SKRIPSI

DETEKSI DINI *ALZHEIMER* MENGGUNAKAN *DEEP LEARNING* DENGAN ARSITEKTUR RESNET152V2 PADA CITRA *MAGNETIC RESONANCE IMAGING* (MRI)

EARLY DETECTION OF ALZHEIMER'S DISEASE USING DEEP LEARNING WITH RESNET152V2 ARCHITECTURE ON MAGNETIC RESONANCE IMAGING (MRI) SCANS

Diajukan untuk memenuhi salah satu syarat memperoleh derajat
Sarjana Sains Ilmu Fisika

RUI COSTA RAKA MILANISTI

20/459212/PA/19873

PROGRAM STUDI S1 FISIKA

DEPARTEMEN FISIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2024

HALAMAN PENGESAHAN

SKRIPSI

DETEKSI DINI *ALZHEIMER* MENGGUNAKAN *DEEP LEARNING* DENGAN ARSITEKTUR RESNET152V2 PADA CITRA *MAGNETIC RESONANCE IMAGING* (MRI)

Telah dipersiapkan dan diusulkan oleh

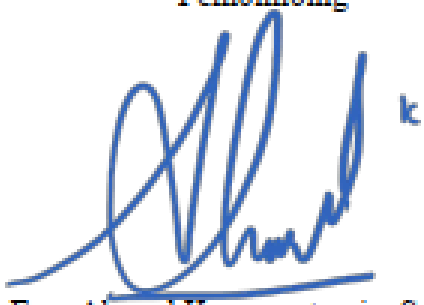
RUI COSTA RAKA MILANISTI

20/459212/PA/19873

Telah disetujui

Pada tanggal 21 November 2024

Pembimbing



Dr.Eng. Ahmad Kusumaatmaja, S.Si., M.Sc.

PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini :

Nama : Rui Costa Raka Milanisti

NIM : 20/459212/PA/19873

Program Studi : S1 Fisika

Fakultas : Matematika dan Ilmu Pengetahuan Alam

Menyatakan bahwa dalam dokumen ilmiah Skripsi ini tidak terdapat bagian dari karya ilmiah lain yang telah diajukan untuk memperoleh gelar akademik di suatu lembaga pendidikan tinggi. Dokumen ilmiah Skripsi ini juga tidak mengandung karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dan disebutkan dalam daftar pustaka.

Yogyakarta, 15 November 2024

Rui Costa Raka Milanisti

MOTTO DAN PERSEMBAHAN

Dengan rasa syukur yang mendalam, penulis mempersembahkan skripsi ini kepada kedua orang tua yang telah memberikan dukungan tanpa henti baik secara materi maupun emosional, yang menjadi motivasi utama dalam menyelesaikan penelitian ini. Tak lupa juga kepada keluarga yang senantiasa memberikan semangat dan perhatian, serta kepada berbagai pihak yang telah memberikan dukungan baik berupa bimbingan, nasihat, maupun bantuan lainnya selama proses perkuliahan dan penyusunan skripsi ini.

"Segala sesuatu bukan tentang bisa atau tidak,
tetapi soal mau atau tidak"

- Jayadi (Ayah Penulis) -

"*The easiest thing in life is to give up, the hardest thing is to keep going, but we have to.*"

- Lewis Hamilton -

"*Just keep swimming.*"

- Dory (Finding Nemo) -

PRAKATA

Puji syukur penulis panjatkan atas segala rahmat dan karunia yang diberikan oleh Allah SWT sehingga penulis dapat menyelesaikan Skripsi yang berjudul "Deteksi Dini *Alzheimer* Menggunakan *Deep Learning* dengan Arsitektur ResNet152v2 pada Citra *Magnetic Resonance Imaging* (MRI)". Skripsi ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Sains pada Program Studi Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada.

Penulis menerima berbagai dukungan, bantuan, bimbingan, dan nasehat yang sangat berarti dari banyak pihak selama proses penelitian dan penyusunan skripsi ini. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua, Bapak Jayadi dan Ibu Sri Puji Rini yang telah memberikan dukungan moril dan materil, dan menjadi motivasi utama penulis dalam menyelesaikan skripsi ini.
2. Bapak Dr. Eng. Ahmad Kusumaatmaja, S.Si., M.Sc. sebagai dosen pembimbing yang sudah memberikan arahan, bimbingan, dan masukan sehingga penelitian dan penulisan skripsi ini dapat berjalan lancar.
3. Bapak Dr. Eko Sulistya, M.Si. dan Bapak Prof. Sholihun, S.Si., M.Sc., Ph.D.Sc**.** sebagai dosen penguji yang memberikan saran dan masukan yang bermanfaat bagi penulis.
4. Bude Ani dan keluarga yang telah menerima penulis untuk tinggal sementara selama masa kuliah dan menjadi sosok orang tua yang memberikan perhatian serta dukungan.
5. Lionel Rangga Milanisti, adik penulis yang telah berjuang bersama dalam menuntut ilmu, memberikan semangat, dan menjadi sumber motivasi untuk menyelesaikan skripsi ini.
6. Teman-teman mahasiswa fisika angkatan 2020 yang telah menemani dan banyak membantu selama masa perkuliahan.

7. Teman-teman masa sekolah yang selalu ada di saat sulit maupun senang, terutama saat ingin bermain untuk beristirahat dari rutinitas perkuliahan.
8. Diwantara Anugrah Putra (Tara Arts) dan Gema Cita Andika (Gema Show), konten kreator bersaudara yang selalu menghibur dan memberi motivasi.
9. Diri penulis sendiri yang telah melalui segala tantangan dan kesulitan selama perkuliahan. Terima kasih karena tetap melangkah meski sering merasa lelah. Terima kasih karena selalu percaya dengan kemampuan diri sendiri untuk melakukan yang terbaik.

Penulis menyadari bahwa masih terdapat banyak kekurangan dalam penyusunan dan penulisan skripsi ini. Oleh karena itu, penulis menerima setiap kritik dan saran sebagai bahan evaluasi dalam menghasilkan karya yang lebih baik di masa yang akan datang. Penulis berharap supaya skripsi ini dapat memberikan manfaat bagi semua pihak.

Yogyakarta, 15 November 2024

Rui Costa Raka Milanisti

DAFTAR ISI

HALAMAN PENGESAHAN i
PERNYATAAN BEBAS PLAGIASI ii
MOTTO DAN PERSEMBAHAN iii
PRAKATA iv
DAFTAR ISI vi
DAFTAR TABEL viii
DAFTAR GAMBAR ix
INTISARI x
ABSTRACT xi
BAB I PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Masalah 4
1.3. Batasan Masalah 4
1.4. Tujuan Penelitian 5
1.5. Manfaat Penelitian 5
1.6. Sistematika Penulisan 6
BAB II TINJAUAN PUSTAKA 7
BAB III LANDASAN TEORI 13
3.1. Demensia 13
3.2. Penyakit Alzheimer 13
3.3 *Magnetic Resonance Imaging* (MRI) 14
3.3. Kecerdasan Buatan 15
3.4. Pemelajaran Mesin (*Machine Learning*) 15
3.5. Computer Vision 17
3.6. Jaringan Saraf Tiruan 18
3.7. *Deep Learning* 20
3.8. *Convolutional Neural Network* (CNN) 24
3.9. *Transfer Learning* 29
3.10. *Residual Network* (ResNet) 30
3.11. Metrik Evaluasi 31
3.12. Python 33

BAB IV METODE PENELITIAN 35
4.1. Alat dan Bahan Penelitian 35
4.2. Prosedur Penelitian 37
4.3. Jadwal Penelitian 42
BAB V HASIL PENELITIAN DAN PEMBAHASAN 43
5.1. Persiapan Dataset 43
5.2. *Exploratory Data Analysis* (EDA) 43
5.3. Implementasi Pra-pemrosesan Data 45
5.4. Implementasi Pembagian Data 46
5.5. Hasil Pemodelan 47
BAB VI KESIMPULAN DAN SARAN 58
6.1. Kesimpulan 58
6.2. Saran 58
DAFTAR PUSTAKA 59
LAMPIRAN 64

DAFTAR TABEL

Tabel 2.1 Ringkasan Penelitian Terdahulu 10
Tabel 4.1 Distribusi data 37
Tabel 4.2 Jadwal penelitian 42
Tabel 5.1 Pembagian dataset 47
Tabel 5.2 Data akurasi dan *loss* pada tiap *epoch* pelatihan 49
Tabel 5.3 Metrik evaluasi model 53
Tabel 5.4 Perbandingan dengan penelitian terdahulu 56

DAFTAR GAMBAR

Gambar 3.1 Perbandingan antara otak sehat dan penderita *Alzheimer* (Dan dkk., 2022) 14
Gambar 3.2 Sampel citra MRI 15
Gambar 3.3 Hubungan Neuron Biologis dan Buatan (Harkiran78, 2023) 19
Gambar 3.4 Arsitektur jaringan saraf tiruan (Zhang dkk., 2019) 20
Gambar 3.5 Ilustrasi hasil perhitungan softmax (Belagatti, 2024) 22
Gambar 3.6 Kurva fungsi aktivasi ReLU (Jia, 2023) 22
Gambar 3.7 Ilustrasi arsitektur CNN secara umum (Mishra, 2020) 24
Gambar 3.8 Komputer melihat gambar sebagai suatu array representasi piksel (Biswal, 2023) 25
Gambar 3.9 Proses konvolusi (Alzubaidi dkk., 2021) 26
Gambar 3.10 Ilustrasi operasi *max pooling*, *average pooling*, dan *global average pooling* (Alzubaidi dkk., 2021) 27
Gambar 3.11 Ilustrasi dropout *regularization* (Lina, 2019) 28
Gambar 3.12 Ilustrasi fully-connected layer (Alzubaidi dkk., 2021) 28
Gambar 3.13 Traditional Machine Learning vs Transfer Learning (Wijaya dkk., 2021) 29
Gambar 3.14 Blok residual *learning* dengan *identity mapping* (He dkk., 2015) 30
Gambar 3.15 *Confusion Matrix* (Karra, 2020) 31
Gambar 4.1 Dataset MRI otak 36
Gambar 4.2 Diagram alir penelitian 38
Gambar 5.1 Pengunduhan dataset 43
Gambar 5.2 Grafik distribusi dataset 44
Gambar 5.3 Jumlah gambar pada setiap kategori dan totalnya 44
Gambar 5.4 Ekstensi *file* dan jumlah datanya 45
Gambar 5.5 Arsitektur model 48
Gambar 5.6 Grafik akurasi pelatihan dan validasi 51
Gambar 5.7 Grafik kerugian (*loss*) pelatihan dan validasi 52
Gambar 5.8 *Confusion matrix* hasil pengujian 53

INTISARI

DETEKSI DINI *ALZHEIMER* MENGGUNAKAN *DEEP LEARNING* DENGAN ARSITEKTUR RESNET152V2 PADA CITRA *MAGNETIC RESONANCE IMAGING* (MRI)

Oleh

RUI COSTA RAKA MILANISTI

20/459212/PA/19873

Penyakit *Alzheimer* merupakan penyakit neurodegeneratif yang ditandai dengan penurunan fungsi kognitif secara progresif dan tidak dapat dipulihkan. Deteksi dini *Alzheimer* menjadi sangat penting untuk penanganan yang lebih efektif. Salah satu pendekatan yang menjanjikan dalam mendeteksi *Alzheimer* pada tahap awal adalah dengan menggunakan teknik *machine learning,* terutama dengan model *deep learning* seperti ResNet152V2. Pada penelitian ini, dilakukan pengembangan model *deep learning* menggunakan ResNet152V2 untuk mendeteksi tahap awal Alzheimer berdasarkan citra *Magnetic Resonance Imaging* (MRI). Model ini dilatih dengan menggunakan metode *transfer learning*, yang memanfaatkan bobot dari model yang sudah dilatih sebelumnya untuk meningkatkan akurasi. Hasil akhir dari penelitian ini diharapkan dapat menghasilkan model yang memiliki akurasi tinggi dalam mendeteksi tahap awal penyakit *Alzheimer*, sehingga dapat membantu dalam proses diagnosis klinis yang lebih cepat dan akurat.

Kata kunci : *Alzheimer*, *deep learning*, ResNet152v2, MRI, *transfer learning*

ABSTRACT

EARLY DETECTION OF ALZHEIMER'S DISEASE USING DEEP LEARNING WITH RESNET152V2 ARCHITECTURE ON MAGNETIC RESONANCE IMAGING (MRI) SCANS

Oleh

RUI COSTA RAKA MILANISTI

20/459212/PA/19873

Alzheimer's disease is a neurodegenerative disorder characterized by progressive and irreversible cognitive decline. Early detection of Alzheimer's is crucial for more effective treatment. One promising approach to early-stage detection is through machine learning techniques, particularly with deep learning models like ResNet152V2. This study focuses on developing a deep learning model using ResNet152V2 to detect early stages of Alzheimer's based on Magnetic Resonance Imaging (MRI) scans. The model is trained using transfer learning, leveraging pre-trained model weights to improve accuracy. The final result of this study aims to produce a high-accuracy model in detecting early-stage Alzheimer's disease, thereby facilitating faster and more accurate clinical diagnoses.

Keywords : Alzheimer's, deep learning, ResNet152v2, MRI, transfer learning

BAB I
PENDAHULUAN

1.1. Latar Belakang

Otak adalah organ vital yang mengendalikan seluruh fungsi tubuh manusia, mulai dari pemikiran, ingatan, emosi, hingga koordinasi motorik. Sebagai organ yang kompleks dan vital, otak bertanggung jawab atas berbagai fungsi kognitif yang memungkinkan manusia untuk berinteraksi dengan lingkungan dan menjalani kehidupan sehari-hari. Namun, otak juga rentan terhadap penyakit degeneratif yang dapat merusak fungsi-fungsi tersebut. Salah satu penyakit yang paling menonjol dan umum terjadi adalah penyakit *Alzheimer*.

Penyakit *Alzheimer* (*Alzheimer's Disease*/AD) adalah penyakit degeneratif otak dan penyebab paling umum (60%-70%) dari demensia, sebuah kondisi hilangnya ingatan yang bisa timbul bersama gejala gangguan perilaku maupun psikologis pada seseorang (Feng dkk., 2020). Nama penyakit *Alzheimer* berasal dari Dr. Alois Alzheimer, seorang dokter Jerman yang pertama kali mengidentifikasinya pada tahun 1906 (Borden, 2021). Dr. Alzheimer mengamati perubahan pada otak seorang pasien perempuan yang meninggal setelah mengalami kehilangan ingatan, masalah berbahasa, dan perilaku yang tidak dapat dijelaskan. Penyakit *Alzheimer* memengaruhi ingatan, kemampuan berpikir, dan perilaku. Hal ini ditandai dengan penurunan memori, kemampuan berbahasa, kemampuan pemecahan masalah, dan keterampilan kognitif lainnya, gejala-gejalanya dapat menjadi semakin parah sehingga mengganggu aktivitas sehari-hari. Penurunan ini terjadi karena sel-sel saraf (neuron) di bagian otak yang terlibat dalam fungsi kognitif telah rusak dan tidak lagi berfungsi normal (Sianturi, 2021).

Menurut *Alzheimer's Association*, ada tiga tahap umum yang digunakan untuk menentukan tingkat keparahan gejala dan perkembangan penyakit *Alzheimer*. Pada stadium awal (*Early*/*Mild stage*), hanya ada sedikit gejala

yang terlihat, seperti perubahan perilaku. Stadium 2 (*Moderate stage)*, gejala yang muncul dapat berupa menurunnya kemampuan sensorik, berbahasa, dan kesadaran. Hilangnya ingatan dan kebingungan juga dapat terlihat dalam beberapa kasus pada stadium ini. Pada stadium akhir, penderita mengalami gangguan otak yang parah. Jaringan otak menyusut dengan sangat parah dan neuron tidak dapat berkomunikasi. Pasien pada tahap ini mengalami demensia berat dan terkadang halusinasi (Ellis dan Yetman, 2024).

Setiap tiga detik, satu orang di dunia mengalami demensia. Insiden demensia *Alzheimer* di seluruh dunia meningkat dengan cepat dan saat ini diperkirakan mendekati 46,8 atau 50 juta orang yang didiagnosis dengan demensia di dunia, 20,9 juta di Asia Pasifik (Alzheimer's Disease International, 2019), ada sekitar 10 juta kasus baru setiap tahun. Di Indonesia sendiri, diperkirakan ada sekitar 1,2 juta orang dengan demensia pada tahun 2016, yang akan meningkat menjadi dua juta di 2030 dan empat juta orang pada tahun 2050 (Alzheimer Indonesia, 2019). Peningkatan ini menunjukkan urgensi untuk menemukan metode diagnosis yang lebih efektif dan efisien.

Para dokter menggunakan beberapa metode untuk menentukan apakah seseorang mengalami kehilangan ingatan akibat AD. Metode pertama dilakukan dengan mengajukan sejumlah pertanyaan mengenai kondisi kesehatan, konsumsi obat-obatan, pola makan, riwayat medis, aktivitas harian, dan perubahan perilaku kepada pasien atau orang-orang terdekatnya. Kemudian akan dilakukan tes ingatan, evaluasi kemampuan memecahkan masalah, kemampuan fokus dan perhatian, menghitung, dan berbahasa. Tes medis standar juga mungkin dilakukan untuk mengidentifikasi penyebab lain yang menyebabkan gejala gejala tersebut. Terakhir dokter dapat menyarankan untuk melakukan pemindaian otak (Herrmann, 2016).

Penyusutan ukuran (artrofi) pada bagian otak seperti hippocampus, pembesaran ruang kosong (ventrikel), dan penyusutan korteks adalah tanda-tanda yang sensitif untuk *Alzheimer* (Pini dkk., 2016). Untuk memeriksa hal ini, dokter menggunakan teknologi pencitraan canggih seperti CT scan, MRI,

atau PET scan. Teknologi-teknologi ini membantu dokter melihat perubahan struktur otak untuk mengonfirmasi kehadiran dan perkembangan *Alzheimer*, serta memastikan bahwa gejala yang dialami tidak disebabkan oleh masalah kesehatan lain (Gunawardena dkk., 2017).

Diagnosis penyakit *Alzheimer* sering kali sulit, terutama pada tahap awal, metode wawancara dan kuesioner untuk mendeteksi *Alzheimer* tahap awal memiliki beberapa kelemahan. Jawaban pasien dan orang terdekatnya seringkali subjektif, dan gejala awal yang halus dapat sulit diidentifikasi. Sementara itu tingkat akurasi diagnosis dengan pencitraan otak sangat bergantung pada pengalaman radiolog yang melakukan interpretasi hasil pemindaian otak (Klöppel dkk., 2008). Misalnya, dua radiolog yang berbeda mungkin memberikan diagnosis yang berbeda berdasarkan pemindaian yang sama. Ini menunjukkan kebutuhan akan metode diagnosis yang lebih objektif dan andal.

Dalam beberapa tahun terakhir, telah banyak dilakukan penelitian mengenai diagnosis otomatis penyakit *Alzheimer* menggunakan berbagai metode. Beberapa klasifikasi pola telah dicoba untuk membedakan subjek berdasarkan data neuroimaging yang berbeda. Metode ekstraksi fitur dan klasifikasi yang berbeda telah digunakan dalam studi-studi terkini ini. Dengan perkembangan teknologi, khususnya dalam bidang *deep learning*, telah memungkinkan pembuatan model yang dapat mengidentifikasi dan mengklasifikasikan tahapan penyakit AD dengan lebih akurat dan efisien. Berbagai model *deep learning* diuji untuk menghasilkan sebuah program klasifikasi yang cepat dan akurat. Penggunaan teknologi akan mempermudah diagnosa awal pada penyakit *Alzheimer*.

