classified('MCI')* menampilkan 6 sampel gambar yang salah diklasifikasikan oleh model. Yang

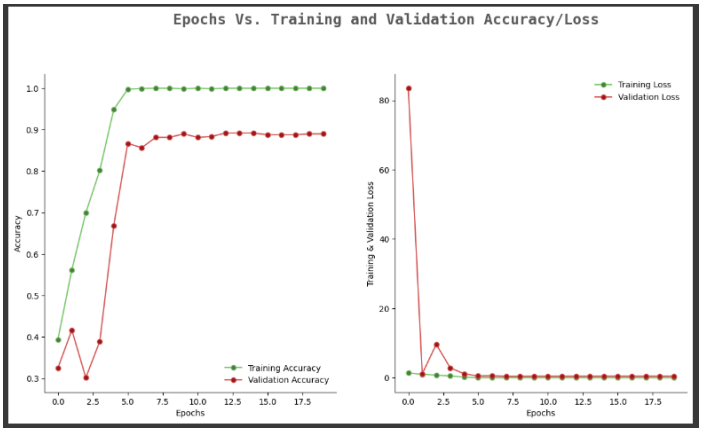
dimana gambar tersebut merupakan kelas MCI tapi diprediksi sebagai kelas AD dan kelas CN.

E. Evaluasi Model

1. Evaluasi Model Berdasarkan *Learning Curve*

Setelah pembuatan model CNN dan penentuan nilai untuk setiap *hyperparameter*, model akan mengalami tahap pelatihan (*training*) di mana akan dihasilkan nilai *loss* dan akurasi yang direpresentasikan dalam *Learning Curve* seperti yang terlihat pada Gambar 4.14.

```
#Proses memvisualisasikan Learning Curve
filterwarnings('ignore')

epochs = [i for i in range(20)]
fig, ax = plt.subplots(1,2,figsize=(14,7))
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']

fig.text (s='Epochs Vs. Training and Validation Accuracy/Loss',
          size=18, fontweight='bold',
          fontname='monospace',
          color=colors_dark[1], y=1, x=0.28, alpha=0.8)
sns.despine()
ax[0].plot(epochs, train_acc, marker='o',
           markerfacecolor=colors_green[2], color=colors_green[3], label = 'Training Accuracy')
ax[0].plot(epochs, val_acc, marker='o',
           markerfacecolor=colors_red[2], color=colors_red[3], label = 'Validation Accuracy')
ax[0].legend(frameon=False)
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Accuracy')

sns.despine()
ax[1].plot(epochs, train_loss, marker='o',
           markerfacecolor=colors_green[2], color=colors_green[3], label = 'Training Loss')
ax[1].plot(epochs, val_loss, marker='o',
           markerfacecolor=colors_red[2], color=colors_red[3], label = 'Validation Loss')
ax[1].legend(frameon=False)
ax[1].set_xlabel('Epochs')
ax[1].set_ylabel('Training & Validation Loss')

fig.show()
```



Gambar 4. 14 Proses Visualiasi *Learning Curve*

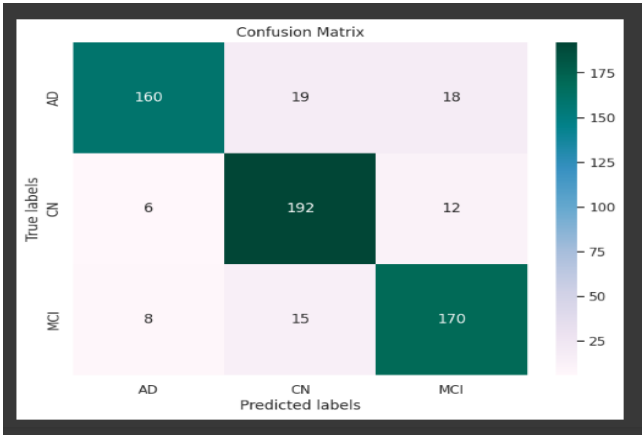
Dari Gambar 4.14, dapat diamati bahwa akurasi pada data pelatihan dan validasi terus meningkat seiring berjalannya *epoch*, sementara nilai *loss* pada kedua dataset tersebut terus menurun. Peningkatan akurasi dan penurunan *loss* pada kedua dataset tersebut kemudian *konvergen* pada epoch ke-5. Fenomena ini menunjukkan performa yang baik dari model, dengan nilai yang stabil. Akurasi mencapai 0.87 atau 87% dan *loss* hanya 0.01 atau 1% pada akhir epoch, menunjukkan kinerja model yang cukup baik. Grafik pada Gambar 4.13 menunjukkan bahwa data yang telah dilatih merupakan hasil *fitting* terbaik yang diperoleh selama percobaan ini.

2. Evaluasi Model Berdasarkan *Confusion Matrix*

Setelah mengevaluasi performa model melalui *Learning Curve*, langkah berikutnya adalah mengevaluasi performa model menggunakan *confusion matrix* pada data yang dilatih. *Confusion matrix* untuk model CNN yang telah dibuat dapat ditemukan dalam Gambar 4.16.

```
#evaluasi model test
predict_test = np.argmax(model.predict(X_test), axis=1)
ytest_ = np.argmax(y_test, axis=1)


# Menghitung confusion matrix
conf_matrix = confusion_matrix(ytest_, predict_test)
test_accu = np.sum(ytest_ == predict_test) / len(ytest_) * 100

#menampilkan hasil akurasi dan classification report
print(classification_report(ytest_, predict_test))
print(f"Test Accuracy: {round(test_accu, 4)} %\n\n")


# Membuat plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, xticklabels=labels, yticklabels=labels, annot=True, fmt='d', cmap='PuBuGn', cbar=True)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```

Gambar 4. 15 Proses Visualiasi *Confusion Matrix*



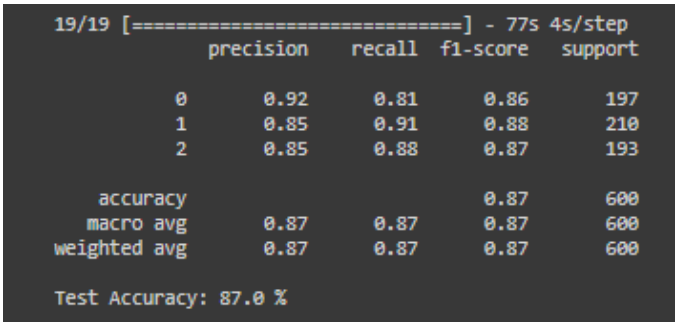
Gambar 4. 16 Hasil Proses Visualiasi *Confusion Matrix*

Pada Gambar 4.15, dapat dilihat bahwa model CNN yang telah dibuat, terdapat 3 kelas yang di prediksikan, yaitu kelas *Alzheimer Disease* (AD), *Cognitive Normal* (CN) dan *Mild Cognitive Impairment* (MCI). Pada kelas AD memprediksi sebanyak 197 data yang dimana 160 diprediksi benar sebagai kelas AD, 19 data diprediksi sebagai CN dan 18 data diprediksi sebagai MCI.

Lalu pada kelas *Cognitive Normal* (CN), memprediksi sebanyak 210 data yang dimana 192 data diprediksi benar sebagai kelas AD, dan sisa nya diprediksi sebagai kelas CN dan MCI dengan masing – masing berjumlah 6 dan 12 data.

Sedangkan pada kelas MCI, ada sebanyak 170 data yang diprediksi benar dan ada 15 data yang diprediksi sebagai kelas CN, 8 data yang diprediksi sebagai kelas AD dari total 193 data.

Berdasarkan informasi yang terdapat dalam *confusion matrix*, akan diperoleh hasil evaluasi berupa presisi (*precision*), *recall*, *f1-score*, dan akurasi. Informasi ini tersedia dalam Gambar 4.17.



Gambar 4. 17 Hasil *Classification Report*

Menurut Gambar 4.16, Presisi model untuk setiap kelas menunjukkan kualitas yang baik. Untuk kategori *Alzheimer Disease* (AD), presisi model adalah 0.92. Sementara itu, presisi untuk kategori *Cognitive Normal* (CN) dan *Mild Cognitive Impairment* (MCI) adalah 0.85.

Dalam hal *recall*, model juga menunjukkan performa yang baik. Untuk kategori *Alzheimer Disease* (AD), recall mencapai 0.81, yang berarti 81% data *Alzheimer Disease* (AD) dapat terdeteksi dengan benar. Recall untuk kategori *Cognitive Normal* (CN) dan *Mild Cognitive Impairment* (MCI) adalah 0.91 dan 0.88.

F1-score, yang mencerminkan keseimbangan antara presisi dan recall, juga menunjukkan hasil yang baik. Nilai F1-score untuk *Alzheimer Disease*

(AD)adalah 0.86, untuk *Cognitive Normal* (CN) adalah 0.88, untuk *Mild Cognitive Impairment* (MCI) adalah 0.87.

Secara keseluruhan, evaluasi pada Gambar 4.16 menunjukkan bahwa model memiliki performa yang baik dalam mengklasifikasikan data, dengan nilai presisi, *recall*, *f1-score*, dan akurasi mendapatkan nilai yang sama yaitu 0.87 atau 87%.

Selanjutnya peneliti melakukan perhitungan performa secara manual yang meliputi nilai *acuraccy, precision, recall* dan juga *f1-score* berdasarkan *confusion matrix*. Lalu untuk penentuan nilai TP, TN, FP, dan FN sudah dijelaskan pada bab sebelumnya yaitu pada Gambar 2.9.

Tabel 4. 1 Hasil *Confusion Matrix*

		Predicted Labels		
		AD	**CN**	**MCI**
True Labels	**AD**	160	19	18
	CN	6	192	12
	MCI	8	15	170

Berdasarkan Tabel 4.1 didapatkan nilai dari *confusion matrix* dari masing-masing kelas maka

selanjutnya akan dilakukan proses perhitungan performa dari model yang digunakan secara manual. Hasil dari perhitungan akurasi, presisi, *recall*, dan *f1-score* yang dilakukan secara manual akan dijelaskan pada proses dibawah ini.

$$Accuracy = \frac{TP_{AD} + TN_{CN} + TN_{MCI}}{Jumlah\ Data} \times 100$$

$$Accuracy = \frac{160 + 192 + 170}{160 + 19 + 18 + 6 + 192 + 12 + 8 + 15 + 170} \times 100$$

$$Accuracy = \frac{522}{600} \times 100$$

$$Accuracy = 0.87 \times 100$$

$$Accuracy = 87\%$$

Setelah diketahui nilai akurasi nya maka selanjutnya yaitu melakukan perhitungan terhadap performal model yang lainnya seperti nilai dari precission, recall, dan f1-score berdasarkan nilai true positif, nilai false positif, dan juga nilai false negatif. Karena dalam menentukan nilai true positif, nilai false positif, dan juga nilai false negatif pada confusion matrix 3x3 sangat sulit maka solusi yang