Tujuan dari penelitian ini adalah untuk mengembangkan model *deep learning* yang mampu mengklasifikasikan tahapan penyakit *Alzheimer* dengan menggunakan gambar pindaian MRI otak. Model akan mempelajari fitur-fitur dari gambar MRI. Dengan demikian, diharapkan bahwa model ini dapat membantu dalam mendeteksi penyakit AD lebih awal dan dengan akurasi

yang lebih tinggi, serta memberikan alat bantu yang berharga bagi para profesional medis dalam pengambilan keputusan klinis.

Deteksi dan Klasifikasi serupa telah dilakukan dan dipublikasikan hasilnya oleh Yildirim dan Cinar (2020). Dalam penelitian tersebut, digunakan model gabungan dengan dasar arsitektur Resnet50, menghasilkan akurasi deteksi sebesar 90%. Buvaneswari dan Gayathri (2021) melakukan penelitian yang sama dengan menggunakan model dengan arsitektur Resnet101, akurasi deteksi yang dicapai sebesar 96,3%.

Dalam penelitian ini akan digunakan model dengan arsitektur model Resnet152V2 yang memanfaatkan metode *transfer learning*. Meskipun hanya empat kondisi/tahapan penyakit *Alzheimer*, diharapkan penelitian ini dapat menjadi referensi bagi akademisi maupun praktisi untuk dapat melakukan deteksi AD sedini mungkin. Selain itu, penelitian ini juga diharapkan berguna sebagai media edukasi mengenai *Alzheimer* dan metode klasifikasi *deep learning* untuk masyarakat.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka diperoleh rumusan masalah adalah sebagai berikut:

1. Bagaimana cara memanfaatkan metode *transfer learning* dengan arsitektur model ResNet152V2 untuk mengembangkan model yang efektif untuk deteksi tahap awal penyakit *Alzheimer*?
2. Bagaimana kinerja model dengan *transfer learning* menggunakan arsitektur model Resnet152V2 pada deteksi tahapan awal penyakit *Alzheimer*?

1.3. Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Dataset merupakan data gambar hasil pemindaian kepala dengan metode *Magnetic Resonance Imaging* (MRI) yang dibagi dalam empat kelas dengan distribusi data yang tidak merata.
2. Penelitian hanya digunakan untuk mengklasifikasikan empat kondisi awal penyakit *Alzheimer* dari hasil pemindaian MRI yaitu *Non-Demented*, *Very Mild Demented*, *Mild Demented*, dan *Moderate Demented*.
3. Penelitian ini hanya berfokus pada hasil satu jenis model saja yaitu Resnet152V2 dengan *library* TensorFlow dan Keras.

1.4. Tujuan Penelitian

Penelitian ini dilakukan dengan tujuan sebagai berikut:

1. Mengembangkan model *deep learning* yang efektif untuk mendeteksi dan mengklasifikasikan tahap awal penyakit *Alzheimer* menggunakan arsitektur ResNet152V2 dengan *transfer learning*.
2. Mengevaluasi akurasi model dengan arsitektur ResNet152V2 dalam mendeteksi tahap awal penyakit *Alzheimer*.

1.5. Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Membantu penulis dalam memahami *deep learning*, pengolahan citra, serta memberikan pengalaman melakukan penelitian.
2. Meningkatkan kesadaran tentang deteksi dini *Alzheimer* dan penggunaan teknologi dalam kesehatan.
3. Memberikan kontribusi pada pengembangan teknologi deteksi penyakit *Alzheimer* menggunakan metode *transfer learning*.
4. Menjadi referensi bagi peneliti selanjutnya dalam topik serupa.

1.6. Sistematika Penulisan

Usulan penelitian ini terdiri dari lima bab yaitu pendahuluan, tinjauan pustaka, landasan teori, metode penelitian, serta daftar pustaka.

BAB I merupakan pendahuluan yang berisi latar belakang mengapa penelitian ini dilakukan, rumusan masalah yang diperoleh dari latar belakang, batasan masalah yang ditetapkan dalam penelitian, tujuan dilakukannya penelitian, manfaat penelitian, dan sistematika penulisan yang berisi garis besar penulisan.

BAB II berisi tinjauan pustaka yang membahas penelitian sebelumnya dengan topik, metode, atau objek yang sama.

BAB III merupakan landasan teori yang membahas konsep dasar dan prinsip yang berkaitan dengan penelitian ini.

BAB IV terdiri atas metode penelitian yang berisi deskripsi umum penelitian, alat dan bahan yang digunakan dalam penelitian, dan prosedur penelitian.

BAB V berisi tempat dan jadwal penelitian yang memaparkan mengenai tempat dilakukannya penelitian dan rincian rencana waktu dan kegiatan penelitian ini.

BAB II
TINJAUAN PUSTAKA

Alzheimer merupakan tipe demensia yang umum terjadi pada usia lanjut. Penyakit ini adalah penyakit neurologis progresif yang menyebabkan kerusakan sel otak. Karena penyakit ini bersifat progresif, masalah yang diakibatkannya akan bertambah seiring waktu. Oleh karena itu, sangat penting melakukan deteksi penyakit *Alzheimer* pada tahapan awal agar perawatan bisa segera dilakukan (Yildirim dan Cinar, 2020).

Penelitian yang dilakukan Yildirim dan Cinar (2020) memastikan hasil pemindaian MRI menunjukkan kondisi *Alzheimer* atau bukan, dan mengidentifikasi tahapan penyakitnya. Penelitian ini menggunakan empat kelompok tahapan penyakit *Alzheimer* yaitu *Non-Demented*, *Very Mild Demented*, *Mild Demented*, dan *Moderate Demented* dengan total data berjumlah 5.121 gambar. Jumlah data pada tiap kelas tidak sama dan tidak dilakukan penyamarataan distribusi data. Data kemudian dibagi menjadi data pelatihan dan data uji dengan rasio 4:1. Model *deep learning* yang digunakan memiliki dasar arsitektur ResNet50 yang ditambahkan beberapa lapisan tambahan sehingga menghasilkan model metode *hybrid* (gabungan). Model yang menggunakan ResNet50 murni berhasil mengelompokkan dengan benar 78 dari 100 gambar, sehingga total akurasi yang dicapai sebesar 78%. Sementara itu, model gabungan berhasil mengidentifikasi 90 dari 100 data uji dengan benar, sehingga akurasi yang dicapai sebesar 90%.

Buvaneswari dan Gayathri (2021) melakukan penelitian dengan mengembangkan model untuk mengklasifikasi gambar MRI ke dalam tiga kelompok yaitu *Cognitive Normal* (CN), *Mild Cognitive Impairment* (MCI), dan *Alzheimer Disease* (AD). Dataset yang digunakan bersumber dari *Alzheimer's Disease Neuroimaging Initiative* (ADNI), berisi 240 gambar dengan distribusi data sama rata. Data disegmentasi *dengan* SegNet sebelum diklasifikasi menggunakan model dengan arsitektur ResNet101. Model yang dikembangkan menunjukkan akurasi klasifikasi sebesar 96,3%

Pada penelitian yang dilakukan oleh Ullah dan Jamjoom (2023) dilakukan pengembangan model untuk mendeteksi *Alzheimer* dan tahapan penyakitnya. Dataset berupa gambar citra MRI dengan total 6.400 gambar, dengan distribusi yang tidak sama rata. Dataset akan melalui proses augmentasi dengan teknik *rotation*, *shearing*, *zooming*, serta *horizontal* dan *vertical flip*. Model yang digunakan terdiri dari tiga lapis *convolutional layer*, dua lapis *pooling layer*, dua *fully connected layer*, dan satu *output layer*. Model mencapai tingkat akurasi yang baik yaitu sebesar 99,38%.

Gunawardena dkk. (2017) dalam penelitiannya membandingkan beberapa metode dalam deteksi AD. *Metode* tersebut diantaranya metode SVM, pengklasifikasian Bayes, dan Jaringan Saraf Buatan (*Artificial Neural Network*/ANN). Ia mengulas bahwa sebagian besar penelitian yang telah dilakukan tidak berfokus pada metode untuk deteksi awal (pra-deteksi) AD. Metode SVM yang sebelumnya banyak digunakan dinilai tidak ideal untuk deteksi kasus AD awal dengan gejala ringan hingga menengah. Sementara itu, metode *Convolutional Neural Network* (CNN) menunjukkan hasil yang paling baik di antara metode klasifikasi gambar yang lain.

Pada eksperimen CNN, semua gambar yang digunakan sebagai masukan untuk model diubah ukurannya menjadi 160 piksel x 160 piksel, kemudian disimpan dalam matriks dalam format *flattened*, dan diberi label. Arsitektur model yang digunakan terdiri dari dua *convolutional layer,* satu *pooling layer*, dan *fully connected layer*. Model diimplementasikan dengan *library deep learning python Theano* dan *Keras*. Menggunakan dataset berisi 1.615 gambar (1.292 untuk training, 323 untuk testing), didapatkan nilai sensitifitas dan spesifisitas model berturut-turut sebesar 96% dan 98% (Gunawardena dkk., 2017)

Pada penelitian yang dilakukan oleh Venkataramanan dkk. (2019), dilakukan pengembangan model *deep learning* untuk mengidentifikasi dan mengklasifikasi penyakit pada tanaman dengan mengamati tekstur daunnya. Dalam penelitian tersebut, digunakan teknik deteksi objek YOLOv3 untuk mengekstrak fitur daun dari gambar masukan. Gambar daun yang telah diekstrak kemudian dianalis melalui

transfer learning dengan menggunakan model ResNet18. Lapian dalam model mengidentifikasi jenis daun dan penyakit yang mungkin dimilikinya. Nilai akurasi yang dicapai model adalah sebesar 96%.

Aref dan Kareem (2021) dalam penelitiannya membandingkan beberapa model untuk mendeteksi infeksi Covid-19 dengan menggunakan citra X-ray dada pasien. Model *Convolutional Neural Network* yang dibandingkan antara lain ResNet50, ResNet101, ResNet152, InceptionV3 dan Inception-ResNetV2. Model akan digunakan untuk mengklasifikasikan data ke dalam empat kelas yaitu Covid-19, normal (sehat), pneumonia virus, dan pneumonia bakteri. Berdasarkan percobaan diketahui model yang mencapai performa klasifikasi tertinggi adalah ResNet50 dengan akurasi sebesar 96,1% pada dataset 1 dan 99,5% pada dataset 2.

Berdasarkan hasil penelitian terdahulu yang disajikan di atas, klasifikasi gambar menggunakan *deep learning* dengan metode CNN adalah metode yang cepat dan tepat untuk berbagai tugas klasifikasi. Berdasarkan penelitian yang dilakukan Aref dan Kareem (2021), variasi *pre-trained* model ResNet menghasilkan akurasi yang sangat baik. Yildirim dan Cinar (2020) menggunakan model dasar ResNet50 untuk melakukan klasifikasi dan memperoleh hasil yang memuaskan. Selanjutnya, penulis akan melaksanakan penelitian dengan mengembangkan model klasifikasi gambar MRI otak untuk mendeteksi empat tahapan penyakit Alzheimer, yaitu *Non-Demented*, *Very Mild Demented*, *Mild Demented*, dan *Moderate Demented*, dengan dataset yang lebih besar daripada yang digunakan pada penelitian Yildirim dan Cinar (2020). Metode yang digunakan adalah *transfer learninig* dengan arsitektur *pre-trained* model ResNet152V2 dengan *optimizer* Adamax. Ringkasan dari penelitian-penelitian yang telah dibahas dapat dilihat pada tabel 2.1 berikut.

Tabel 2.1 Ringkasan Penelitian Terdahulu

Deteksi	**Peneliti**	**Dataset**	**Metode**	**Hasil**
Deteksi empat tahapan awal penyakit *Alzheimer* yaitu *Non-Demented*, *Very Mild Demented*, *Mild Demented*, dan *Moderate Demented*.	Yildirim & Cinar. (2020)	Dataset terdiri dari gambar pindaian MRI dengan total 5.121 gambar.	• Deteksi dan klasifikasi empat tahapan awal *Alzheimer*. • Menggunakan dua arsitektur model yaitu model ResNet50 murni dan model ResNet50 dengan lapisan tambahan.	• Model yang dibangun dengan arsitektur ResNet50 murni mendapatkan akurasi sebesar 78%. • Model ResNet50 dengan lapisan tambahan mendapat akurasi yang lebih baik, yaitu 90%.
Deteksi tiga kondisi pasien yaitu Normal, Kelainan kognitif ringan, dan *Alzheimer*.	Buvaneswari & Gayathri. (2021)	Dataset terdiri dari tiga kelompok dengan total 240 gambar.	• Segmentasi data menggunakan SegNet untuk ekstraksi fitur gambar • Model klasifikasi menggunakan arsitektur ResNet101.	• Model yang dikembangkan menunjukkan akurasi klasifikasi sebesar 96,3%.
Deteksi empat tahapan awal penyakit	Ullah & Jamjoom. (2023)	Dataset berupa gambar citra MRI dengan total 6.400	• Dataset akan melalui proses augmentasi dengan	• Model mencapai tingkat akurasi yang baik

Alzheimer yaitu *Non Demented*, *Very Mild Demented*, *Mild Demented*, dan *Moderate Demented*.		gambar, dengan distribusi yang tidak sama rata.	teknik *rotation*, *shearing*, *zooming*, serta *horizontal* dan *vertical flip*. • Model yang digunakan terdiri dari tiga lapis *convolutional layer*, dua lapis *pooling layer*, dua *fully connected layer*, dan satu *output layer*.	yaitu sebesar 99,38%.
Deteksi kondisi pasien ke dalam tiga kelas AD, MCI, dan NL.	Gunawardena dkk. (2017)	Menggunakan dataset berisi 1.615 gambar (1.292 untuk training, 323 untuk testing).	• Data melalui *pre-processing* dengan menggunakan *library* OpenCV. • Gambar dipertajam menggunakan *unsharp masking filter* dan diterapkan algoritma deteksi tepian *Canny.*	• Nilai sensitifitas dan spesifisitas yang dicapai model berturut-turut sebesar 96% dan 98%.

Identifikasi penyakit pada tanaman dengan mengamati daunnya.	Venkatarama nan dkk. (2019)	Dataset berisi gambar daun berbagai tanaman dengan total 36.148 gambar.	• Menggunaka n teknik YOLOv3 untuk deteksi fitur daun dari gambar • Model yang menganalisis gambar menggunaka n arsitektur ResNet18.	• Model mencapai nilai akurasi sebesar 96%.
Membandin gkan beberapa algoritma untuk deteksi infeksi Covid-19 dengan gambar X-Ray dada pasien	Aref & Kareem. (2021)	Menggunaka n dua dataset X-Ray berbeda	• Model yang dibandingka n memiliki arsitektur ResNet50, ResNet101, ResNet152, InceptionV3 dan Inception-ResNetV2.	• Model yang mencapai performa klasifikasi tertinggi adalah ResNet50 dengan akurasi sebesar 96,1% pada dataset 1 dan 99,5% pada dataset 2.

BAB III
LANDASAN TEORI

Teori dan informasi ilmiah sangat penting untuk membantu pemahaman dan mendukung penelitian. Dengan dasar yang kuat, penelitian dapat lebih terarah dan menghasilkan kesimpulan yang tepat. Penjelasan mengenai konsep, teori, dan data ilmiah yang relevan akan membantu memperkuat dasar penelitian ini. Selanjutnya, teori-teori dan informasi ilmiah terkait akan dijelaskan untuk mendukung proses penelitian.

3.1. Demensia

Demensia adalah penurunan kemampuan berpikir, mengingat, dan bernalar yang mengganggu aktivitas sehari-hari seseorang. Tingkat keparahannya bervariasi, mulai dari tahap ringan, di mana pengaruhnya masih kecil, hingga tahap berat, di mana seseorang perlu bantuan penuh dari orang lain untuk melakukan kegiatan dasar sehari-hari. Penyebab demensia dapat berbeda tergantung perubahan yang terjadi pada otak. Beberapa bentuk demensia antara lain *Alzheimer*, *Lewy body dementia*, *frontotemporal disorder*s, dan *vascular dementia* (Susanti dkk., 2024).

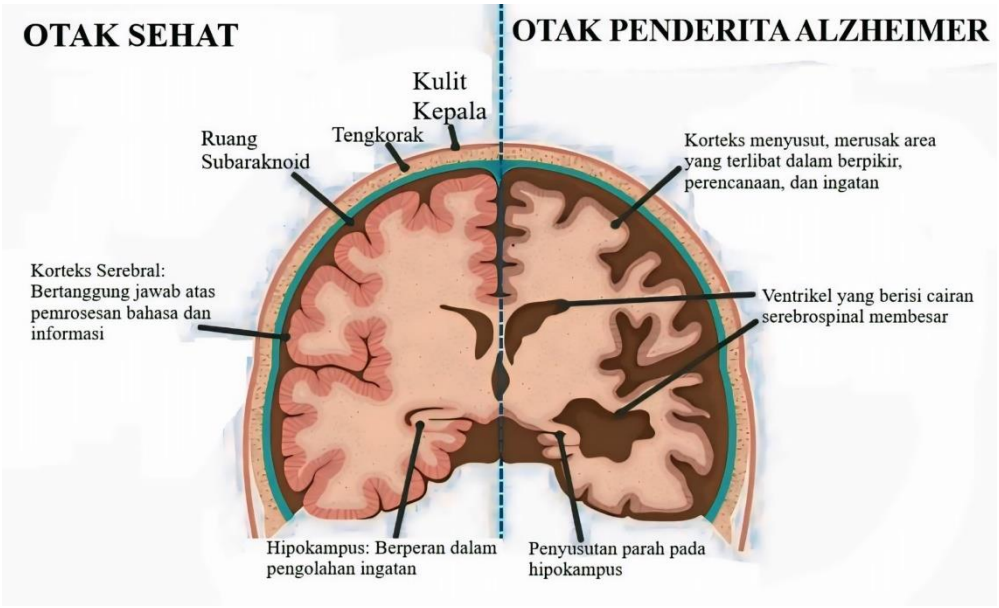
3.2. Penyakit Alzheimer

Alzheimer merupakan penyebab terbanyak dari kasus demensia, menyumbang mayoritas dari seluruh kasus demensia yang terjadi. Penyakit ini menyebabkan penurunan kemampuan kognitif yang signifikan, seperti kemampuan berpikir, mengingat, dan bernalar, yang akhirnya mengganggu aktivitas sehari-hari penderitanya. Penurunan ini terjadi karena sel-sel saraf (neuron) di bagian otak yang terlibat dalam fungsi kognitif telah rusak dan tidak lagi berfungsi normal (Sianturi, 2021).

Dokter menggunakan berbagai metode untuk menentukan apakah seseorang mengalami kehilangan ingatan akibat Alzheimer. Mereka akan menanyakan tentang kesehatan, obat-obatan, riwayat medis, aktivitas harian,

dan perubahan perilaku pasien. Setelah itu, tes memori dan kemampuan berpikir dilakukan. Pemeriksaan medis standar juga dijalankan untuk menyingkirkan penyebab lain, dan pemindaian otak mungkin disarankan untuk analisis lebih mendalam (Herrmann, 2016).

Penyusutan ukuran (artrofi) pada bagian otak seperti hippocampus, pembesaran ruang kosong (ventrikel), dan penyusutan korteks adalah tanda-tanda yang sensitif untuk *Alzheimer* (Pini dkk., 2016). Untuk memeriksa hal ini, dokter menggunakan teknologi pencitraan canggih seperti CT (*Computed Tomography*) scan, MRI (*Magnetic Resonance Imaging)*, atau PET (*Positron Emission Tomography*) scan. Teknologi-teknologi ini membantu dokter melihat perubahan struktur otak untuk mengonfirmasi kehadiran dan perkembangan *Alzheimer*, serta memastikan bahwa gejala yang dialami tidak disebabkan oleh masalah kesehatan lain (Gunawardena dkk., 2017).



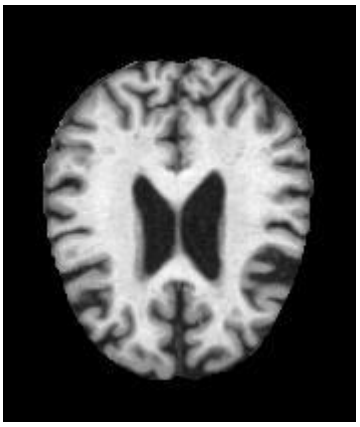
Gambar 3.1 Perbandingan antara otak sehat dan penderita *Alzheimer* (Dan dkk., 2022)

3.3 *Magnetic Resonance Imaging* (MRI)

Magnetic Resonance Imaging (MRI) adalah teknik diagnostik medis yang memanfaatkan medan magnet dan gelombang radio untuk menghasilkan citra internal organ dan jaringan tubuh manusia (Coskun, 2011). Teknologi ini

memungkinkan pencitraan tiga dimensi yang rinci dari struktur dalam tubuh, sehingga mendukung diagnosis yang lebih tepat dan pengobatan yang sesuai.

MRI memanfaatkan sifat magnetik alami tubuh untuk bekerja. Proses MRI dimulai dengan menciptakan medan magnet kuat di area yang akan diperiksa. Medan ini mempengaruhi proton dalam tubuh, sementara gelombang radio memberikan energi kepada proton tersebut. Setelah gelombang radio dihentikan, proton melepaskan energi dalam bentuk sinyal frekuensi radio. Sinyal ini kemudian ditangkap dan diolah oleh mesin MRI untuk menghasilkan gambar struktur tubuh (Berger, 2002).



Gambar 3.2 Sampel citra MRI

3.3. Kecerdasan Buatan

Kecerdasan Buatan (*Artificial Intelligence*/AI) adalah sistem komputer yang dapat menjalankan tugas-tugas yang biasanya memerlukan kecerdasan manusia. Teknologi ini memungkinkan sistem untuk membuat keputusan dengan menganalisis dan menggunakan data yang tersedia. Proses AI melibatkan pemelajaran, penalaran, dan koreksi diri, mirip dengan cara manusia menganalisis sebelum membuat keputusan (Kurniawan dkk., 2023). Kecerdasan buatan dapat diaplikasikan ke berbagai bidang diantaranya adalah *gaming, Natural Language Processing, Expert System, Vision System, Speech Recognition,* dan *Handwriting Recognition* (Shoumi dkk., 2022).

3.4. Pemelajaran Mesin (*Machine Learning*)

Machine learning adalah bagian dari domain kecerdasan buatan yang difokuskan pada penggunaan algoritma dan metode tertentu untuk memungkinkan komputer melakukan prediksi, pengenalan pola, dan klasifikasi dari data yang diberikan (Lestari dan Rahayu, 2023). Inti dari pemelajaran mesin adalah kemampuan sistem untuk belajar dari pengalaman. Setelah diberikan sejumlah contoh pelatihan, sistem harus mampu membuat model umum yang bisa digunakan untuk mengambil keputusan pada kasus baru dengan akurasi yang memadai. Berdasarkan pendekatan ini, ada tiga metode dalam pemelajaran mesin (Amaratunga, 2020):

1. *Supervised Learning*

   Sistem diberi contoh-contoh berlabel (set pelatihan) dan diminta membuat model yang bisa diterapkan pada kasus baru. Set pelatihan ini berfungsi sebagai data acuan yang digunakan oleh sistem untuk mengenali pola dan hubungan antara input dan output. Dengan menggunakan algoritma pemelajaran terawasi, sistem secara bertahap belajar untuk membuat prediksi atau keputusan berdasarkan data baru yang belum pernah dilihat sebelumnya. Proses ini melibatkan optimasi model untuk meminimalkan kesalahan prediksi, sehingga model yang dihasilkan dapat memberikan hasil yang akurat dan andal saat dihadapkan pada data baru di masa mendatang. *Supervised Learning* sangat berguna dalam berbagai aplikasi, seperti klasifikasi gambar, pengenalan suara, dan prediksi penyakit berdasarkan data medis. Beberapa metode pada algoritma ini adalah *decision tree*, *random forest*, *k-nearest neighbor*, dan *logistic regression* (Shoumi dkk., 2022).

2. *Unsupervised Learning*

   Sistem diberi contoh-contoh tanpa label dan diminta menemukan pola dalam data tersebut. Ini cocok untuk menemukan pola tersembunyi. Tanpa label, sistem harus mengidentifikasi kelompok

atau struktur dalam data secara mandiri, sering kali menggunakan teknik seperti pengelompokan (*clustering*) atau pengurangan dimensi (*dimensionality reduction*). Metode ini sangat berguna dalam analisis data eksploratif, di mana peneliti ingin menemukan wawasan atau anomali yang sebelumnya tidak terdeteksi dalam dataset besar. Beberapa contoh algoritmanya antara lain *k-means clustering* dan algoritma apriori (Shoumi dkk., 2022).

3. *Reinforcement Learning*

   Sistem mengambil tindakan dan diberi hadiah (*reward*) atau hukuman (*punishment*) berdasarkan seberapa baik tindakan tersebut sesuai dengan situasi yang diberikan. Sistem belajar tindakan mana yang menghasilkan hadiah terbanyak dalam berbagai situasi. Salah satu algoritma pemelajaran mesin ini adalah *markov decision process* (Shoumi dkk., 2022).

3.5. Computer Vision

Computer Vision adalah teknologi otomatis yang digunakan oleh komputer untuk menganalisis gambar dan video agar dapat memahami dan mendapatkan informasi tentang objek tertentu. Ini adalah kemampuan komputer untuk melihat dan mengenali gambar, dengan ketepatan yang setara atau bahkan lebih baik dari penglihatan manusia (Mulya dkk., 2023). Tugas utama visi komputer adalah tentang bagaimana membuat komputer memahami gambar digital serta data visual dari dunia nyata. Dalam menganalisis informasi dari masukan tersebut dan membuat keputusan melibatkan beberapa proses yaitu sebagai berikut (Arnita dkk., 2022).

1. Pengolahan Sinyal/Citra

   Pengolahan citra digital adalah manipulasi dan interprestasi digital dari citra dengan bantuan komputer. *Input* dari pengolahan citra adalah gambar, sedangkan *output*-nya adalah citra hasil pengolahan.

Citra secara umum merupakan suatu gambar, foto ataupun berbagai tampilan dua dimensi yang menggambarkan suatu visualisasi objek (Arnita dkk., 2022).

2. Klasifikasi Citra

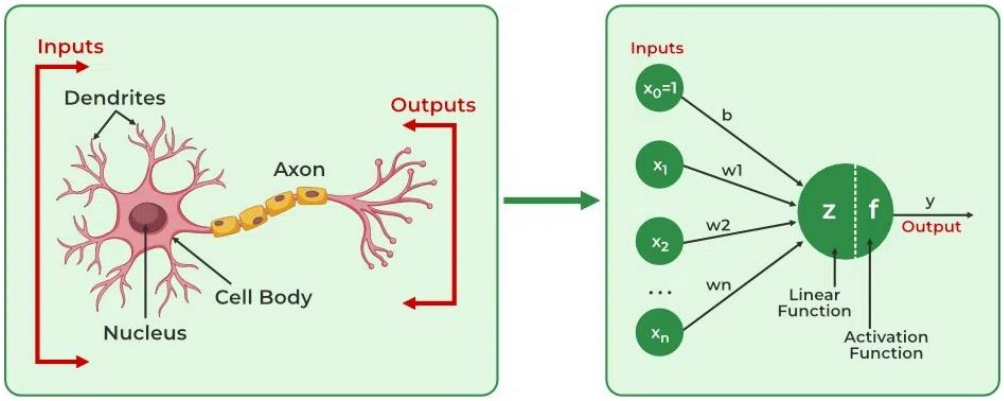
Klasifikasi citra adalah proses analisis data secara numeris yang bertujuan untuk mengidentifikasi objek atau mengenali pola. Pengenalan pola objek (*spectral pattern* recognition) mengevaluasi informasi objek berdasarkan ciri-ciri dalam citra penginderaan jauh untuk menginterpretasi citra digital. Sistem pengenalan pola ini melibatkan pengkategorian piksel dalam citra berdasarkan hubungan spasial antar piksel tersebut (Arnita dkk., 2022).

Computer vision mempunyai keterkaitan dengan beberapa bidang yaitu, *image processing* (pengolahan citra) dan *machine vision* (visi mesin). Ada kesamaan yang signifikan dalam berbagai teknik dan aplikasi yang mencakup tiga bidang ini. Hal ini menunjukan teknik dasar yang digunakan dan dikembangkan kurang lebih sama. Secara luas *computer vision* berhubungan dan diterapkan dengan bidang lain seperti *artificial intelligence (AI),* robotika, otomasi industry, pengolahan sinyal, optik fisik, *neurobiology*, dan lain-lain (Arnita dkk., 2022).

Computer vision sekarang ini telah sering digunakan untuk berbagai hal, contohnya saja mendeteksi wajah pada gambar (*face detection*, mengenali wajah (*facial expression recognition*) dan dalam prakteknya sering digunakan bersama dengan jaringan saraf tiruan (*artificial neural network*) (Dompeipen dan Sompie, 2020).

3.6. Jaringan Saraf Tiruan

Jaringan saraf tiruan (*Artificial Neural Networks/ANN*) terinspirasi oleh model awal pemrosesan sensorik oleh otak. Jaringan saraf tiruan dapat dibuat dengan mensimulasikan jaringan neuron model dalam komputer. Dengan menerapkan algoritma yang meniru proses neuron asli, kita dapat membuat jaringan 'belajar' untuk menyelesaikan berbagai jenis masalah. Neuron ini menerima input dari sejumlah unit lain atau sumber eksternal, memberi bobot pada setiap input, dan menjumlahkannya (Krogh, 2008).

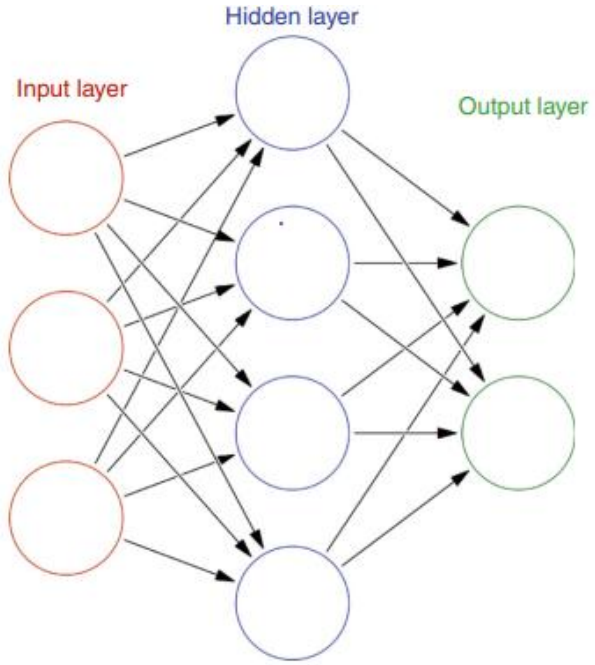


Gambar 3.3 Hubungan Neuron Biologis dan Buatan (Harkiran78, 2023)

Sistem jaringan saraf tiruan terinspirasi oleh struktur dan fungsi neuron manusia, di mana neuron model dalam jaringan saraf tiruan meniru cara neuron asli berkomunikasi dan memproses informasi di otak. Jika keduanya dibandingkan, maka masukan dalam ANN merepresentasikan dendrite, *nodes* (fungsi linear & aktivasi) adalah inti sel, *weights* adalah sinapsis, dan axon adalah keluaran dari sistem ANN. *Nodes* akan menerima nilai masukan dan akan memproses menggunakan fungsi aktivasi untuk menghasilkan nilai keluaran. Dalam jaringan saraf tiruan, fungsi aktivasi adalah fungsi matematika yang memungkinkan jaringan untuk mempelajari pola dan hubungan dalam data, serta memetakan input menjadi output pada (Harkiran78, 2023).

Secara umum jaringan saraf tiruan terdiri dari *input layer, hidden layer,* dan *output layer*. Dalam setiap *input* dan *hidden layer* terdapat neuron-neuron yang berbeda antara satu sama lain (Sumin dan Prihantono, 2020). Neuron-

neuron pada satu lapisan akan dihubungkan dengan lapisan-lapisan lainnya. Informasi yang didapatkan pada sebuah neuron akan disampaikan ke semua lapisan-lapisan yang ada, mulai dari lapisan masukan sampai dengan lapisan keluaran melalui lapisan tersembunyi (hidden layer).



Gambar 3.4 Arsitektur jaringan saraf tiruan (Zhang dkk., 2019)

3.7. *Deep Learning*

Pemelajaran mendalam (*Deep Learning*) adalah subset dari pemelajaran mesin yang berfokus pada algoritma yang terinspirasi oleh pemahaman kita tentang cara kerja otak untuk memperoleh pengetahuan. Ini juga disebut sebagai pemelajaran terstruktur mendalam atau pemelajaran hierarkis. Pemelajaran mendalam membangun ide jaringan saraf tiruan dan mengembangkannya dengan memperdalam jaringan tersebut agar mampu mengonsumsi sejumlah besar data. Melalui jaringan yang lebih dalam, model pemelajaran mendalam memiliki kemampuan untuk mengekstraksi fitur dari data mentah dan "mempelajari" fitur-fitur tersebut sedikit demi sedikit di setiap lapisan, hingga membentuk pengetahuan tingkat tinggi tentang data tersebut (Amaratunga, 2020). Beberapa komponen penting dalam

implementasi *deep learning* diantaranya adalah fungsi aktivasi, *optimizer*, dan *loss function*.

3.7.1. Fungsi Aktivasi

Dalam ranah machine learning dan deep learning, fungsi aktivasi memainkan peran penting dalam kemampuan jaringan saraf untuk membuat keputusan dan prediksi yang kompleks (Belagatti, 2024). Fungsi aktivasi digunakan untuk menghitung bobot masukan dan bias (jika ada) serta memutuskan apakah suatu neuron dapat diaktifkan atau tidak (Panneerselvam, 2024). Fungsi aktivasi yang sering digunakan diantaranya softmax dan ReLU.

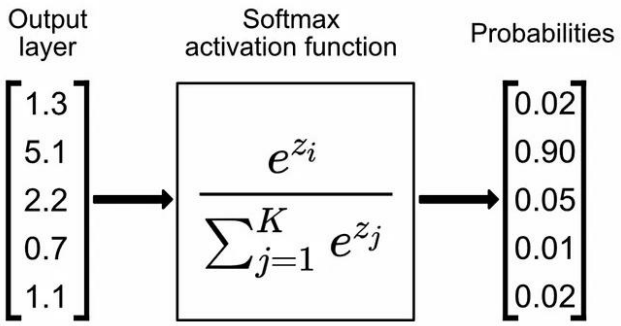
1. Softmax

   Fungsi softmax, yang biasanya diterapkan pada lapisan akhir model jaringan saraf untuk klasifikasi, mengubah skor output mentah (dikenal sebagai logits) menjadi probabilitas. Ini dilakukan dengan mengambil eksponensial dari setiap output dan menormalkan nilai-nilai tersebut dengan membaginya dengan jumlah dari semua eksponensial. Proses ini memastikan nilai output berada dalam rentang 0 hingga 1 dan jumlah totalnya sama dengan 1, sehingga nilai-nilai tersebut dapat diinterpretasikan sebagai probabilitas (Belagatti, 2024). Fungsi softmax dapat dinyatakan dengan persamaan berikut (Alzubaidi dkk., 2021):

$$f(x)_{softmax(i)} = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}} \qquad (3.1)$$

   Pada persamaan tersebut, ekspresi e^{a_i} merepresentasikan *output* saraf ke-i yang belum dinormalisasi pada lapisan sebelumnya. Ekspresi $\sum_{k=1}^{N} e^{a_k}$ adalah jumlah seluruh *output* saraf pada lapisan sebelumnya. Hasil dari persamaan berikut

adalah nilai dengan rentang 0–1. Berikut ilustrasi perhitungannya:



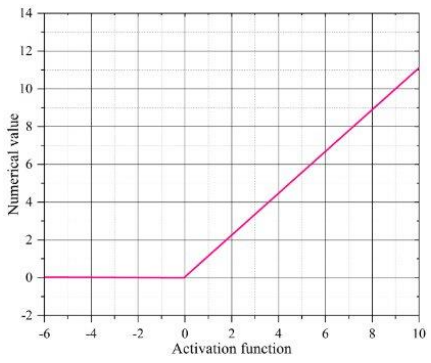
Gambar 3.5 Ilustrasi hasil perhitungan softmax (Belagatti, 2024)

2. ReLU

ReLU (Rectified Linear Unit) adalah fungsi aktivasi yang paling umum digunakan dalam model deep learning. Fungsi ini mengembalikan nilai 0 jika menerima input negatif, tetapi untuk setiap nilai positif (x), fungsi ini mengembalikan nilai (x) tersebut (Mkale, 2022). Keunggulan utama ReLU dibandingkan dengan fungsi aktivasi lainnya adalah beban komputasinya yang lebih ringan. Fungsi aktivasi ReLU dapat dituliskan sebagai berikut (Alzubaidi dkk., 2021):

$$f(x)_{ReLU} = \max(0, x) \tag{3.2}$$

Kurva fungsi ReLU dapat dilihat pada gambar berikut



Gambar 3.6 Kurva fungsi aktivasi ReLU (Jia, 2023)

3.7.2. Optimizer

Optimisasi (*optimizer*) adalah algoritma atau metode yang digunakan untuk meminimalkan fungsi kerugian (*loss function*) atau untuk memaksimalkan efisiensi produksi. *Optimizer* adalah fungsi matematika yang bergantung pada parameter yang dapat dipelajari model, yaitu bobot (*weights*) dan bias. *Optimizer* membantu menentukan bagaimana mengubah bobot dan laju pemelajaran jaringan saraf untuk mengurangi kerugian (Musstafa, 2021). Beberapa aspek yang diatur dan disesuaikan oleh *optimizer* adalah bobot dan laju pemelajaran. Bobot merupakan parameter yang menyimpan informasi tentang hubungan antar neuron. Laju pemelajaran menentukan seberapa besar perubahan yang diterapkan pada bobot di setiap iterasi.