diberikan untuk mempermudah pencarian masing–masing nilai yaitu dengan memcah kolom 3X3 menjadi kolom 2X2 terlebih dahulu yang akan ditunjukkan pada table tabel di bawah ini:

a. Perhitungan pada Kelas AD

Tabel 4. 2 Perhitungan Kelas AD

True/Predic	AD	Bukan AD
AD	TP = 160	FN = 19+18 = 37
Bukan AD	FP=6+8 = 14	TN=192+12+15+170 = 389

Berdasarkan Tabel 4.2, didapatkan nilai TP=160, nilai FP=14, nilai FN=37, dan nilsi TN=389. Setelah itu, mulai menghitung nilai precission, recall, dan f1-score.

$$Precision = \frac{TP}{TP + FP}$$

$$Precision = \frac{160}{160 + 14}$$

$$Precision = \frac{160}{174}$$

$$Precision = 0.919$$

$$Precision = 0.92$$

$$Recall = \frac{TP}{TP + FN}$$

$$Recall = \frac{160}{160 + 37}$$

$$Recall = \frac{160}{197}$$

$$Recall = 0.812$$

$$Recall = 0.81$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$F1 - Score = 2 \times \frac{0.92 \times 0.81}{0.92 + 0.81}$$

$$F1 - Score = \frac{1.4904}{1.73}$$

$$F1 - Score = 0.861$$

$$F1 - Score = 0.86$$

b. Perhitungan pada Kelas CN

Tabel 4. 3 Perhitungan Kelas CN

True/Predic	CN	Bukan CN
CN	TP = 192	FN = 6+12 = 18
Bukan CN	FP=19+15= 34	TN=160+18+8+170 = 354

Berdasarkan Tabel 4.3, didapatkan nilai TP=192, nilai FP=34, nilai FN=18, dan nilai TN=354. Setelah itu, mulai menghitung nilai precission, recall, dan f1-score.

$$Precision = \frac{TP}{TP + FP}$$
$$Precision = \frac{192}{192 + 34}$$
$$Precision = \frac{192}{226}$$
$$Precision = 0.849$$
$$Precision = 0.85$$

$$Recall = \frac{TP}{TP + FN}$$
$$Recall = \frac{192}{192 + 18}$$
$$Recall = \frac{192}{210}$$
$$Recall = 0.914$$
$$Recall = 0.91$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$
$$F1 - Score = 2 \times \frac{0.85 \times 0.91}{0.85 + 0.91}$$

$$F1 - Score = \frac{1.547}{1.76}$$

$$F1 - Score = 0.878$$

$$F1 - Score = 0.88$$

c. Perhitungan pada Kelas MCI

Tabel 4. 4 Perhitungan Kelas MCI

True/Predic	MCI	Bukan MCI
MCI	TP = 170	FN = 8+15 = 23
Bukan MCI	FP= 18+12 = 30	TN= 160+19+6+192 = 377

Berdasarkan Tabel 4.4, didapatkan nilai TP=170, nilai FP=30, nilai FN=23, dan nilai TN=377. Setelah itu, mulai menghitung nilai precission, recall, dan f1-score.

$$Precision = \frac{TP}{TP + FP}$$

$$Precision = \frac{170}{170 + 30}$$

$$Precision = \frac{170}{200}$$

$$Precision = 0.85$$

$$Recall = \frac{TP}{TP + FN}$$

$$Recall = \frac{170}{170 + 23}$$

$$Recall = \frac{170}{193}$$

$$Recall = 0.88$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$F1 - Score = 2 \times \frac{0.85 \times 0.88}{0.85 + 0.88}$$

$$F1 - Score = \frac{1.496}{1.73}$$

$$F1 - Score = 0.87$$

Berikut merupakan hasil dari perhitungan diatas yang masukkan ke dalam tabel 4.5.

Tabel 4. 5 Hasil Perhitungan Performa Model

Kelas	**TP**	**FP**	**FN**	***Precision***	***Recall***	***F1-Score***
AD	160	14	37	0.92	0.81	0.86
CN	192	34	18	0.85	0.91	0.88
MCI	170	30	23	0.85	0.88	0.87

Berdasarkan tabel 4.5, dilakukan perhitungan manual untuk *Precision, Recall,* dan *F1-Score.* Hasil peritungan yang dilakukan secara manual akan dijelaskan pada proses di bawah.

Untuk perhitungan *Precision* menggunakan persamaan di bawah ini :

$$Precision = \frac{TP_{AD} + TP_{CN} + TP_{MCI}}{TP_{total} + F_{total}}$$

$$Precision = \frac{160 + 192 + 170}{160 + 192 + 170 + 14 + 34 + 30}$$

$$Precision = \frac{522}{600}$$

$$Precision = 0.87$$

Untuk perhitungan *Recall* menggunakan persamaan di bawah ini :

$$Recall = \frac{TP_{AD} + TP_{CN} + TP_{MCI}}{TP_{total} + FN_{total}}$$

$$Recall = \frac{160 + 192 + 170}{160 + 192 + 170 + 37 + 18 + 23}$$

$$Recall = \frac{522}{600}$$

$$Recall = 0.87$$

Untuk perhitungan *Recall* menggunakan persamaan di bawah ini :

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$F1 - Score = 2 \times \frac{0.87 \times 0.87}{0.87 + 0.87}$$

$$F1 - Score = \frac{1.5138}{1.74}$$

$$F1 - Score = 0.87$$

Selain uji coba pada perbandingan data 80:20, dilakukan juga uji coba pada perbandingan 90:10 dengan hasil akurasi seperti padaTabel 4.6.

Tabel 4. 6 Hasil Akurasi Berdasarkan Perbandingan Data

Perbandingan Data	Hasil Akurasi
90:10	94.67%
80:20	87%

BAB V
KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan penerapan dan pengujian model *ResNet50* untuk klasifikasi MRI *Alzheimer* yang telah berhasil dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Dengan memanfaatkan metode *Convolutional Neural Network* (CNN) dan menggunakan model arsitektur *ResNet50*, sebuah model *machine learning* telah berhasil dikembangkan untuk mendeteksi penyakit *Alzheimer*.