Satu algoritma yang menonjol dalam hal melatih jaringan saraf tiruan adalah optimisasi Adam. Adam, singkatan dari *Adaptive Moment Estimation*, adalah algoritma dengan laju pemelajaran adaptif yang dirancang untuk mempercepat proses pelatihan jaringan saraf dalam-dalam dan mencapai konvergensi dengan cepat. Algoritma ini menyesuaikan laju pemelajaran setiap parameter berdasarkan riwayat gradiennya, sehingga membantu jaringan saraf belajar dengan efisien secara keseluruhan (Agarwal, 2023). Adamax adalah versi yang lebih maju dari algoritma Adam, yang menggunakan cara berbeda untuk mengatur langkah pembaruan selama pelatihan model.

3.7.3. *Loss Function*

Dalam *machine learning* (ML), fungsi kerugian (*loss function*) digunakan untuk mengukur kinerja model dengan menghitung deviasi prediksi model dari prediksi yang benar, atau "*ground truth*". Jika prediksi model akurat, kerugiannya kecil. Jika prediksi tidak akurat, kerugiannya besar (Bergmann dan Stryker, 2024). Jenis *loss function* yang biasanya digunakan dalam tugas klasifikasi adalah *Cross-*

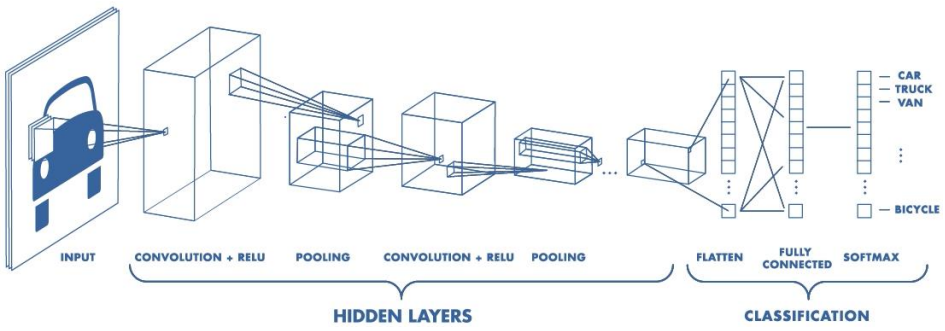
Entropy. Kerugian Cross-*Entropy* dapat dinyatakan dengan persamaan berikut:

$$Loss = H(p,q) = -\sum_{i=1}^{n} p(x_i)\, lnq\ (x_i) \qquad (3.3)$$

Pada persamaan tersebut, simbol p menunjukkan probabilitas asli. Sedangkan q adalah probabilitas perkiraan yang dihasilkan oleh model.

3.8. *Convolutional Neural Network* (CNN)

CNN, atau *Convolutional Neural Network*, adalah jenis model *deep learning* yang dirancang khusus untuk memproses data dengan pola grid, seperti gambar. Model ini bekerja dengan menggunakan lapisan konvolusi untuk mengekstraksi fitur-fitur penting dari data input. Dengan kemampuan ini, CNN sangat efektif dalam tugas-tugas seperti pengenalan objek, klasifikasi gambar, dan analisis visual lainnya, karena mampu menangkap pola spatial dan temporal dalam data dengan lebih baik dibandingkan model-model deep learning lainnya (Yamashita dkk., 2018).



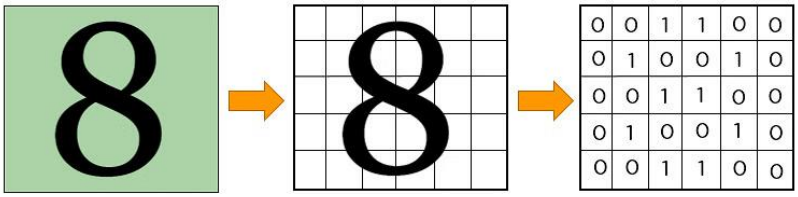
Gambar 3.7 Ilustrasi arsitektur CNN secara umum (Mishra, 2020)

CNN adalah sebuah sistem matematis yang terdiri dari tiga jenis lapisan utama: lapisan konvolusi, lapisan *pooling*, dan lapisan sepenuhnya terhubung (*fully*-connected). Lapisan konvolusi bertugas untuk menangkap fitur dari gambar dengan menggunakan filter yang bergerak di atas gambar, mendeteksi pola seperti tepi atau tekstur. Setelah itu, lapisan *pooling* mengurangi ukuran peta fitur sambil tetap menjaga informasi penting, biasanya dengan cara

mengambil nilai maksimum atau rata-rata dari bagian-bagian peta fitur. Terakhir, lapisan sepenuhnya terhubung menghubungkan fitur-fitur yang telah diproses ke neuron-neuron pada lapisan output, menghasilkan hasil akhir seperti klasifikasi atau prediksi. Dengan gabungan ketiga lapisan ini, CNN bisa memproses data gambar, mengekstrak fitur penting, dan memberikan hasil yang akurat (Yamashita dkk., 2018).

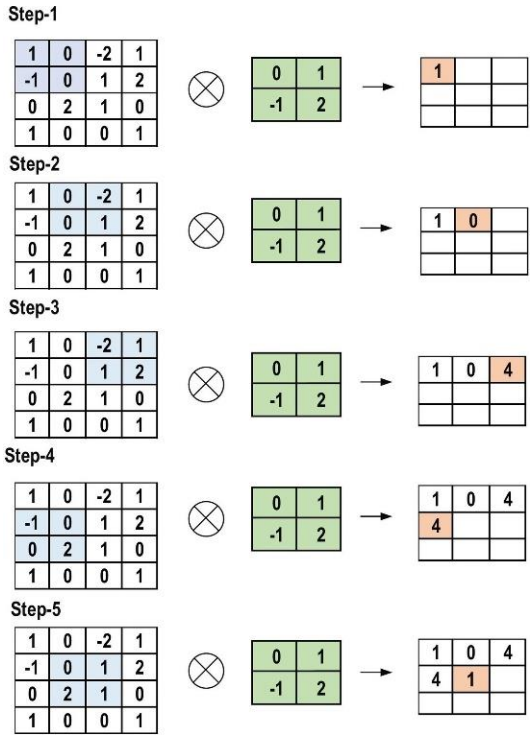
3.8.1. *Convolution Layer*

Lapisan konvolusi (*convolution layer*) adalah bertugas untuk menangkap fitur dari gambar dengan menggunakan filter (kernel) yang bergerak di atas gambar, mendeteksi pola seperti tepi atau tekstur (Lina, 2019). Hasil dari proses konvolusi ini disebut sebagai *activation map* atau *feature map*.



Gambar 3.8 Komputer melihat gambar sebagai suatu array representasi piksel (Biswal, 2023)

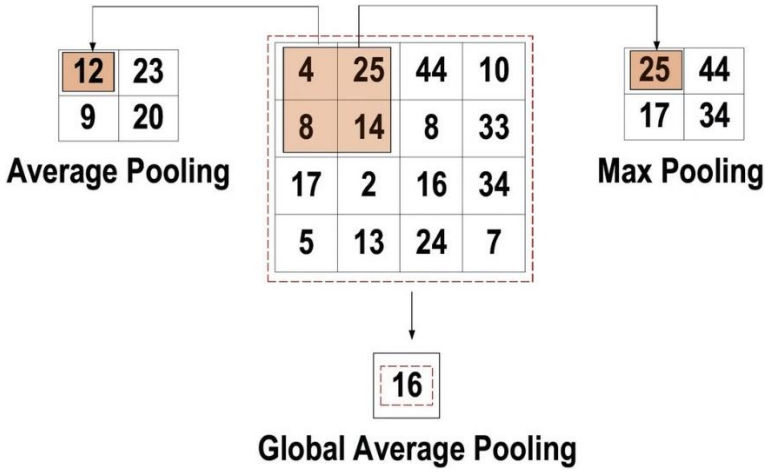
Ukuran kernel konvolusi dapat disesuaikan dengan kebutuhan spesifik dalam suatu tugas pemrosesan citra atau sinyal. Proses konvolusi dapat diilustrasikan dengan gambar di bawah ini.



Gambar 3.9 Proses konvolusi (Alzubaidi dkk., 2021)

3.8.2. *Pooling Layer*

Pooling layer bertugas untuk memperkecil ukuran *feature maps* yang dihasilkan dari proses konvolusi. Lapisan ini tetap mempertahankan sebagian besar informasi dominan pada setiap langkah *pooling*. Beberapa metode *pooling* yang paling sering digunakan diantaranya *max pooling*, *average pooling*, dan GAP (global *average pooling*) (Alzubaidi dkk., 2021). Ketiga operasi tersebut memiliki perbedaan dalam cara mengambil nilai dari *feature map*. *Max pooling* mengambil nilai tertinggi dari wilayah kecil *feature map*, *average pooling* mengambil nilai rata-rata dari wilayah kecil *feature map*, sedangkan GAP mengambil nilai rata-rata dari keseluruhan *feature map*.



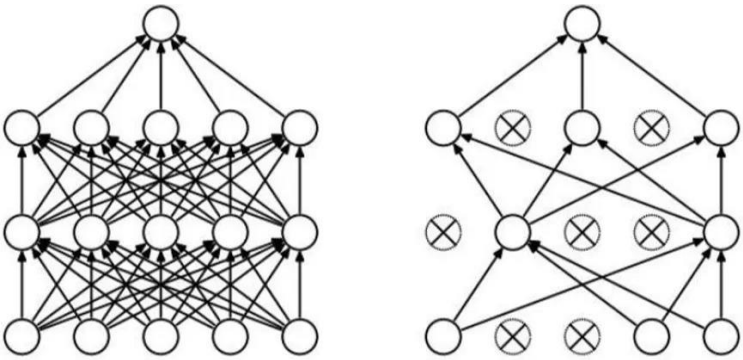
Gambar 3.10 Ilustrasi operasi *max pooling*, *average pooling*, dan *global average pooling* (Alzubaidi dkk., 2021)

3.8.3. *Batch Normalization*

Batch normalization merupakan suatu metode dalam *deep learning* yang membuat pelatihan model menjadi lebih cepat dan stabil (Huber, 2020). Fungsi utama dari *Batch Normalization* adalah untuk menormalkan input dari setiap lapisan dalam jaringan saraf sehingga output dari setiap lapisan tidak terlalu bervariasi dan lebih mudah diatur.

3.8.4. *Dropout Regularization*

Dropout adalah suatu teknik regularisasi jaringan saraf yang bekerja dengan cara memilih beberapa neuron secara acak untuk tidak digunakan dalam proses pelatihan. Teknik ini bertujuan untuk mencegah terjadinya *overfitting* dan mempercepat proses pelatihan (Lina, 2019). Neuron yang dihilangkan dapat berasal dari lapisan *hidden* maupun *visible*. Melatih model menggunakan *dropout* menghasilkan lebih sedikit kesalahan generalisasi akibat model yang terlalu menghafal data pelatihan (Srivastava, 2013).



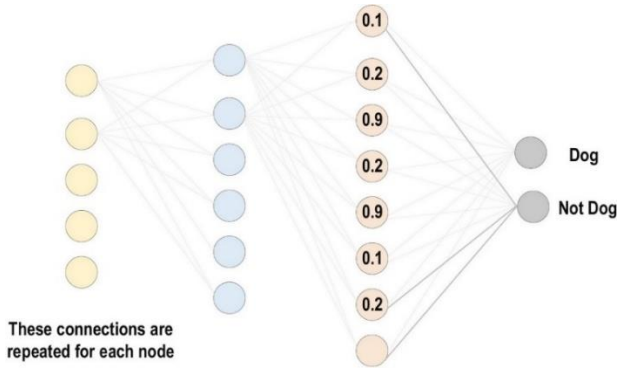
(a) Jaringan Syaraf Biasa

(b) Jaringan Syaraf Setelah dikenakan Teknik *Dropout*

Gambar 3.11 Ilustrasi *dropout regularization* (Lina, 2019)

3.8.5. *Fully-Connected* Layer

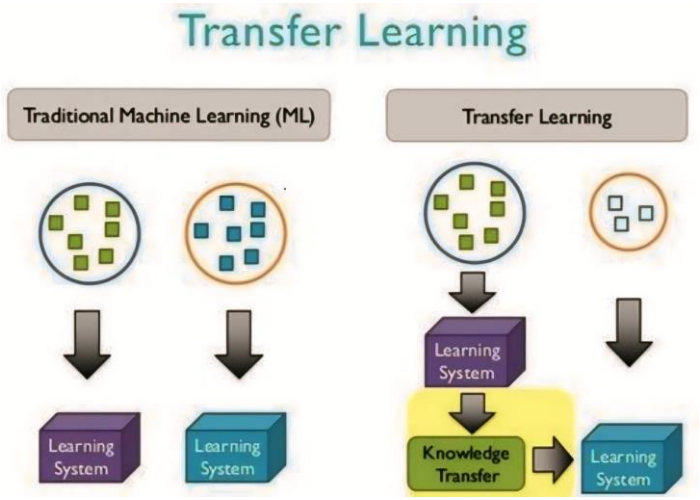
Lapisan sepenuhnya terhubung (*fully-connected layer*), adalah lapisan di dalam jaringan saraf di mana setiap neuron terhubung dengan semua neuron di lapisan sebelumnya (Alzubaidi dkk., 2021). Ini berarti bahwa setiap neuron di lapisan ini menerima informasi dari seluruh neuron di lapisan sebelumnya, memungkinkan integrasi lengkap dari semua fitur yang telah diekstraksi oleh lapisan sebelumnya. Lapisan ini biasanya terletak di akhir arsitektur model. Pada lapisan ini terjadi proses klasifikasi gambar sesuai kelas berdasarkan identifikasi fitur pada lapisan-lapisan sebelumnya.



Gambar 3.12 Ilustrasi fully-connected layer (Alzubaidi dkk., 2021)

3.9. *Transfer Learning*

Transfer learning (TL) adalah teknik yang melatih model saat ini dengan menggunakan model yang telah dilatih sebelumnya (*pre-trained model*) pada tugas-tugas terkait yang serupa (Hosna dkk., 2022). Dalam TL, *pre-trained model* berfungsi sebagai dasar untuk memulai pelatihan pada tugas baru, memungkinkan model untuk memanfaatkan pengetahuan yang sudah ada sehingga pelatihan menjadi lebih efisien dan cepat. Perbandingan antara metode *machine learning* tradisional dan *transfer learning* dapat dilihat pada gambar berikut

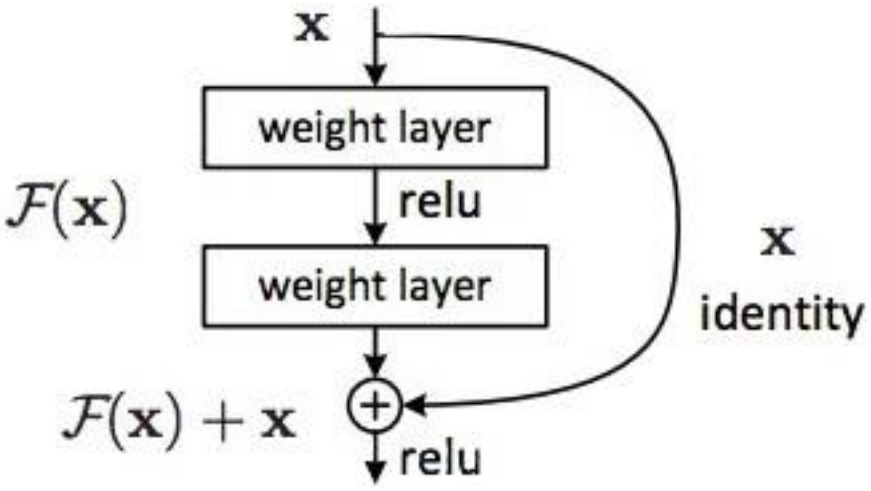


Gambar 3.13 Traditional Machine Learning vs Transfer Learning
(Wijaya dkk., 2021)

Dalam *transfer learning*, model yang telah dilatih sebelumnya (*pre-trained model*) dapat digunakan sebagai titik awal untuk melatih model pada tugas baru yang serupa. Proses ini melibatkan penggunaan seluruh atau sebagian dari model yang sudah ada, tergantung pada teknik pemodelan yang dipilih. Setelah diadopsi, model ini dapat disesuaikan atau disempurnakan lebih lanjut berdasarkan data *input-output* yang spesifik untuk tugas baru tersebut. Pendekatan ini memungkinkan pemanfaatan pengetahuan yang telah diperoleh dari tugas sebelumnya, sehingga mempercepat dan meningkatkan efisiensi pelatihan model pada tugas baru.

3.10. *Residual Network* (ResNet)

Residual Network (ResNet) adalah salah satu arsitektur *deep learning* yang sangat populer, diperkenalkan oleh Kaiming He dan rekan-rekannya dari Microsoft Research pada tahun 2015 (He dkk., 2015). ResNet muncul sebagai solusi untuk masalah degradasi performa pada jaringan neural yang sangat dalam, di mana penambahan lapisan justru mengurangi akurasi. Inti dari ResNet adalah konsep *residual learning*, yang memungkinkan jaringan untuk mempelajari perbedaan atau *residual* antara *input* dan *output* yang diharapkan, alih-alih mempelajari fungsi langsung.



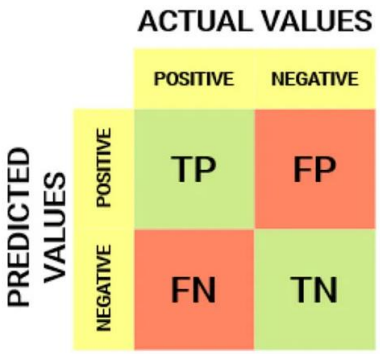
Gambar 3.14 Blok residual *learning* dengan *identity mapping* (He dkk., 2015)

Dalam *Residual Network* (ResNet), konsep utama yang perlu dipahami adalah *Skip Connection* dengan pemetaan identitas (*identity mapping*). Pemetaan identitas ini adalah fitur penting yang tidak memiliki parameter dan berfungsi untuk menambahkan *output* dari lapisan sebelumnya langsung ke lapisan berikutnya (Shorten, 2019). Dengan menggunakan *skip connection*, jaringan dapat meneruskan informasi dari lapisan sebelumnya tanpa perubahan, memungkinkan jaringan untuk mempelajari fungsi identitas. Ini membantu dalam pelatihan jaringan yang lebih dalam dengan meningkatkan kemampuan untuk memanfaatkan informasi yang sudah ada dari lapisan sebelumnya (Banjara, 2023).

ResNet telah mengalami banyak pengembangan dan pembaruan, dimulai dengan ResNet-18 yang memiliki 18 lapisan, varian ini terus berkembang hingga ResNet-152 dengan 152 lapisan. Pengembangan juga menghasilkan varian lain seperti ResNeXt, *Wide* ResNet, dan ResNet *with Attention Mechanism* (Chaure, 2024). Setiap varian menawarkan peningkatan kapasitas untuk menangani data yang lebih kompleks sambil menjaga efisiensi pelatihan, menjadikan ResNet sebagai fondasi penting dalam berbagai aplikasi *deep learning*.

3.11. Metrik Evaluasi

Metrik evaluasi (*Evaluation Metrics*) adalah ukuran atau indikator yang digunakan untuk menilai kinerja dan efektivitas model pemelajaran mesin atau algoritma. Metrik ini memberikan gambaran seberapa baik model tersebut bekerja dalam memprediksi atau mengklasifikasikan data, dan digunakan untuk membandingkan hasil antar model atau memilih model terbaik. Dalam tugas klasifikasi data, metrik evaluasi digunakan dalam dua tahap, yaitu tahap pelatihan dan pengujian (Hossin dan Sulaiman, 2015). Contoh metrik evaluasi meliputi akurasi (*accuracy*), *presisi*, *recall*, dan *f1-score*. Metrik evaluasi tersebut bisa dihitung menggunakan nilai-nilai dari *confusion matrix*.



Gambar 3.15 *Confusion Matrix* (Karra, 2020)

True positive (TP) terjadi ketika data yang sebenarnya bernilai positif diprediksi dengan benar sebagai positif. *True negative* (TN) adalah kondisi di

mana model memprediksi data yang bernilai negatif dengan tepat sebagai negatif. *False positive* (FP) terjadi ketika data yang sebenarnya negatif diprediksi sebagai positif. *False negative* (FN) adalah situasi di mana data yang seharusnya positif diprediksi sebagai negatif. Dengan menggunakan kombinasi dari empat nilai tersebut, dapat dihitung metrik evaluasi di bawah ini.

3.11.1. *Akurasi*

Secara umum, akurasi (*accuracy*) merupakan perbandingan antara prediksi benar terhadap jumlah total contoh yang dievaluasi. Nilai akurasi dapat dihitung dengan menggunakan persamaan berikut (Hossin & Sulaiman, 2015):

$$accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (3.4)$$

3.11.2. *Precision*

Precision digunakan untuk mengukur pola positif yang diprediksi dengan benar dari total pola yang diprediksi dalam kelas positif. Secara matematis, *precision* dapat ditulis dalam bentuk persamaan berikut (Hossin & Sulaiman, 2015):

$$precision = \frac{TP}{TP+FP} \quad (3.5)$$

3.11.3. *Recall*

Recall merupakan perbandingan antara jumlah prediksi positif yang benar dengan jumlah keseluruhan yang diprediksi benar. *Recall* dapat diungkapkan secara matematis melalui persamaan berikut (Hossin & Sulaiman, 2015):

$$recall = \frac{TP}{TP+TN} \quad (3.6)$$

3.11.4. *F1-score*

F1-score digunakan untuk menghitung rata-rata harmonik antara nilai *precision* dan *recall*. Dalam bentuk matematis, *f1-score* dapat dituliskan dengan persamaan berikut (Hossin & Sulaiman, 2015):

$$f1 - score = \frac{2 \times recall \times precision}{recall + precision} \quad (3.7)$$

3.12. Python

Python adalah bahasa pemrograman tingkat tinggi yang mulai dikembangkan pada tahun 1989 oleh *programmer* asal belanda, Guido van Rossum (Moltzau, 2019). Bahasa pemrograman tingkat tinggi adalah bahasa yang dirancang untuk lebih mudah dipahami dan digunakan oleh manusia, dengan sintaksis dan struktur yang lebih mirip dengan bahasa alami atau logika matematis. Bahasa ini menyediakan tingkat abstraksi yang tinggi dari perangkat keras, memungkinkan programmer untuk fokus pada logika dan fungsionalitas tanpa harus mengelola detail teknis rendah seperti manajemen memori secara langsung.

Python adalah bahasa pemrograman *open-source* yang populer dan serbaguna, digunakan dalam berbagai aplikasi mulai dari pengembangan web, analisis data, kecerdasan buatan, dan otomatisasi skrip . Python juga didukung oleh ribuan pustaka (*library*) yang kuat, yang memungkinkan pengembang untuk dengan mudah mengimplementasikan solusi di berbagai bidang, mulai dari ilmu data dengan pandas dan NumPy, hingga pengembangan web dengan Django dan Flask, serta machine learning dengan TensorFlow dan Keras.

TensorFlow adalah sebuah pustaka *open-source* yang digunakan dalam pemelajaran mesin dan kecerdasan buatan yang dikembangkan oleh Google. TensorFlow dapat digunakan untuk banyak tugas, termasuk pelatihan dan *deployment* model pemelajaran mesin (Kılınç, 2023). Salah satu keunggulan TensorFlow adalah kemampuannya berjalan pada berbagai *platform* seperti *desktop*, perangkat *mobile*, dan lingkungan komputasi *cloud*. TensorFlow juga

didukung oleh komunitas besar pengguna dan pengembang, yang mempermudah pencarian bantuan dan akses ke berbagai sumber daya secara daring.

Keras adalah API untuk mengembangkan jaringan saraf tiruan. Aplikasi dari Keras sangat luas di mana kita dapat membangun jaringan saraf tiruan untuk klasifikasi gambar, pemrosesan bahasa alami, pengenalan suara, dan prediksi *time series* (Wahidi, 2021). TensorFlow dan Keras bekerja sama untuk memudahkan pengembangan model *machine learning*. Keras, sebagai antarmuka tingkat tinggi, menyediakan API yang sederhana dan intuitif untuk membangun dan melatih model neural network, sementara TensorFlow berfungsi sebagai backend yang kuat untuk komputasi dan eksekusi model. Kolaborasi ini memungkinkan pengguna untuk memanfaatkan kemudahan penggunaan Keras dengan kekuatan dan fleksibilitas TensorFlow, menggabungkan kemudahan pengembangan dengan performa tinggi dalam aplikasi machine learning.

BAB IV

METODE PENELITIAN

Metode penelitian ini mencakup alat dan bahan, prosedur pelaksanaan, serta rencana jadwal penelitian. Alat dan bahan dijelaskan untuk memberikan informasi tentang kebutuhan penelitian. Prosedur pelaksanaan dijabarkan secara rinci agar setiap tahapan dapat diikuti dengan jelas. Rencana jadwal penelitian disusun untuk menggambarkan waktu pelaksanaan setiap tahapan penelitian.

4.1. Alat dan Bahan Penelitian

4.1.1. Alat

Pada penelitian ini digunakan peralatan sebagai berikut:

1. Perangkat Komputer

   Model : Lenovo IdeaPad Flex 5 14IIL05

   CPU : Intel® Core™ i3-1005G1

   GPU : Intel UHD Graphics

   RAM : 8 GB

   Penyimpanan : 512 GB SSD

   Sistem Operasi : Microsoft Windows 11

2. *Platform* dan Perangkat Lunak

   Bahasa Pemrograman : Python 3.10.12

   IDE : Google Colab

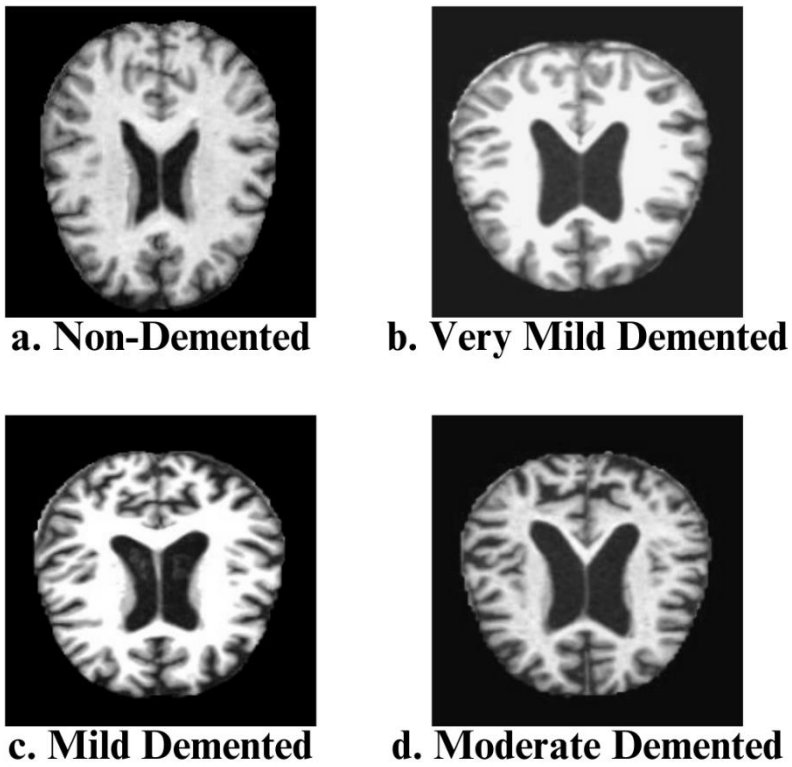
   Library Machine Learning : Tensorflow dan Keras

   Library Augmentasi Data : ImageDataGenerator

   Library Evaluasi Model : Matplotlib

4.1.2. Bahan

Penelitian ini menggunakan dataset yang berisi citra MRI otak yang dikelompokkan ke dalam empat kategori kondisi pasien, yaitu *non demented*, *very mild demented*, *mild* demented, dan *moderate demented*. Dataset yang digunakan dalam penelitian ini adalah dataset open source yang dipublikasikan oleh Uraninjo pada tahun 2022 di situs Kaggle.com, yang merupakan hasil augmentasi dari data yang diunggah oleh Dubey pada tahun 2019. Data tersebut berasal dari publikasi oleh Open Access Series of Imaging Studies (OASIS). Contoh data gambar MRI pada masing-masing kategori dapat dilihat pada gambar 4.1.



Gambar 4.1 Dataset MRI otak

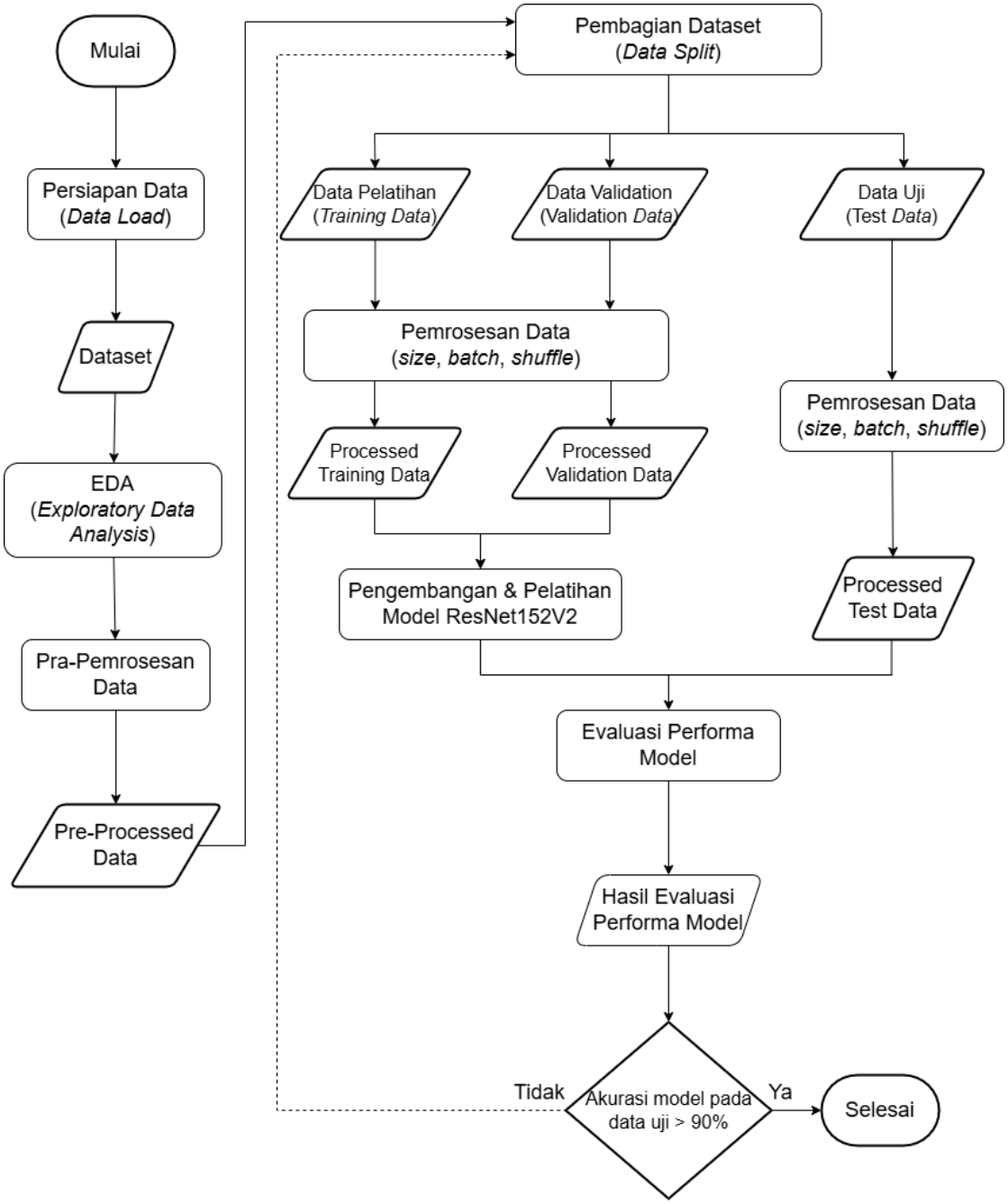
Dataset berisi total 40.384 data gambar. Distribusi data dari tiap kategori dapat dilihat pada tabel 4.1.

Tabel 4.1 Distribusi data

No	Kategori	Jumlah Data
1	Non-Demented	12.800
2	Very Mild Demented	11.200
3	Mild Demented	9.856
4	Moderate Demented	6.528
Total		40.384

4.2. Prosedur Penelitian

Penelitian ini secara garis besar dibagi dalam enam tahapan. Tahap pertama melibatkan persiapan data, yang dilakukan dengan mencari dan mengumpulkan data sekunder terlebih dahulu, kemudian memuatnya ke dalam proyek (*data load*) untuk proses lebih lanjut. Tahap kedua adalah melakukan Exploratory Data Analysis (EDA) untuk menganalisis dataset dengan menampilkan distribusi data pada setiap kelas serta mengidentifikasi ekstensi file gambar yang terdapat dalam dataset. Tahap ketiga melibatkan pra-pemrosesan data dengan mengatur ukuran gambar, *batch*, melakukan pengacakan (*shuffle*), dan memastikan keseragaman ekstensi *file* agar gambar siap digunakan dalam model. Tahap keempat adalah membagi data (*data split*) yang sudah diproses menjadi data pelatihan, data validasi, dan data uji dengan rasio tertentu. Tahap kelima adalah pengembangan dan pelatihan model yang dilakukan dengan mengembangkan model dengan metode *transfer learning* menggunakan arsitektur ResNet152V2 dan melatihnya dengan data pelatihan. Tahap terakhir adalah mengevaluasi model dengan metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *f1-score*. Melalui metrik evaluasi ini dapat ditentukan kemampuan model dalam mengklasifikasi gambar empat tahapan penyakit Alzheimer. Diagam alir tahapan penelitian dapat dilihat pada gambar 4.2.



Gambar 4.2 Diagram alir penelitian

4.2.1. Persiapan Data

Dataset yang digunakan adalah dataset sekunder dari Kaggle.com yang dipublikasikan oleh pengguna dengan nama Uraninjo (2022). Dataset memiliki total 40.384 data gambar yang terdiri atas empat kategori/tahapan penyakit Alzheimer yaitu *Non-Demented*, *Very Mild*

Demented, *Mild Demented*, dan *Moderate Demented*. Pada tahap ini, dataset diunduh dan disimpan di Google Drive sebagai basis data untuk pembuatan model. Untuk menggunakannya, dataset dimuat ke dalam lingkungan proyek di Google Colab.

4.2.2. *Exploratory Data Analysis* (EDA)

Data yang telah dimuat ke dalam proyek lalu dianalisis untuk mengetahui struktur dan polanya. Pada dataset ini dilakukan analisis untuk distribusi data dan keragaman ekstensi file. Distribusi data dapat dilihat dengan melakukan visualisasi terhadap jumlah data pada masing-masing kategori. Selain itu, dianalisa juga keragaman ekstensi file pada dataset dengan menampilkan seluruh ekstensi file gambar.

4.2.3. Pra-pemrosesan Data

Pada tahap ini, ketidaksesuaian dalam dataset akan diproses. Beberapa langkah yang akan dilakukan meliputi penyeragaman ekstensi *file* dan pemrosesan data, seperti pengubahan ukuran gambar, *batch*, dan pengacakan (shuffle). Penyeragaman ekstensi file dilakukan dengan menghapus gambar yang memiliki ekstensi selain jpg, jpeg, dan png. Pemrosesan data, termasuk pengubahan ukuran gambar, *batch*, dan pengacakan, akan dilakukan menggunakan ImageDataGenerator. Pengubahan ukuran gambar (*image resizing*) bertujuan untuk menyamakan ukuran gambar sesuai *input* model; *batch* adalah jumlah sampel yang diproses pada setiap iterasi; dan pengacakan (*shuffle*) menentukan apakah data akan diacak di setiap iterasi atau tidak. Meskipun pemrosesan ini dilakukan setelah tahap pembagian data, teknik ini tetap dianggap sebagai bagian dari pra-pemrosesan data.

4.2.4. Pembagian Data

Dataset telah dibagi menjadi data pelatihan dan data uji dengan rasio 5:1, memastikan bahwa sebagian besar data digunakan untuk

pelatihan model, sementara sisanya dialokasikan untuk pengujian. Selanjutnya, data pelatihan yang telah dipisahkan kemudian dibagi lagi menjadi dua bagian: data pelatihan dan data validasi, dengan rasio 3:1. Pembagian ini bertujuan untuk memberikan set data yang cukup untuk melatih model secara efektif, sekaligus memungkinkan evaluasi performa model selama proses pelatihan melalui data validasi. Data pelatihan dan data validasi digunakan pada tahap pelatihan model. Sedangkan data uji digunakan untuk mengevaluasi model seetelah pelatihan dengan melakukan pengujian terhadap data yang belum pernah dilihat sebelumnya.

4.2.5. Pengembangan Model

Pada penelitian ini, digunakan metode pembelajaran mendalam (deep learning) yang berbasis CNN. Metode yang diterapkan adalah transfer learning dengan arsitektur model ResNet152V2 yang sebagian besar berisikan lapisan konvolusi dan mengandalkan konsep *skip connection* dan pemetaan identitas. Model ini kemudian dilengkapi dengan lapisan tambahan seperti *batch normalization*, *dropout*, dan *fully-connected* untuk menyesuaikan dataset.

Input yang digunakan untuk model adalah gambar yang memiliki tiga dimensi yaitu tinggi, lebar, dan kedalaman. *Input* akan masuk ke dalam serangkaian lapisan konvolusi yang diikuti oleh *max pooling*, dengan setiap rangkaian memiliki parameter yang berbeda. Setiap gambar yang melewati lapisan konvolusi dan *max pooling* akan mengalami reduksi dimensi dan resolusi gambar (*down sampling*).

Setelah melalui *base model*, gambar akan melewati beberapa lapisan tambahan yaitu *batch normalization*, *dropout*, dan *fully-connected* atau lapisan *dense*. *Batch normalization* berfungsi untuk mempercepat proses konvergensi dan meningkatkan stabilitas selama pelatihan. Sementara itu, *dropout* digunakan untuk mengurangi

overfitting dengan menghapus sebagian unit (neuron) secara acak dalam lapisan selama fase pelatihan. Sementara itu, *fully-connected layer* berfungsi untuk menghubungkan setiap neuron dari lapisan sebelumnya dengan setiap neuron di lapisan selanjutnya. Hal ini memungkinkan model untuk mempelajari representasi fitur yang kompleks dari data. Di lapisan ini juga terjadi proses klasifikasi gambar sesuai dengan kelas yang ada, berdasarkan ekstraksi fitur yang dilakukan sebelumnya.