2. Melalui analisis *Learning Curve*, ditemukan bahwa model mencapai *konvergensi* pada sekitar epoch ke-5, yang menunjukkan bahwa proses pelatihan telah stabil dan model telah mempelajari pola yang cukup dari data latih. Ini menegaskan keefektifan arsitektur *ResNet50* dalam menangani dataset kompleks seperti citra MRI otak untuk klasifikasi penyakit *Alzheimer*.
3. Dari hasil *confusion matrix* dan *classification report,* ditemukan bahwa model memiliki tingkat akurasi sebesar 87%, *precision* sebesar 87%, *recall* sebesar

87%, dan *f1-score* sebesar 87% untuk perbandingan data 80:20. Hal ini menunjukkan bahwa model memiliki keseimbangan yang baik antara kemampuan untuk mengidentifikasi kelas positif dan kelas negatif, serta mengurangi jumlah kesalahan klasifikasi. Sedangkan pada percobaan pada perbandingan data 90:10 didapatkan hasil akurasi, *precision, recall, dan f1-score* sebesar 94.67%. Namun, evaluasi pada set pengujian dengan perbandingan 90:10 dapat menjadi kurang andal karena ukurannya yang lebih kecil, sehingga *trade-off* antara pelatihan model yang lebih baik dan evaluasi yang lebih akurat harus dipertimbangkan secara cermat.

B. Saran

Berikut adalah beberapa saran yang dapat dipertimbangkan untuk penelitian mendatang:

1. Meningkatkan jumlah dataset pelatihan dapat menjadi langkah yang penting untuk meningkatkan kinerja model *Convolutional Neural Network* (CNN). Dengan dataset yang lebih besar, model dapat mempelajari pola yang lebih akurat dan dapat menghasilkan prediksi yang lebih tepat.

2. Selain model *ResNet50*, penelitian selanjutnya dapat mengeksplorasi berbagai arsitektur CNN yang berbeda, seperti *Inception*, *MobileNet*, dan sebagainya. Membandingkan performa dari berbagai model ini dapat memberikan wawasan yang lebih baik tentang model mana yang paling efektif untuk tugas deteksi penyakit *Alzheimer*.
3. Untuk penelitian selanjutnya, disarankan untuk menggunakan data medis asli tanpa mengubah format, seperti menggunakan file NIfTI langsung, karena dapat memberikan keuntungan tertentu. Menggunakan data medis asli memungkinkan peneliti untuk mempertahankan semua informasi dan detail yang terkandung dalam citra medis tersebut. Misalnya, dapat menyimpan data dengan kedalaman bit yang tinggi dan metadata tambahan yang penting untuk analisis medis. Dengan mempertahankan format asli, peneliti dapat memastikan bahwa tidak ada informasi yang hilang atau terdistorsi selama proses konversi.

DAFTAR PUSTAKA

Asma-Ull, H., Yun, I. D., & Yun, B. La. (2023). Regression to Classification: Ordinal Prediction of Calcified Vessels Using Customized ResNet50. *IEEE Access*, *11*, 48783–48796. https://doi.org/10.1109/ACCESS.2023.3270562

Basha, S. H. S., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2020). Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, *378*, 112–119. https://doi.org/10.1016/j.neucom.2019.10.008

Benoit, J. S., Chan, W., Piller, L., & Doody, R. (2020). Longitudinal Sensitivity of Alzheimer's Disease Severity Staging. *American Journal of Alzheimer's Disease & Other Dementiasr*, *35*, 153331752091871. https://doi.org/10.1177/1533317520918719

Bisriyah, S. (2022). Directive Speech Acts and Wisdom Values in Q.S. Al Mu'min (A Pragmatics Study). *Islah: Journal of Islamic Literature and History*, *3*(1), 59–78. https://doi.org/10.18326/islah.v3i1.59-78

Chandarana, H., Wang, H., Tijssen, R. H. N., & Das, I. J. (2018). Emerging role of MRI in radiation therapy. *Journal of Magnetic Resonance Imaging*, *48*(6), 1468–1478. https://doi.org/10.1002/jmri.26271

Chelghoum, R., Ikhlef, A., Hameurlaine, A., & Jacquir, S. (2020). *Transfer Learning Using Convolutional Neural Network Architectures for Brain Tumor Classification from MRI Images* (pp. 189–200). https://doi.org/10.1007/978-3-030-49161-1_17

Cho, S. I., & Kang, S.-J. (2019). Gradient Prior-Aided CNN Denoiser With Separable Convolution-Based Optimization of Feature

Dimension. *IEEE Transactions on Multimedia*, *21*(2), 484–493. https://doi.org/10.1109/TMM.2018.2859791

Christlein, V., Spranger, L., Seuret, M., Nicolaou, A., Kral, P., & Maier, A. (2019). Deep Generalized Max Pooling. *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 1090–1096. https://doi.org/10.1109/ICDAR.2019.00177

Clare, L. (2022). Alzheimer´s Disease International: World Alzheimer Report 2022 – Life after diagnosis. *Cognitive Rehabilitation: A Personalised, Strengths-Based Approach to Supporting Functional Ability*, 208–211. https://www.alzint.org/u/World-Alzheimer-Report-2022.pdf

Daanouni, O., Cherradi, B., & Tmiri, A. (2022). NSL-MHA-CNN: A Novel CNN Architecture for Robust Diabetic Retinopathy Prediction Against Adversarial Attacks. *IEEE Access*, *10*, 103987–103999. https://doi.org/10.1109/ACCESS.2022.3210179

Dubey, S. R., Chakraborty, S., Roy, S. K., Mukherjee, S., Singh, S. K., & Chaudhuri, B. B. (2020). diffGrad: An Optimization Method for Convolutional Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(11), 4500–4511. https://doi.org/10.1109/TNNLS.2019.2955777

Elhadad, A., Jamjoom, M., & Abulkasim, H. (2024). Reduction of NIFTI files storage and compression to facilitate telemedicine services based on quantization hiding of downsampling approach. *Scientific Reports*, *14*(1), 5168. https://doi.org/10.1038/s41598-024-54820-4

Esrayanti Simanjuntak, Nurul Khairina, Zulfikar Sembirirng, Rizki Muliono, & Muhathir Muhathir. (2023). Analisis Fungsi Aktivasi pada Algoritma Backpropagation dalam Pengenalan

Aksara Batak Toba. *JUSTINDO (Jurnal Sistem Dan Teknologi Informasi Indonesia)*, *8*(2), 99–107. https://doi.org/10.32528/justindo.v8i2.331

Falakhi, B., Achmal, E. F., Rizaldi, M., Athallah, R. R. R., & Yudistira, N. (2022). Perbandingan Model AlexNet dan ResNet dalam Klasifikasi Citra Bunga Memanfaatkan Transfer Learning. *Jurnal Ilmu Komputer Dan Agri-Informatika*, *9*(1), 70–78. https://doi.org/10.29244/jika.9.1.70-78

Firmansyah, I., & Hayadi, B. H. (2022). Komparasi Fungsi Aktivasi Relu Dan Tanh Pada Multilayer Perceptron. *JIKO (Jurnal Informatika Dan Komputer)*, *6*(2), 200. https://doi.org/10.26798/jiko.v6i2.600

Gan, J., Wang, M., Liu, S., Chen, Z., Wang, X.-D., & Ji, Y. (2021). Effect of Multiple Medicines on Dementia Initial Treatment: Experience and Thinking. *American Journal of Alzheimer's Disease & Other Dementias®*, *36*, 153331752110531. https://doi.org/10.1177/15333175211053134

Grattarola, D., & Alippi, C. (2021). Graph Neural Networks in TensorFlow and Keras with Spektral [Application Notes]. *IEEE Computational Intelligence Magazine*, *16*(1), 99–106. https://doi.org/10.1109/MCI.2020.3039072

Hidayatul, N., & Sinuraya, R. K. (2016). Biomarker miRNA-146a sebagai 1. Hidayatul N, Sinuraya RK. Biomarker miRNA-146a sebagai Deteksi Dini yang Efektif untuk Alzheimer. Farmaka. 2016;15(2):159–77. Deteksi Dini yang Efektif untuk Alzheimer. *Farmaka*, *15*(2), 159–177.

Hridhee, R. A., Bhowmik, B., & Hossain, Q. D. (2023). Alzheimer's Disease Classification From 2D MRI Brain Scans Using Convolutional Neural Networks. *2023 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 1–6.

https://doi.org/10.1109/ECCE57851.2023.10101539

Imoto, K., Nakai, T., Ike, T., Haruki, K., & Sato, Y. (2019). A CNN-Based Transfer Learning Method for Defect Classification in Semiconductor Manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, *32*(4), 455–459. https://doi.org/10.1109/TSM.2019.2941752

Kang, S. K. (2019). Measuring the value of MRI: Comparative effectiveness & outcomes research. *Journal of Magnetic Resonance Imaging*, *49*(7). https://doi.org/10.1002/jmri.26647

Kulkarni, A., Chong, D., & Batarseh, F. A. (2020). Foundations of data imbalance and solutions for a data democracy. In *Data Democracy* (pp. 83–106). Elsevier. https://doi.org/10.1016/B978-0-12-818366-3.00005-8

Kulsum, U., & Cherid, A. (2023). Penerapan Convolutional Neural Network Pada Klasifikasi Tanaman Menggunakan ResNet50. *SIMKOM*, *8*(2), 221–228. https://doi.org/10.51717/simkom.v8i2.191

Liu, T. T. (2020). MRI in systems medicine. *WIREs Systems Biology and Medicine*, *12*(1). https://doi.org/10.1002/wsbm.1463

Lu, L., Yi, Y., Huang, F., Wang, K., & Wang, Q. (2019). Integrating Local CNN and Global CNN for Script Identification in Natural Scene Images. *IEEE Access*, *7*, 52669–52679. https://doi.org/10.1109/ACCESS.2019.2911964

Luque, A., Carrasco, A., Martín, A., & de las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, *91*, 216–231. https://doi.org/10.1016/j.patcog.2019.02.023

Mahadevkar, S. V., Khemani, B., Patil, S., Kotecha, K., Vora, D. R.,

Abraham, A., & Gabralla, L. A. (2022). A Review on Machine Learning Styles in Computer Vision—Techniques and Future Directions. *IEEE Access*, *10*, 107293–107329. https://doi.org/10.1109/ACCESS.2022.3209825