Penelitian ini menggunakan sejumlah parameter penting yang dipilih secara khusus untuk memaksimalkan kinerja model. Parameter tersebut meliputi *optimizer* dan nilai *learning rate*. *Optimizer* yang digunakan adalah Adamax, yang merupakan versi terbaru dari Adam. Nilai learning rate yang diterapkan adalah 0,001. Penggunaan *learning rate* berpengaruh terhadap akurasi model dan kecepatan pelatihan. Secara umum, nilai *learning rate* berkisar antara 0,1 hingga 0,0001. Semakin kecil *learning rate*, semakin cepat proses pelatihan, namun berisiko model tidak mencapai *global optimum*, sehingga akurasinya tidak optimal. Sebaliknya, *learning rate* yang terlalu besar akan memperlambat pelatihan, tetapi memastikan model mencapai akurasi terbaik.

4.2.6. Metrik Evaluasi

Metrik evaluasi digunakan untuk menilai kinerja model yang telah dilatih sebelumnya. Proses evaluasi dilakukan dengan memberikan tugas klasifikasi pada data uji yang belum pernah dilihat oleh model. Metrik evaluasi yang digunakan meliputi akurasi (*accuracy*), *precision*, *recall*, dan *f1-score*, yang dihitung berdasarkan *confusion matrix* dari *library* Matplotlib. Semakin banyak data uji yang diprediksi dengan benar, semakin baik performa model tersebut.

4.3. Jadwal Penelitian

Penelitian dilakukan selama lima bulan dari bulan Juli 2024 sampai dengan bulan November 2024 dengan tahapan penelitian seperti dalam tabel berikut.

Tabel 4.2 Jadwal penelitian

Kegiatan Penelitan	Bulan				
	1	2	3	4	5
Studi Literatur	■	■			
Mencari dataset untuk membuat model		■			
Pengembangan dan pelatihan model		■	■		
Evaluasi model			■	■	
Menyusun draft skripsi			■	■	
Ujian skripsi					■

BAB V

HASIL PENELITIAN DAN PEMBAHASAN

Bab ini akan memaparkan implementasi dari prosedur penelitian yang telah disampaikan pada bab sebelumnya, serta memberikan pembahasan terhadap hasil yang diperoleh.

5.1. Persiapan Dataset

Dalam penelitian ini, digunakan dataset berupa data sekunder yang diambil dari sumber *open source Kaggle.com*. Dataset ini dikumpulkan, dilakukan augmentasi, dan dipublikasikan oleh pengguna bernama Uraninjo pada tahun (2022). Terdapat total 40.384 gambar MRI otak pasien, yang terbagi ke dalam empat kategori atau tahapan Alzheimer: *Non-Demented*, *Very Mild Demented*, *Mild Demented*, dan *Moderate Demented*. Dataset tersebut diunduh ke penyimpanan lokal, kemudian diunggah ke Google Drive sebagai basis data. Setelah itu, dataset dimuat ke dalam Google Colab dengan membuat folder khusus. Proses ini dapat dilihat pada gambar 5.1.

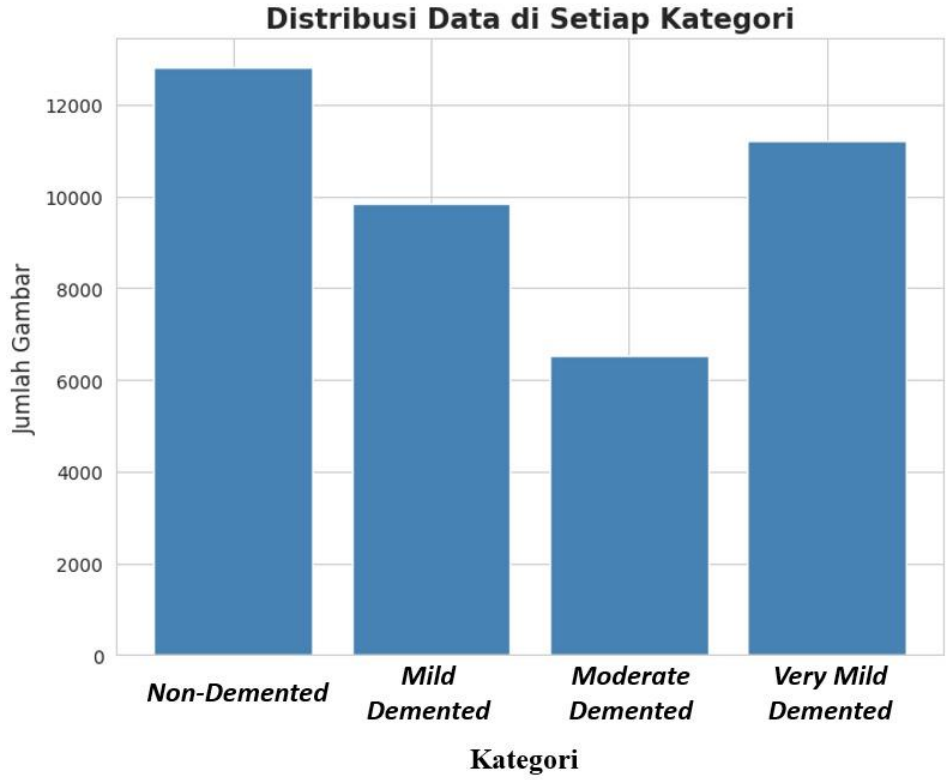
```
# Mengunduh Dataset dari Google Drive
!gdown --id 1AQLWiTNzwrIQQRjmpiylho9uAHRBmqdG -O MRI.zip

/usr/local/lib/python3.10/dist-packages/gdown/__main__.py
  warnings.warn(
Downloading...
From (original): https://drive.google.com/uc?id=1AQLWiTNz
From (redirected): https://drive.google.com/uc?id=1AQLWiT
To: /content/MRI.zip
100% 397M/397M [00:13<00:00, 29.0MB/s]
```

Gambar 5.1 Pengunduhan dataset

5.2. *Exploratory Data Analysis* (EDA)

Dataset yang telah dimuat ke dalam Google Colab kemudian dieksplorasi untuk memperoleh informasi penting terkait isi dataset. Proses eksplorasi ini meliputi penayangan gambar dari setiap kategori, penghitungan jumlah data per kategori, visualisasi distribusi data, serta identifikasi berbagai ekstensi *file* gambar yang

tersedia. Langkah eksplorasi ini penting untuk merencanakan tahap-tahap selanjutnya agar data siap digunakan dalam proses pelatihan.



Gambar 5.2 Grafik distribusi dataset

```
Jumlah gambar dari kategori NonDemented = 12800
Jumlah gambar dari kategori MildDemented = 9856
Jumlah gambar dari kategori ModerateDemented = 6528
Jumlah gambar dari kategori VeryMildDemented = 11200

Total jumlah gambar = 40384
```

Gambar 5.3 Jumlah gambar pada setiap kategori dan totalnya

Berdasarkan gambar 5.2 dan 5.3 terlihat bahwa jumlah data pada setiap kategori tidak sama. Meskipun distribusi dataset yang digunakan tidak seimbang, dengan jumlah data yang bervariasi di setiap kategori, penyamarataan distribusi data tidak direncanakan. Keputusan ini didasarkan pada pertimbangan bahwa model yang dikembangkan diharapkan mampu menangani ketidakseimbangan data secara efektif. Ketidakseimbangan ini juga dapat memberikan tantangan tambahan bagi model selama proses pembelajaran, yang berpotensi meningkatkan kemampuan generalisasi tanpa perlu melakukan penyamarataan distribusi data.

```
Folder: VeryMildDemented
Format: jpg, Jumlah: 11200


Folder: MildDemented
Format: jpg, Jumlah: 9856


Folder: NonDemented
Format: jpg, Jumlah: 12800


Folder: ModerateDemented
Format: jpg, Jumlah: 6528
```

Gambar 5.4 Ekstensi *file* dan jumlah datanya

Hasil eksplorasi yang ditampilkan pada gambar 5.4 menunjukkan bahwa semua file gambar memiliki ekstensi yang sama, yaitu *.jpg* atau *.jpeg* (Joint Photographic Experts Group). Format ini merupakan format gambar umum yang sering digunakan untuk menyimpan foto dengan kualitas baik dan ukuran file yang lebih kecil. Karena semua ekstensi file sudah seragam, proses penyeragaman ekstensi tidak perlu dilakukan.

5.3. Implementasi Pra-pemrosesan Data

Berdasarkan informasi yang diperoleh dari tahap *Exploratory Data Analysis* (EDA), langkah berikutnya adalah melakukan penyesuaian dataset untuk memastikan dataset siap digunakan dalam proses pelatihan dan pengujian. Penyesuaian ini mencakup pengubahan ukuran gambar, penentuan *batch size*, serta pengacakan dataset.

5.3.1. Implementasi Pemrosesan Data

Pemrosesan data dilakukan bertujuan untuk menyesuaikan dataset dengan kebutuhan model dalam proses pelatihan dan pengujian. Penyesuaian ini meliputi pengubahan ukuran gambar, penetapan *batch size*, serta pengacakan dataset (*shuffle*). Pengubahan ukuran gambar (*image resizing*) dilakukan untuk menyamakan dimensi gambar agar sesuai dengan *input* model. Penetapan *batch size* berfungsi menentukan jumlah gambar yang

akan diproses dalam setiap iterasi, sementara pengacakan (*shuffle*) bertujuan untuk mengacak urutan gambar dalam dataset sebelum model dilatih.

Pengubahan ukuran gambar mengikuti ukuran *input* gambar pada model yang digunakan. Pada penelitian ini digunakan arsitektur model ResNet152V2 yang memiliki ukuran *input* gambar sebesar 224×224 piksel. Penerapan *image resizing* akan mengubah ukuran seluruh gambar pada dataset menjadi ukuran yang dibutuhkan, dilakukan dengan parameter *target_size*.

Penentuan *batch size* dilakukan dengan parameter *batch_size* yang diatur menggunakan nilai 64. Nilai ini berarti model akan mengambil 64 gambar dalam satu iterasi. Nilai *batch size* akan memengaruhi durasi pelatihan dan kemampuan model dalam melakukan tugasnya.

Pengacakan gambar (*shuffle*) dilakukan dengan mengatur parameter *shuffle* sebagai '*True*' untuk melakukan pengacakan dan '*False*' jika tanpa pengacakan. Pengacakan adalah proses mengacak urutan data sebelum pelatihan model, dengan tujuan agar model tidak menghafal urutan data dan mampu belajar secara lebih umum untuk menghasilkan prediksi yang lebih akurat.

5.4. Implementasi Pembagian Data

Pembagian dataset dilakukan dengan membagi dataset menjadi tiga bagian, yaitu data pelatihan, data validasi, dan data uji. Data pelatihan dan data validasi akan digunakan dalam proses pelatihan model, sedangkan data uji digunakan untuk mengevaluasi kinerja model. Pembagian dataset dilakukan dengan acak dengan rasio tertentu antara data pelatihan, validasi, dan pengujian. Dataset dibagi menjadi data pelatihan dan data uji dengan rasio 5:1, kemudian data pelatihan dibagi lagi menjadi dua bagian yaitu data pelatihan dan data validasi dengan rasio 3:1. Rasio pembagian ini memastikan model mendapatkan cukup data untuk melakukan pelatihan dan evaluasi (validasi), serta memiliki data terpisah yang belum dilihat sebelumnya untuk tahap pengujian. Hasil pembagian data dapat dilihat pada tabel 5.1 berikut

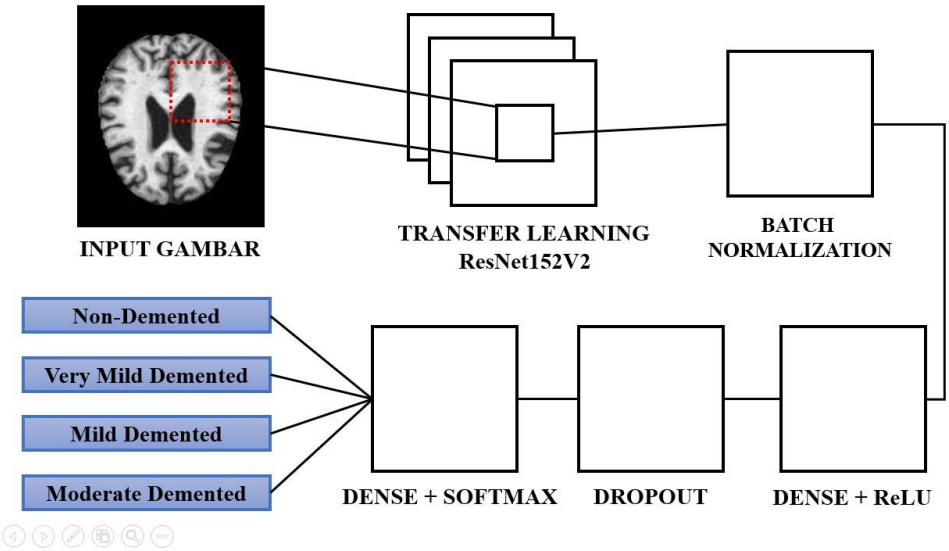
Tabel 5.1 Pembagian dataset

No	Kategori	Data Pelatihan	Data Validasi	Data Uji	Jumlah
1	Non-Demented	7259	2341	3200	12800
2	Very Mild Demented	6695	2265	2240	11200
3	Mild Demented	6669	2291	896	9856
4	Moderate Demented	4865	1599	64	6528
Total		25488	8496	6400	40384

5.5. Hasil Pemodelan

5.5.1. Pengembangan Model

Model dilatih dengan menggunakan metode *transfer learning* ResNet152V2 dengan tambahan lapisan *batch normalization*, *dense* dengan fungsi aktivasi ReLU, *dropout* dengan nilai sebesar 0,3, dan *dense* dengan fungsi aktivasi *softmax* pada lapisan akhir. Sebagai arsitektur dasar model, ResNet152V2 dikonfigurasi dengan beberapa parameter, yaitu include_top = ‘False’; weights = ‘imagenet’; input_shape = ‘IMG_SHAPE’; dan pooling = ‘max’. Parameter *include_top* yang dikonfigurasi *False* ini berarti lapisan *fully-connected* (*dense*) terakhir pada arsitektur model tidak disertakan dan hanya mneggunakan dasar *convolutional* modelnya saja. Parameter *weights* yang dikonfigurasi imagenet berarti bobot yang digunakan pada model adalah bobot hasil pelatihan dari dataset ImageNet. Parameter *input_shape* dikonfigurasi berdasarkan dimensi tinggi, lebar, dan jumlah kanal dari *input* gambar model yaitu (224, 224, 3). Sedangkan parameter *pooling* yang dikonfigurasi *max* berarti model akan menggunakan *max pooling* setelah lapisan konvolusi dimana hanya nilai maksimal dari setiap *feature map* yang akan dipertahankan.



Gambar 5.5 Arsitektur model

Penggunaan lapisan *batch normalization* berperan penting dalam menstabilkan serta mempercepat proses pelatihan model dengan menormalkan data dari setiap *batch* menggunakan perhitungan rata-rata dan variansi. Proses ini membantu menyelaraskan data di setiap lapisan, mengurangi risiko masalah seperti perubahan nilai yang terlalu besar atau kecil, sehingga model dapat belajar lebih cepat dan efisien. Lapisan berikutnya adalah lapisan *dense* dengan fungsi aktivasi ReLU, yang berfungsi untuk mengubah nilai input menjadi positif. Aktivasi ini membantu model dalam mempelajari hubungan kompleks pada data, serta meningkatkan performa dan kecepatan pelatihan. Selain itu, lapisan ini juga menggunakan regularisasi untuk mencegah terjadinya *overfitting*. Selanjutnya, digunakan lapisan *dropout* yang berfungsi untuk secara acak menonaktifkan sejumlah neuron pada setiap iterasi pelatihan. Nilai 0,3 yang ditetapkan pada lapisan ini menunjukkan bahwa 30% unit pada lapisan tersebut akan diabaikan, sedangkan sisanya digunakan selama pelatihan. Pada lapisan terakhir, digunakan lapisan *dense* dengan aktivasi *softmax*, yang bertugas mengonversi nilai probabilitas setiap kelas ke dalam rentang $0 - 1$. Probabilitas tertinggi di antara kelas-kelas tersebut dianggap sebagai hasil klasifikasi.

Setelah model selesai dibangun, langkah berikutnya adalah mendefinisikan optimizer, mengompilasi model, dan mendefinisikan

callbacks. Optimizer yang digunakan adalah Adamax, salah satu varian dari Adam yang lebih stabil dan menunjukkan performa baik, dengan *learning rate* sebesar 0,001. Selanjutnya, dilakukan kompilasi model, yang merupakan tahap penting sebelum pelatihan dimulai. Parameter yang digunakan dalam kompilasi model meliputi *optimizer*, *loss*, dan *metrics*. Parameter *loss* berfungsi untuk mengukur seberapa baik kinerja model selama pelatihan dengan menghitung kerugian menggunakan fungsi *categorical crossentropy*. Sementara itu, *metrics* digunakan untuk mengevaluasi performa model selama pelatihan dan pengujian dengan menghitung jumlah prediksi yang benar dari total prediksi yang dihasilkan oleh model. Langkah terakhir sebelum pelatihan adalah mendefinisikan *callbacks*, yang akan menghentikan pelatihan secara otomatis ketika kondisi tertentu tercapai. Dalam hal ini, digunakan *callbacks* untuk menghentikan pelatihan jika tidak ada peningkatan akurasi pada dataset validasi selama 5 *epoch* atau iterasi berturut-turut.

5.5.2. Pelatihan Model

Pada tahap pelatihan model, digunakna beberapa parameter seperti dataset pelatihan, *epochs*, dataset validasi, *batch_size*, dan *callbacks*. Dataset pelatihan merupakan dataset yang digunakan model untuk proses pelatihan. Parameter *epochs* adalah berapa kali seluruh dataset dilalui selama pelatihan, dalam pelatihan ini ditetapkan maksimal sebanyak 50 *epochs*. Dataset validasi merupakan dataset hasil pemrosesan sebelumnya yang akan digunakan untuk validasi model selama pelatihan. Parameter *batch_size* yang digunakan sebesar 48. Sedangkan *callbacks* yang digunakan sesuai dengan yang telah didefinisikan pada tahap pengembangan model.