Mahfudh, A. A., & Mustofa, H. (2019). Klasifikasi Pemahaman Santri Dalam Pembelajaran Kitab Kuning Menggunakan Algoritma Naive Bayes Berbasis Forward Selection. *Walisongo Journal of Information Technology*, *1*(2), 101. https://doi.org/10.21580/wjit.2019.1.2.4529

Mao, Y., He, Z., Ma, Z., Tang, X., & Wang, Z. (2019). Efficient Convolution Neural Networks for Object Tracking Using Separable Convolution and Filter Pruning. *IEEE Access*, *7*, 106466–106474. https://doi.org/10.1109/ACCESS.2019.2932733

Marques, J. P., Simonis, F. F. J., & Webb, A. G. (2019). Low-field MRI: An MR physics perspective. *Journal of Magnetic Resonance Imaging*, *49*(6), 1528–1542. https://doi.org/10.1002/jmri.26637

Mavroudis, I., Petridis, F., Kazis, D., Njau, S. N., Costa, V., & Baloyannis, S. J. (2019). Purkinje Cells Pathology in Alzheimer's Disease. *American Journal of Alzheimer's Disease & Other Dementias®*, *34*(7–8), 439–449. https://doi.org/10.1177/1533317519859200

Murugan, S., Venkatesan, C., Sumithra, M. G., Gao, X.-Z., Elakkiya, B., Akila, M., & Manoharan, S. (2021). DEMNET: A Deep Learning Model for Early Diagnosis of Alzheimer Diseases and Dementia From MR Images. *IEEE Access*, *9*, 90319–90329. https://doi.org/10.1109/ACCESS.2021.3090474

Mutasa, S., Sun, S., & Ha, R. (2020). Understanding artificial intelligence based radiology studies: What is overfitting? *Clinical Imaging*, *65*, 96–99.

https://doi.org/10.1016/j.clinimag.2020.04.025

Nisa, K. M., & Lisiswanti, R. (2016). Faktor Risiko Demensia. *Majority*, *5*(4), 86–87. http://juke.kedokteran.unila.ac.id/index.php/majority/article/view/890

Pang, Y., Sun, M., Jiang, X., & Li, X. (2018). Convolution in Convolution for Network in Network. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(5), 1587–1597. https://doi.org/10.1109/TNNLS.2017.2676130

Rasel, M. A., Obaidellah, U. H., & Kareem, S. A. (2022). Convolutional Neural Network-Based Skin Lesion Classification With Variable Nonlinear Activation Functions. *IEEE Access*, *10*, 83398–83414. https://doi.org/10.1109/ACCESS.2022.3196911

Ravichandiran, S. (2019). *Hands-On Deep Learning Algorithms with Python*. Packt.

Ribani, R., & Marengoni, M. (2019). A Survey of Transfer Learning for Convolutional Neural Networks. *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, 47–57. https://doi.org/10.1109/SIBGRAPI-T.2019.00010

Ruuska, S., Hämäläinen, W., Kajava, S., Mughal, M., Matilainen, P., & Mononen, J. (2018). Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle. *Behavioural Processes*, *148*, 56–62. https://doi.org/10.1016/j.beproc.2018.01.004

Sanchez, S. A., Romero, H. J., & Morales, A. D. (2020). A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework. *IOP Conference Series: Materials Science and Engineering*, *844*,

012024. https://doi.org/10.1088/1757-899X/844/1/012024

Shinde, P. P., & Shah, S. (2018). A Review of Machine Learning and Deep Learning Applications. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 1–6. https://doi.org/10.1109/ICCUBEA.2018.8697857

Viering, T., & Loog, M. (2023). The Shape of Learning Curves: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(6), 7799–7819. https://doi.org/10.1109/TPAMI.2022.3220744

Wang, W., Wang, Z., Zhou, Z., Deng, H., Zhao, W., Wang, C., & Guo, Y. (2021). Anomaly detection of industrial control systems based on transfer learning. *Tsinghua Science and Technology*, *26*(6), 821–832. https://doi.org/10.26599/TST.2020.9010041

Wildah, S. K., Agustiani, S., S, M. R. R., Gata, W., & Nawawi, H. M. (2020). DETEKSI PENYAKIT ALZHEIMER MENGGUNAKAN ALGORITMA NAÏVE BAYES DAN CORRELATION BASED FEATURE SELECTION. *Jurnal Informatika*, *7*(2), 166–173. https://doi.org/10.31294/ji.v7i2.8226

Xu, Q., Zhang, M., Gu, Z., & Pan, G. (2019). Overfitting remedy by sparsifying regularization on fully-connected layers of CNNs. *Neurocomputing*, *328*, 69–74. https://doi.org/10.1016/j.neucom.2018.03.080

Zhang, H., Zhang, L., & Jiang, Y. (2019). Overfitting and Underfitting Analysis for Deep Learning Based End-to-end Communication Systems. *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, 1–6. https://doi.org/10.1109/WCSP.2019.8927876

Zhang, N., Cai, Y.-X., Wang, Y.-Y., Tian, Y.-T., Wang, X.-L., & Badami, B. (2020). Skin cancer diagnosis based on optimized convolutional neural network. *Artificial Intelligence in Medicine*, *102*, 101756. https://doi.org/10.1016/j.artmed.2019.101756

DAFTAR LAMPIRAN

LAMPIRAN 1 : Script Code Proses Konversi Data

```
import os
import cv2
import numpy as np
import nibabel as nib

def Nifti_to_Image(path):
    nii_img = nib.load(path)
    img_data = nii_img.get_fdata()
    img_data = (img_data / np.max(img_data) * 255).astype(np.uint8)


    if len(img_data.shape) == 3:
        img_data = img_data[:, :, img_data.shape[2] // 2]
    return img_data

def main():
    input_folder = '/content/drive/MyDrive/Data-Nifti/AD'
    output_folder = '/content/drive/MyDrive/Data-Nifti/Data-
Convert/AD'

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

    nifti_files = [file for file in os.listdir(input_folder) if
file.endswith('.nii')]

    for file_name in nifti_files:
```

```
        input_path = os.path.join(input_folder, file_name)
        output_image = Nifti_to_Image(input_path)

        output_path = os.path.join(output_folder, file_name[:-4] + '.jpg')
        cv2.imwrite(output_path, output_image)

        print(f"Konversi Berhasil")

if __name__ == "__main__":
    main()
```

LAMPIRAN 2 : Script Code Proses Penelitian

```
from google.colab import drive
drive.mount('/content/drive')

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import cv2
import tensorflow as tf
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tqdm import tqdm
import os
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.callbacks import
EarlyStopping,ReduceLROnPlateau, TensorBoard
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
import ipywidgets as widgets
import io
from PIL import Image
from IPython.display import display, clear_output
from warnings import filterwarnings
for dirname, _, filenames in os.walk('/content/drive/MyDrive/tes'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

#Palette Warna untuk membuat grafik dan tabel confusion matrix
```

```
colors_dark = ['#1F1F1F', '#313131', '#636363', '#AEAEAE',
'#DADADA']
colors_red = ['#331313', '#582626', '#9E1717', '#D35151', '#E9B4B4']
colors_green = ['#01411C', '#4B6F44', '#4F7942', '#74C365',
'#D0F0C0']

sns.palplot(colors_dark)
sns.palplot(colors_green)
sns.palplot(colors_red)

#Melabeli setiap kelas
labels = ['AD', 'CN', 'MCI']

#Resize gambar dan mengkonversi data kedalam bentuk array
X_train = []
y_train = []
image_size = 192
for i in labels:
    folderPath = os.path.join('/content/drive/MyDrive/tes/train', i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath, j))
        img = cv2.resize(img,(image_size, image_size))
        X_train.append(img)
        y_train.append(i)

for i in labels:
    folderPath = os.path.join('/content/drive/MyDrive/tes/val', i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath, j))
        img = cv2.resize(img,(image_size, image_size))
        X_train.append(img)
```

```
        y_train.append(i)

X_train = np.array(X_train)
y_train = np.array(y_train)

#Proses mengambil sampel, 1 sampel setiap kelas
k=0
fig, ax = plt.subplots(1,3,figsize=(20,20))
fig.text(s='', size=18, fontweight='bold', fontname='DejaVu Sans',
color=colors_dark[1],  y=0.62, x=0.4, alpha=0.8)
for i in labels:
    j=0
    while True:
        if y_train[j]==i:
            ax[k].imshow(X_train[j])
            ax[k].set_title(y_train[j])
            ax[k].axis('off')
            k+=1
            break
        j+=1

#Mengacak urutan data
X_train, y_train = shuffle(X_train, y_train, random_state=101)

#Mengetahui bentuk data
X_train.shape

#Mengambil sampel dengan teknik random sampling data untuk data
uji
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train,
test_size=0.2, random_state=101)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape

# Mengonversi label menjadi numerik menggunakan LabelEncoder
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
# Perbaiki jumlah kelas yang diharapkan
num_classes = len(label_encoder.classes_)
# One Hot Encoding untuk y_train
y_train = tf.keras.utils.to_categorical(y_train_encoded,
num_classes=num_classes)
# One Hot Encoding untuk y_test
y_test = tf.keras.utils.to_categorical(y_test_encoded,
num_classes=num_classes)

#Proses inisialisasi  model ResNet50
base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=(image_size,image_size,3))