Tabel 5.2 Data akurasi dan *loss* pada tiap *epoch* pelatihan

Epoch	*Accuracy*	*Validation Accuracy*	*Loss*	*Validation Loss*
1	0,4743	0,4512	2,4074	2,8739
2	0,6537	0,6534	1,9573	2,1827
3	0,7619	0,7931	1,6825	1,6212
4	0,8232	0,7027	1,4553	2,1931

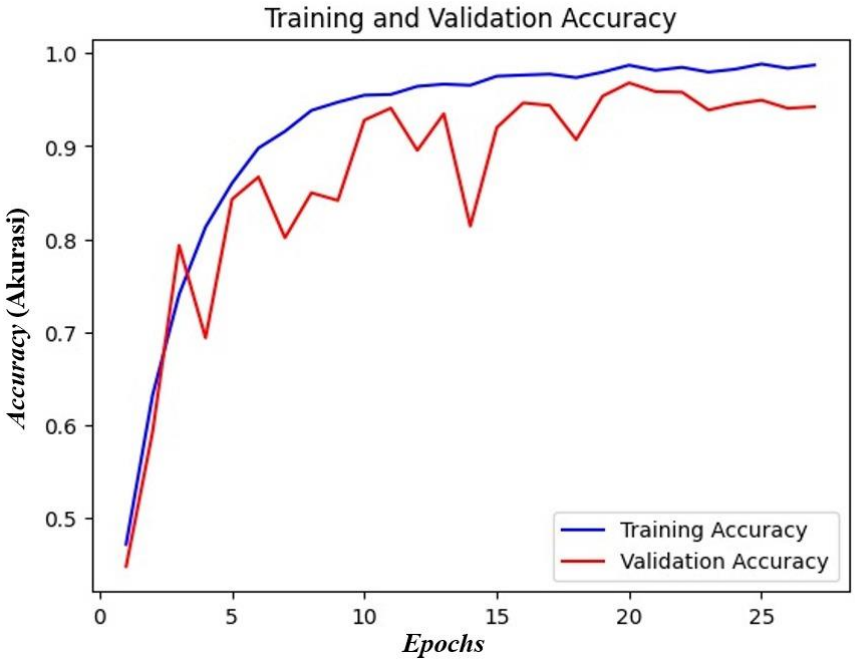
5	0,8625	0,8416	1,2743	1,2879
6	0,9038	0,8641	1,1324	1,1983
7	0,9217	0,8021	0,9945	1,3345
8	0,9342	0,8485	0,8531	1,1594
9	0,9448	0,8356	0,7549	1,2235
10	0,9523	0,9017	0,6571	0,7342
11	0,9497	0,9258	0,621	0,6521
12	0,9615	0,8918	0,5432	0,8438
13	0,9592	0,9234	0,5012	0,5892
14	0,9703	0,8053	0,4589	1,029
15	0,9728	0,9037	0,4012	0,6243
16	0,9691	0,9238	0,3369	0,3987
17	0,9773	0,9199	0,3041	0,3847
18	0,9786	0,9001	0,2573	0,5021
19	0,9754	0,934	0,221	0,3572
20	0,9878	0,9621	0,2154	0,2674
21	0,9805	0,9582	0,2221	0,2689
22	0,9836	0,9581	0,1926	0,2461
23	0,9774	0,9388	0,1874	0,3237
24	0,9808	0,9454	0,1743	0,2649
25	0,9879	0,9495	0,1314	0,2798
26	0,9795	0,9407	0,1584	0,2696
27	0,986	0,9424	0,1179	0,2898

Berdasarkan tabel 5.4, dapat diamati bahwa model mencapai akurasi terbaik pelatihan pada *epoch* ke-25 dengan nilai 0,9879 atau 98,8%. Pada *epoch* ke-27 fungsi *callbacks* aktif dikarenakan tidak terjadi penurunan nilai kerugian (*loss*) pada dataset validasi selama 5 *epoch*. Nilai kerugian (*loss*) pelatihan yang diperoleh sudah cukup kecil yaitu 0,1314. Sedangkan pada *epoch* yang sama, diperoleh akurasi validasi sebesar 0,9495 atau 95%, dengan nilai *loss* validasi sebesar 0,2798. Nilai akurasi antara dataset pelatihan dan validasi yang berada di bawah 5% menunjukkan bahwa model tidak mengalami *overfitting*.

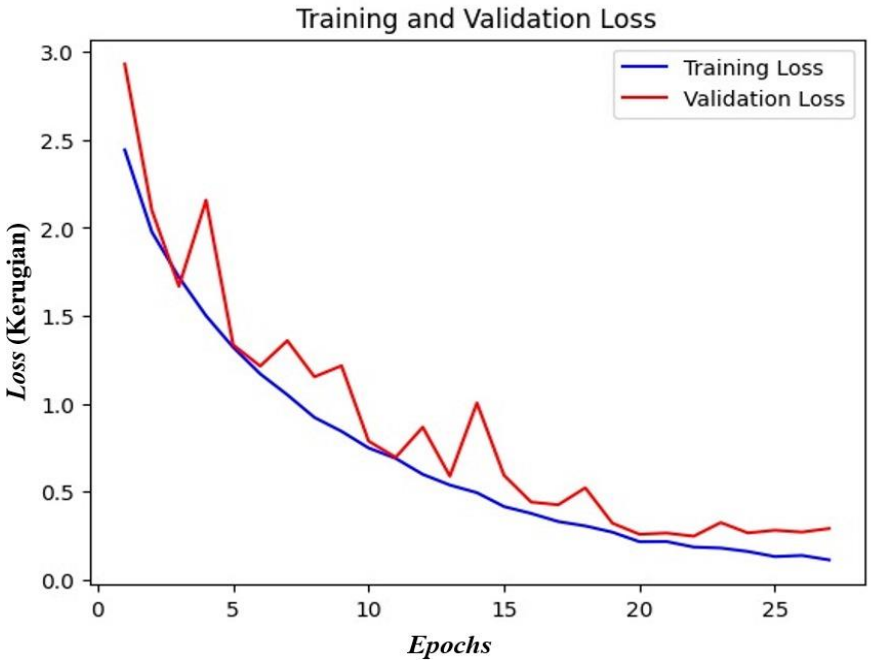
Pada umumnya, ketika nilai akurasi meningkat, *loss* cenderung menurun. Namun, keduanya tidak selalu bergerak seiringan secara sempurna seperti yang terlihat pada beberapa *epoch* dalam tabel di atas. Hal ini bisa terjadi ketika model mulai mendekati titik konvergensi. Selain itu, metrik yang digunakan dalam mengukur akurasi dan *loss* juga dapat menyebabkan perilaku ini. Akurasi hanya melihat prediksi benar atau salah, sedangkan *loss* menghitung seberapa jauh prediksi dari nilai sebenarnya, sehingga nilai *loss* bisa berfluktuasi lebih dibandingkan akurasi.

5.5.3. Evaluasi Model

Berdasarkan hasil pelatihan yang ditunjukkan pada tabel 5.2 akan dilakukan visualisasi hasil akurasi dan kerugian (*loss*) pada proses pelatihan dan validasi dalam sebuah grafik. Visualiasi ini membantu dalam membandingkan hasil dari pelatihan dan validasi. Grafik visualisasi dapat dilihat pada gambar berikut.

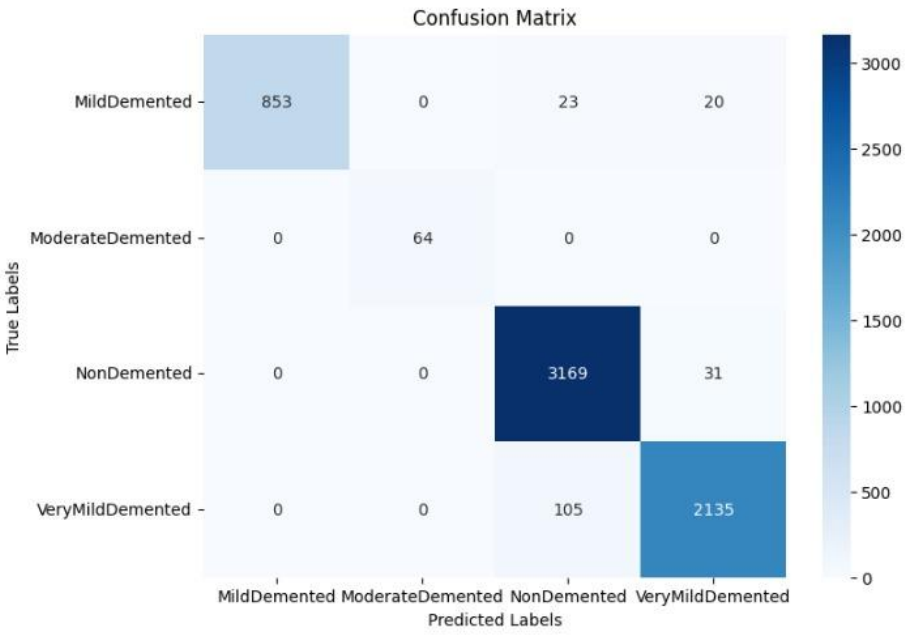


Gambar 5.6 Grafik akurasi pelatihan dan validasi



Gambar 5.7 Grafik kerugian (*loss*) pelatihan dan validasi

Berdasarkan grafik di atas, dapat dikatakan bahwa akurasi pelatihan mengalami peningkatan yang lebih stabil dibandingkan validasi yang cenderung fluktuatif. Grafik kerugian menunjukkan pola yang baik dengan nilai kerugian yang semakin kecil dengan dataset pelatihan memiliki pola yang lebih stabil. Perbedaan kestabilan antara proses pelatihan dan validasi ini wajar terjadi karena perbedaan distribusi data antara dataset pelatihan dan validasi. Selain itu, ukuran dataset pelatihan yang lebih besar memungkinkan model lebih mudah mempelajari pola di dalamnya, sementara pada data validasi, model mungkin menunjukkan hasil yang lebih fluktuatif karena variasi yang lebih tinggi atau kurangnya representasi pola yang serupa. Namun perbedaan tersebut dapat berisiko menyebabkan *overfitting*. Andrew Ng (2011) dalam kursus daring yang dipublikasikannya menjelaskan bahwa selisih 5 – 10% antara akurasi pelatihan dan validasi merupakan indikasi terjadinya *overfitting*. Perbedaan akurasi yang dicapai dalam penelitian ini masih di bawah 5%, sehingga belum menunjukkan tanda-tanda *overfitting*.



Gambar 5.8 *Confusion matrix* hasil pengujian

Berdasarkan *confusion matrix* di atas dapat dilihat bahwa model dapat menebak 6221 gambar dari total 6400 pada dataset pengujian yang belum pernah dilihat sebelumnya. Model berhasil menebak benar 3169 gambar kondisi Non-Demented, 2135 gambar kondisi Very Mild Demented, 853 gambar kondisi Mild Demented, dan 64 gambar kondisi Moderate Demented. *Confusion matrix* menunjukkan bahwa model mengalami sedikit kesulitan dalam membedakan sampel dari beberapa kelas yang memiliki karakteristik serupa, seperti *Non-Demented*, *Very Mild Demented*, dan *Mild Demented*. Secara umum model menunjukkan kinerja yang baik dengan kemampuan generalisasi yang cukup baik terhadap data baru dengan akurasi yang ditunjukkan pada tabel 5.3 berikut.

Tabel 5.3 Metrik evaluasi model

	Precision	*Recall*	*F1-score*	*Support*
Non-Demented	0,96	0,99	0,98	3200
Very Mild Demented	0,98	0,95	0,96	2240
Mild Demented	1,00	0,95	0,98	896
Moderate Demented	1,00	1,00	1,00	64
Accuracy			0,97	6400

Macro Average	0,98	0,97	0,98	6400
Weighted Average	0,97	0,97	0,97	6400

Berdasarkan tabel 5.3, dapat dilihat bahwa model menghasilkan akurasi sebesar 97% pada dataset pengujian. Pada kategori *Non-Demented* mencapai nilai *precision* sebesar 96%, *recall* sebesar 99%, *f1-score* sebesar 98% dengan jumlah data sebanyak 3200 gambar. Pada kategori *Very Mild Demented* mencapai nilai *precision* sebesar 98%, *recall* sebesar 95%, *f1-score* sebesar 96% dengan jumlah data sebanyak 2240 gambar. Pada kategori *Mild Demented* mencapai nilai *precision* sebesar 100%, *recall* sebesar 95%, *f1-score* sebesar 98% dengan jumlah data sebanyak 896 gambar. Pada kategori *Moderate Demented* mencapai nilai *precision* sebesar 100%, *recall* sebesar 100%, *f1-score* sebesar 100% dengan jumlah data sebanyak 64 gambar.

Metrik *macro average* yang merupakan rata-rata metrik dari semua kategori menunjukkan nilai *precision* sebesar 98%, *recall* sebesar 97%, dan *f1-score* sebesar 98%. *Weighted average* menghitung rata-rata metrik dengan bobto yang sebanding dengan jumlah sampel pada tiap kategori menunjukkan nilai *precision* sebesar 97%, *recall* sebesar 97%, dan *f1-score* sebesar 97%. Géron (2022) dalam bukunya menyatakan suatu model *multi class classification* yang baik memiliki metrik evaluasi dengan nilai ambang batas sebesar 80% untuk *precision* dan *recall*, serta 60% untuk *f1-score*.

Secara keseluruhan, performa model dalam proses pelatihan, validasi, dan pengujian menunjukkan hasil yang sangat baik. Selisih antara akurasi pelatihan dan validasi dalam penelitian ini masih di bawah ambang batas yang mengindikasikan *overfitting*. Model juga dapat menggeneralisasi data baru dengan baik, ditunjukkan oleh metrik evaluasi pada dataset pengujian. Perbedaan antara akurasi pelatihan dan validasi dalam penelitian ini masih di bawah ambang batas yang mengindikasikan *overfitting*.

5.6. Perbandingan Hasil Penelitian dengan Penelitian Terdahulu

Pada penelitian yang dilakukan oleh Yildirim dan Cinar (2020), dilakukan pengembangan model deteksi dengan empat kategori yang sama

dengan penelitian ini. Penelitian menggunakan model ResNet50 dalam dua versi yaitu model murni dan model dengan penambahan lapisan tambahan. Akurasi pengujian yang dicapai sebesar 70% untuk model ResNet50 murni dan 90% untuk model dengan lapisan tambahan.

Pada penelitian oleh Buvaneswari dan Gayathri (2021), dilakukan pengembangan model untuk mendeteksi tiga kondisi yaitu normal, kelainan kognitif ringan, dan *Alzheimer*. Dataset yang digunakan merupakan dataset kecil yang hanya berisi 240 gambar. Penelitian menggunakan model dengan dasar arsitektur ResNet101. Akurasi yang dicapai model adalah sebesar 96,3%.

Pada penelitian oleh Ullah dan Jamjoom (2023) menggunakan model yang dibangun sendiri untuk mendeteksi empat kondisi penyakit *Alzheimer* yang sama dengan penelitian ini. Dataset yang digunakan berisi total 6400 gambar. Model mencapai tingkat akurasi yang baik yaitu sebesar 99,38%.

Penelitian dengan objek berbeda dilakukan oleh Aref dan Kareem (2021) dengan model deteksi infeksi Covid-19 dan penelitian oleh Venkataramanan dkk. (2019) dengan model deteksi penyakit pada tanaman. Kedua penelitian ini mencapai akurasi yang baik menggunakan model dengan dasar arsitektur ResNet yang berbeda versi. Pada penelitian Aref dan Kareem (2021) diperoleh akurasi sebesar 96,1% dan 99,5% pada dua dataset berbeda dengan model ResNet50. Sedangkan pada penelitian oleh Venkataramanan dkk. (2019) model dengan dasar ResNet18 memperoleh akurasi sebesar 96%.

Berdasarkan hasil yang diperoleh oleh peneliti, penggunaan *transfer learning* dengan dasar model ResNet152V2 menghasilkan akurasi yang baik dalam mengklasifikasi gambar dalam jumlah yang besar sekalipun. Pengmbangan dalam penelitian dilakukan dengan menggunakan dataset empat tahapan penyakit *Alzheimer* yang lebih besar dan penggunaan model ResNet152V2. Empat kategori tahapan penyakit *Alzheimer* yang dimaksud terdiri dari *Non-Demented*, *Very Mild Demented*, *Mild Demented*, dan *Moderate Demented*. Dataset juga telah melalui pra-proses disesuaikan dengan kebutuhan model. Pengembangan lain adalah penggunaan model dasar ResNet152V2 dengan lapisan tambahan, pemilihan *optimizer* Adamax,

pengaturan *learning rate* sebesar 0,001, dan beberapa parameter lain. Dengan langkah-langkah yang telah dilaksanakan, diperoleh model dengan selisih akurasi pelatihan dan validasi yang kecil, serta akurasi pengujian mencapai 97%.

Tabel 5.4 Perbandingan dengan penelitian terdahulu

Deteksi	**Peneliti**	**Hasil**
Deteksi empat tahapan awal penyakit *Alzheimer* yaitu *Non-Demented*, *Very Mild Demented*, *Mild Demented*, dan *Moderate Demented*.	Yildirim & Cinar. (2020)	• Model yang dibangun dengan arsitektur ResNet50 murni mendapatkan akurasi sebesar 78%. • Model ResNet50 dengan lapisan tambahan mendapat akurasi yang lebih baik, yaitu 90%.
Deteksi tiga kondisi pasien yaitu Normal, Kelainan kognitif ringan, dan *Alzheimer*.	Buvaneswari & Gayathri. (2021)	• Model yang dikembangkan menunjukkan akurasi klasifikasi sebesar 96,3%.
Deteksi empat tahapan awal penyakit *Alzheimer* yaitu *Non Demented*, *Very Mild Demented*, *Mild Demented*, dan *Moderate Demented*.	Ullah & Jamjoom. (2023)	• Model mencapai tingkat akurasi yang baik yaitu sebesar 99,38%.
Identifikasi penyakit pada tanaman dengan mengamati daunnya.	Venkataramanan dkk. (2019)	• Model mencapai nilai akurasi sebesar 96%.

Membandingkan beberapa algoritma untuk deteksi infeksi Covid-19 dengan gambar X-Ray dada pasien	Aref & Kareem. (2021)	• Model yang mencapai performa klasifikasi tertinggi adalah ResNet50 dengan akurasi sebesar 96,1% pada dataset 1 dan 99,5% pada dataset 2.
Deteksi empat tahapan awal penyakit *Alzheimer* yaitu *Non-Demented*, *Very Mild Demented*, *Mild Demented*, dan *Moderate Demented*.	Rui Costa Raka Milanisti (Penelitian ini)	• Model dengan dasar ResNet152V2 mencapai akurasi 98,8% pada pelatihan dan 97% pada pengujian.

BAB VI

KESIMPULAN DAN SARAN

6.1. Kesimpulan

Berdasarkan penelitian yang telah berhasil dilakukan, diperoleh beberapa kesimpulan, yaitu:

1. Pengembangan model untuk mendeteksi tahapan awal penyakit Alzheimer dilakukan menggunakan metode *transfer learning* dengan arsitektur ResNet152V2. Dataset yang digunakan memiliki jumlah data lebih banyak dibandingkan penelitian sebelumnya. Performa model yang baik dalam deteksi dan klasifikasi juga didukung oleh pengaturan serta penggunaan lapisan *batch normalization*, *dense* dengan *regularization*, *dropout* bernilai 0,3, *optimizer* Adamax, laju pembelajaran, dan parameter lainnya.
2. Berdasarkan proses pelatihan model, diperoleh akurasi pelatihan sebesar 98,8% dan akurasi validasi sebesar 95%. Kemampuan generalisasi model terhadap data baru pada tahap pengujian dapat dikatakan sangat baik dengan nilai akurasi pengujian sebesar 97%

6.2. Saran

Penelitian ini dapat dikembangkan lebih lanjut dengan beberapa saran sebagai berikut:

1. Penelitian ini dapat dikembangkan dengan memanfaatkan dataset yang lebih besar dan distribusi data yang lebih merata, serta mencoba arsitektur model dan pengaturan parameter yang berbeda untuk meningkatkan stabilitas dan mempercepat durasi pelatihan.

DAFTAR PUSTAKA

Agarwal, R. (2023, September 13). *Complete Guide to the Adam Optimization Algorithm*. builtin.com.