# Membuat top layers baru di atas basis model
model = base_model.output
model = tf.keras.layers.GlobalAveragePooling2D()(model)
model = tf.keras.layers.Dropout(rate=0.5)(model)
model = tf.keras.layers.Dense(3, activation='softmax')(model)
model = tf.keras.models.Model(inputs=base_model.input,
outputs=model)
model.summary()

#Proses Compile
model.compile(loss='categorical_crossentropy',  optimizer = 'Adam',
metrics=['accuracy'])
```

```
#Mengatur Callback unutk pemantauan dan pengendalian pelatihan
model
tensorboard = TensorBoard(log_dir='logs')
reduce_lr=ReduceLROnPlateau(monitor="val_accuracy", factor=0.3,
patience=2, min_delta=0.001, mode="auto", verbose=1)

#Proses pelatihan akan diinisiasi
history = model.fit(X_train,y_train,validation_split=0.2,
            epochs = 20,
            verbose=1,
            batch_size=32,
            callbacks=[tensorboard,reduce_lr])

#Proses memvisualisasikan Learning Curve
filterwarnings('ignore')

epochs = [i for i in range(20)]
fig, ax = plt.subplots(1,2,figsize=(14,7))
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']

fig.text (s='Epochs Vs. Training and Validation Accuracy/Loss', size=18,
fontweight='bold', fontname='monospace', color=colors_dark[1], y=1,
x=0.28, alpha=0.8)
sns.despine()
ax[0].plot(epochs, train_acc, marker='o',
markerfacecolor=colors_green[2], color=colors_green[3], label =
'Training Accuracy')
```

```
ax[0].plot(epochs, val_acc, marker='o', markerfacecolor=colors_red[2],
color=colors_red[3], label = 'Validation Accuracy')
ax[0].legend(frameon=False)
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Accuracy')

sns.despine()
ax[1].plot(epochs, train_loss, marker='o',
markerfacecolor=colors_green[2], color=colors_green[3], label =
'Training Loss')
ax[1].plot(epochs, val_loss, marker='o', markerfacecolor=colors_red[2],
color=colors_red[3], label = 'Validation Loss')
ax[1].legend(frameon=False)
ax[1].set_xlabel('Epochs')
ax[1].set_ylabel('Training & Validation Loss')

fig.show()

#evaluasi model test
predict_test = np.argmax(model.predict(X_test), axis=1)
ytest_ = np.argmax(y_test, axis=1)


# Menghitung confusion matrix
conf_matrix = confusion_matrix(ytest_, predict_test)
test_accu = np.sum(ytest_ == predict_test) / len(ytest_) * 100

#menampilkan hasil akurasi dan classification report
print(classification_report(ytest_, predict_test))
print(f"Test Accuracy: {round(test_accu, 4)} %\n\n")
```

```
# Membuat plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, xticklabels=labels, yticklabels=labels,
annot=True, fmt='d', cmap='PuBuGn', cbar=True)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()

def plot_true_classified(alzheimer):
    # Find the index of the class label in the 'labels' list
    alzheimer_index = labels.index(alzheimer)

    true_indices = np.where((ytest_ == predict_test) &
(ytest_==alzheimer_index))[0]
    print(f"Total {len(true_indices)} true labels out of
{len(np.where(ytest_==alzheimer_index)[0])} for Alzheimer's disease
class")

    # Limit to only 5 images
    true_indices = true_indices[:6]
    cols = 3
    rows = 2
    fig = plt.figure(1, (10, rows * 2))

    for i, idx in enumerate(true_indices):
        sample_img = X_test[idx,:,:,:]
        sample_img = sample_img.reshape(1,*sample_img.shape)
        true_label = labels[ytest_[idx]]  # Get the correct label from ytest_
        pred = labels[np.argmax(model.predict(sample_img), axis=1)[0]]

        ax = plt.subplot(rows, cols, i+1)
```

```
        ax.imshow(sample_img[0,:,:,:],  cmap='gray') # Displaying image,
assuming it's grayscale
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_title(f"True:{true_label}, Predicted:{pred}")

plot_true_classified('AD')
plot_true_classified('CN')
plot_true_classified('MCI')

def plot_miss_classified(alzheimer):
    # Find the index of the class label in the 'labels' list
    alzheimer_index = labels.index(alzheimer)

    miss_indices = np.where((ytest_ != predict_test) &
(ytest_==alzheimer_index))[0]
    print(f"Total {len(miss_indices)} miss labels out of
{len(np.where(ytest_==alzheimer_index)[0])} for Alzheimer's disease
class")

    # Limit to only 5 images
    miss_indices = miss_indices[:6]
    cols = 3
    rows = 2
    fig = plt.figure(1, (10, rows * 2))

    for i, idx in enumerate(miss_indices):
        sample_img = X_test[idx,:,:,:]
        sample_img = sample_img.reshape(1,*sample_img.shape)
        true_label = labels[ytest_[idx]]  # Get the correct label from ytest_
        pred = labels[np.argmax(model.predict(sample_img), axis=1)[0]]
```

```
        ax = plt.subplot(rows, cols, i+1)
        ax.imshow(sample_img[0,:,:,:], cmap='gray') # Displaying image,
assuming it's grayscale
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_title(f"True:{true_label}, Predicted:{pred}")

plot_miss_classified('AD')
plot_miss_classified('CN')
plot_miss_classified('MCI')
```