Alzheimer Indonesia. (2019, April 22). *Statistik tentang Demensia*. Yayasan Alzheimer Indonesia. https://alzi.or.id/statistik-tentang-demensia/

Alzheimer's Disease International. (2019). *World Alzheimer Report 2019: Attitudes to dementia*. https://www.alzint.org/resource/world-alzheimer-report-2019/

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1). https://doi.org/10.1186/s40537-021-00444-8

Amaratunga, T. (2020). Deep Learning on Windows: Building Deep Learning Computer Vision Systems on Microsoft Windows. Dalam *Deep Learning on Windows: Building Deep Learning Computer Vision Systems on Microsoft Windows*. Springer. https://doi.org/10.1007/978-1-4842-6431-7

Aref, N., & Kareem, H. (2021). Detection of Covid-19 Based on Chest Medical Imaging and Artificial Intelligent Techniques: A Review. *Iraqi Journal for Electrical and Electronic Engineering*, *17*(2), 176–182. https://doi.org/10.37917/ijeee.17.2.19

Arnita, Marpaung, F., Aulia, F., Suryani, N., & Cyra Nabila, R. (2022). *COMPUTER VISION DAN PENGOLAHAN CITRA DIGITAL* (A. B. Surya, Ed.; Vol. 1). Pustaka Aksara. www.pustakaaksara.co.id

Banjara, B. (2023, Februari 7). *Deep Residual Learning for Image Recognition (ResNet Explained)*. medium.com. https://medium.com/@bbabina/deep-residual-learning-for-image-recognition-resnet-explained-d2b3c06f7c0a

Belagatti, P. (2024, Maret 12). *Softmax Activation Function in Neural Networks: A Guide to AI/ML Engineers!* https://medium.com/gitconnected/softmax-activation-function-in-neural-networks-a-guide-to-ai-ml-engineers-ebc25b581975

Berger, A. (2002). How does it work?: Magnetic resonance imaging. *BMJ*, *324*(7328), 35–35. https://doi.org/10.1136/bmj.324.7328.35

Bergmann, D., & Stryker, C. (2024, Juli 12). *What is a loss function?* IBM.com. https://www.ibm.com/think/topics/loss-function

Biswal, A. (2023, November 7). *Convolutional Neural Network Tutorial*. simplilearn.com. https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network#layers_in_a_convolutional_neural_network

Borden, J. (2021, Oktober 10). *Alzheimers and Dementia: A Possible Future Free from Illness*. medium.com. https://medium.com/predict/alzheimers-and-dementia-a-possible-future-free-from-illness-8f0a912b8729

Buvaneswari, P. R., & Gayathri, R. (2021). Deep Learning-Based Segmentation in Classification of Alzheimer's Disease. *Arabian Journal for Science and Engineering*, *46*(6), 5373–5383. https://doi.org/10.1007/s13369-020-05193-z

Chaure, N. (2024, April 27). *Variants of ResNet: A Comparative Analysis*. medium.com. https://medium.com/@nayanchaure601/variants-of-resnet-a-comparative-analysis-63fdc1573b34

Coskun, O. (2011). Magnetic resonance imaging and safety aspects. *Toxicology and Industrial Health*, *27*(4), 307–313. https://doi.org/10.1177/0748233710386413

Dan, S., Sharma, D., Rastogi, K., Shaloo, Ojha, H., Pathak, M., & Singhal, R. (2022). Therapeutic and diagnostic applications of nanocomposites in the treatment Alzheimer's disease studies. *Biointerface Research in Applied Chemistry*, *12*(1), 940–960. https://doi.org/10.33263/BRIAC121.940960

Dompeipen, T. A., & Sompie, R. U. A. S. (2020). COMPUTER VISION IMPLEMENTATION FOR DETECTION AND COUNTING THE NUMBER OF HUMANS. *Jurnal Teknik Informatika*, *15*(4). https://doi.org/https://doi.org/10.35793/jti.16.1.2021.31471

Dubey, S. (2019, Desember 27). *Alzheimer's Dataset (4 class of Images)*. Kaggle.com. https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images/data

Ellis, M. E., & Yetman, D. (2024, Maret 28). *What Are the Stages of Alzheimer's Disease?* healthline.com. https://www.healthline.com/health/stages-progression-alzheimers

Feng, W., Halm-Lutterodt, N. Van, Tang, H., Mecum, A., Mesregah, M. K., Ma, Y., Li, H., Zhang, F., Wu, Z., Yao, E., & Guo, X. (2020). Automated MRI-Based Deep Learning Model for Detection of Alzheimer's Disease Process. *International Journal of Neural Systems*, *30*(6). https://doi.org/10.1142/S012906572050032X

Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (3 ed.). O'Reilly Media, Inc.

Gunawardena, N., Rajapakse, R. N., & Kodikara, N. D. (2017). Applying Convolutional Neural Networks for Pre-detection of Alzheimer's Disease from Structural MRI data. Dalam *24th International Conference on Mechatronics and Machine Vision in Practice*.

Harkiran78. (2023, Juni 2). *Artificial Neural Networks and its Applications*. geeksforgeeks.com. https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. http://arxiv.org/abs/1512.03385

Herrmann, N. (2016, November 15). *How Alzheimer's disease is diagnosed*. The Memory Doctor. https://medium.com/the-memory-doctor/how-alzheimers-disease-is-diagnosed-e6e7663adc38

Hosna, A., Merry, E., Gyalmo, J., Alom, Z., Aung, Z., & Azim, M. A. (2022). Transfer learning: a friendly introduction. *Journal of Big Data*, *9*(1). https://doi.org/10.1186/s40537-022-00652-w

Hossin, M., & Sulaiman, M. N. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, *5*(2), 01–11. https://doi.org/10.5121/ijdkp.2015.5201

Huber, J. (2020, November 7). *Batch normalization in 3 levels of understanding*. medium.com. https://medium.com/p/14c2da90a338#5920

Jia, H. (2023). Adaptive Style Transfer Method of Art Works Based on Laplace Operator. *IJACSA) International Journal of Advanced Computer Science and Applications*, *14*(7). www.ijacsa.thesai.org

Karra, S. (2020, Oktober 28). *Evaluations For A Classifier In Machine Learning*. medium.com. https://medium.com/@sailajakarra/evaluations-for-a-classifier-in-machine-learning-1d7cff3e115e

Kılınç, Ç. (2023, Januari 6). *What is TensorFlow?* medium.com. https://medium.com/@cgtyklnc/what-is-tensorflow-9107668fa92d

Klöppel, S., Stonnington, C. M., Barnes, J., Chen, F., Chu, C., Good, C. D., Mader, I., Mitchell, L. A., Patel, A. C., Roberts, C. C., Fox, N. C., Jack, C. R., Ashburner, J., & Frackowiak, R. S. J. (2008). Accuracy of dementia diagnosis - A direct comparison between radiologists and a computerized method. *Brain*, *131*(11), 2969–2974. https://doi.org/10.1093/brain/awn239

Krogh, A. (2008). What are artificial neural networks? Dalam *NATURE BIOTECHNOLOGY* (Vol. 26). http://www.r-project.org/

Kurniawan, A., Atmaja, G., Nawawi, M., Hidayat, O., Latif, A., & Syahputra, R. E. (2023). Sistem Pakar Diagnosa Kerusakan Mesin Sepeda Motor Dengan Menggunakan Metode Forward Chaining. *Teknik dan Multimedia*, *1*(2). https://scholar.google.com.

Lestari, E., & Rahayu, W. I. (2023). PREDIKSI KEGANASAN KANKER PAYUDARA DENGAN PENDEKATAN MACHINE LEARNING: SYTEMATIC LITERATURE REVIEW. Dalam *Jurnal Mahasiswa Teknik Informatika* (Vol. 7, Nomor 3).

Lina, Q. (2019, Januari 2). *Apa itu Convolutional Neural Network?* medium.com. https://medium.com/@16611110/apa-itu-convolutional-neural-network-836f70b193a4

Mishra, M. (2020, Agustus 27). *Convolutional Neural Networks, Explained*. towardsdatascience.com. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

Mkale. (2022, Januari 10). *ReLU Activation Function*. https://medium.com/@mkale9067/relu-activation-function-f5cd4cc3033f

Moltzau, A. (2019, September 18). *The Evolution of Python and The Fear of Dying Languages*. medium.com. https://medium.com/@alexmoltzau/the-evolution-of-python-and-the-fear-of-dying-languages-69dc18d0d660

Mulya, M. A., Zaenul Arif, & Syefudin. (2023). Tinjauan Pustaka Sistematis : Penerapan Metode Gabor Wavelet Pada Computer Vision. *Journal Of Computer Science And Technology (JOCSTEC)*, *1*(2), 83–88. https://doi.org/10.59435/jocstec.v1i2.78

Musstafa. (2021, Maret 28). *Optimizers in Deep Learning*. medium.com. https://medium.com/@musstafa0804/optimizers-in-deep-learning-7bf81fed78a0

Ng, A. (2011). *Supervised Machine Learning: Regression and Classification*. Coursera. https://www.coursera.org/learn/machine-learning

Panneerselvam, L. (2024, Maret 15). *Activation Functions Neural Networks: A Quick & Complete Guide*.

Pini, L., Pievani, M., Bocchetta, M., Altomare, D., Bosco, P., Cavedo, E., Galluzzi, S., Marizzoni, M., & Frisoni, G. B. (2016). Brain atrophy in Alzheimer's Disease and aging. *Ageing Research Reviews*, *30*, 25–48. https://doi.org/10.1016/j.arr.2016.01.002

Shorten, C. (2019, Januari 25). *Introduction to ResNets*. towards data science. https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4

Shoumi, M. N., Syulistyo, A. R., Wijayaningrum, V. N., & Yunhasnawa, Y. (2022). *Teori dan Aplikasi Kecerdasan Buatan Menggunakan Python*.

Sianturi, A. G. M. (2021). Stadium, Diagnosis, dan Tatalaksana Penyakit Alzheimer. *Majalah Kesehatan Indonesia*, *2*(2), 39–44. https://doi.org/10.47679/makein.202132

Srivastava, N. (2013). *Improving Neural Networks with Dropout* [Master of Science Degree]. University of Toronto.

Sumin, & Prihantono. (2020). *Pemodelan Jaringan Syaraf Tiruan*. https://digilib.iainptk.ac.id/xmlui/handle/123456789/2764

Susanti, N., Hairini Siregar, N., Ramadhani, N., & Sihite, R. N. (2024). *ALZHEIMER DAN DIMENSIA*. *5*(2).

Ullah, Z., & Jamjoom, M. (2023). A Deep Learning for Alzheimer's Stages Detection Using Brain Images. *Computers, Materials and Continua*, *74*(1), 1457–1473. https://doi.org/10.32604/cmc.2023.032752

Uraninjo. (2022, September 21). *Uraninjo Dataset*. Kaggle.com. https://www.kaggle.com/datasets/uraninjo/augmented-alzheimer-mri-dataset-v2

Venkataramanan, A., Kumar, D., Honakeri, P., & Agarwal, P. (2019). *Plant Disease Detection and Classification Using Deep Neural Networks*.

Wahidi, D. D. (2021, Juli 13). *TensorFlow.Keras*. Howdy Sysinfo. https://medium.com/sysinfo/tensorflowkeras-66dd489ae52f

Wijaya, A. E., Swastika, W., & Kelana, O. H. (2021). IMPLEMENTASI TRANSFER LEARNING PADA CONVOLUTIONAL NEURAL NETWORK UNTUK DIAGNOSIS COVID-19 DAN PNEUMONIA PADA CITRA X-RAY. Dalam *SAINSBERTEK Jurnal Ilmiah Sains & Teknologi* (Vol. 2).

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, *9*(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

Yildirim, M., & Cinar, A. (2020). Classification of Alzheimer's disease MRI images with CNN based hybrid method. *Ingenierie des Systemes d'Information*, *25*(4), 413–418. https://doi.org/10.18280/isi.250402

Zhang, Q., Yu, H., Barbiero, M., Wang, B., & Gu, M. (2019). Artificial neural networks enabled by nanophotonics. Dalam *Light: Science and Applications* (Vol. 8, Nomor 1). Nature Publishing Group. https://doi.org/10.1038/s41377-019-0151-0

LAMPIRAN

Source Code

1. Import Library

```
[ ] import os
    import numpy as np
    import pandas as pd

    from mpl_toolkits.axes_grid1 import ImageGrid
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    import seaborn as sns

    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras import regularizers
    from tensorflow.keras.preprocessing import image
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.optimizers import Adam, Adamax
    from tensorflow.keras.metrics import categorical_crossentropy
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
    from tensorflow.keras.callbacks import EarlyStopping
    from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
    from sklearn.model_selection import train_test_split
    from PIL import Image

    import warnings
    warnings.filterwarnings('ignore')
```

2. Load Data

```
[ ] # Mengunduh Dataset dari Google Drive
    !gdown --id 1AQLWiTNzwrIQQRjmpiylho9uAHRBmqdG -O MRI.zip
```

```
/usr/local/lib/python3.10/dist-packages/gdown/__main__.py:140: FutureWarning: Option `--id` was deprec
  warnings.warn(
Downloading...
From (original): https://drive.google.com/uc?id=1AQLWiTNzwrIQQRjmpiylho9uAHRBmqdG
From (redirected): https://drive.google.com/uc?id=1AQLWiTNzwrIQQRjmpiylho9uAHRBmqdG&confirm=t&uuid=730
To: /content/MRI.zip
100% 397M/397M [00:05<00:00, 68.3MB/s]
```

```
[ ] #unzip data
    !unzip -q '/content/MRI.zip' -d 'MRI'
```

```
[ ] #Defining
    SAMPLE_PER_CATEGORY = 200
    #SEED = 42
    WIDTH = 224
    HEIGHT = 224
    DEPTH = 3
    INPUT_SHAPE = (WIDTH, HEIGHT, DEPTH)

    data_dir = '/content/MRI/data'
    train_dir = os.path.join(data_dir, 'train')
    test_dir = os.path.join(data_dir, 'val')
```

3. Exploratory Data Analysis (EDA)

```
[ ] CATEGORIES = ['MildDemented', 'ModerateDemented', 'NonDemented', 'VeryMildDemented']
    NUM_CATEGORIES = len(CATEGORIES)
    NUM_CATEGORIES
```

```
4
```

```
[ ] # prompt: checking entire dataset image extensions

    import os

    def check_image_extensions(data_dir):
      """Checks the extensions of all images in a directory."""
      extensions = set()
      for root, _, files in os.walk(data_dir):
        for file in files:
          if file.lower().endswith(('.jpg', '.jpeg', '.png')):
            extensions.add(os.path.splitext(file)[1].lower())
      return extensions

    extensions = check_image_extensions(data_dir)
    print("Image extensions found in the dataset:", extensions)
```

```
Image extensions found in the dataset: {'.jpg'}
```

```
print('Jumlah Data Tiap Kategori dalam Data Pelatihan')
total_train = 0
for category in CATEGORIES:
    print('\nJumlah gambar dari kategori {} = {}'.format(category, len(os.listdir(os.path.join(train_dir, category)))))
    print('-'*50)
    total_train += len(os.listdir(os.path.join(train_dir, category)))

print(f'\nTotal Data = {total_train}')
```

```
Jumlah Data Tiap Kategori dalam Data Pelatihan

Jumlah gambar dari kategori MildDemented = 8960
--------------------------------------------------

Jumlah gambar dari kategori ModerateDemented = 6464
--------------------------------------------------

Jumlah gambar dari kategori NonDemented = 9600
--------------------------------------------------

Jumlah gambar dari kategori VeryMildDemented = 8960
--------------------------------------------------

Total Data = 33984
```

```
print('Jumlah Data Tiap Kategori dalam Data Uji')
total_test = 0
for category in CATEGORIES:
    print('\nJumlah gambar dari kategori {} = {}'.format(category, len(os.listdir(os.path.join(test_dir, category)))))
    print('-'*50)
    total_test += len(os.listdir(os.path.join(test_dir, category)))

print(f'\nTotal Data = {total_test}')
```

```
Jumlah Data Tiap Kategori dalam Data Uji

Jumlah gambar dari kategori MildDemented = 896
--------------------------------------------------

Jumlah gambar dari kategori ModerateDemented = 64
--------------------------------------------------

Jumlah gambar dari kategori NonDemented = 3200
--------------------------------------------------

Jumlah gambar dari kategori VeryMildDemented = 2240
--------------------------------------------------

Total Data = 6400
```

4. Creating Train and Validation DataFrame

```
train = []
for category_id, category in enumerate(CATEGORIES):
    for file in os.listdir(os.path.join(train_dir, category)):
        train.append(['train/{}/{}'.format(category, file), category_id, category])
train = pd.DataFrame(train, columns=['file', 'category_id', 'category'])
train.shape
```

```
(33984, 3)
```

```
train = train.sample(frac=1)
X = train.drop(columns = 'category_id')
y = train['category_id']
```

```
x_train, x_valid, y_train, y_valid = train_test_split(X, y, test_size=0.25, random_state=4)
```

```
train = pd.concat([x_train, y_train], axis=1)
validation = pd.concat([x_valid, y_valid], axis=1)
```

```
train = train.reset_index()
train = train.drop(columns = 'index')
validation = validation.reset_index()
validation = validation.drop(columns = 'index')
print(train.shape)
print(validation.shape)
```

```
(25488, 3)
(8496, 3)
```

```
train.head()
```

	file	category	category_id
0	train/NonDemented/594ece05-c73d-4d99-abd2-60f1...	NonDemented	2
1	train/NonDemented/c262b95c-2cff-4ec4-a0ef-e6c4...	NonDemented	2
2	train/NonDemented/70efb58f-254f-491d-a295-a0b2...	NonDemented	2
3	train/NonDemented/47fba467-b289-4088-abb3-562a...	NonDemented	2
4	train/ModerateDemented/8e099798-cc6e-4e17-adf6...	ModerateDemented	1

```
validation.head()
```

	file	category	category_id
0	train/NonDemented/d31788ec-c337-4848-9ead-1901...	NonDemented	2
1	train/MildDemented/9f9b7c22-5ea7-4c5a-89ab-f01...	MildDemented	0
2	train/MildDemented/004dd801-1100-4d89-985d-ef6...	MildDemented	0
3	train/VeryMildDemented/b4e119b7-7649-4125-b535...	VeryMildDemented	3
4	train/VeryMildDemented/275d0f75-4d51-4592-89dd...	VeryMildDemented	3

```
# prompt: images count in each categories of train dataframe and total

# Calculate the number of images in each category for the training data
train_counts = train.groupby('category')['file'].count()

print(train_counts)
print(f"\nTotal images in train dataframe: {train_counts.sum()}")
```

```
category
MildDemented        6715
ModerateDemented    4864
NonDemented         7214
VeryMildDemented    6695
Name: file, dtype: int64

Total images in train dataframe: 25488
```

```
# prompt: images count in each categories of validation dataframe and total

# Calculate the number of images in each category for the validation data
validation_counts = validation.groupby('category')['file'].count()

print(validation_counts)
print(f"\nTotal images in validation dataframe: {validation_counts.sum()}")
```

```
category
MildDemented        2245
ModerateDemented    1600
NonDemented         2386
VeryMildDemented    2265
Name: file, dtype: int64

Total images in validation dataframe: 8496
```

5. Creating Test DataFrame

```
test = []
for category_id, category in enumerate(CATEGORIES):
    for file in os.listdir(os.path.join(test_dir, category)):
        test.append(['val/{}/{}'.format(category, file), category_id, category])
test = pd.DataFrame(test, columns=['file', 'category_id', 'category'])
test.shape
```

```
(6400, 3)
```

```
test.head()
```

	file	category_id	category
0	val/MildDemented/mildDem491.jpg	0	MildDemented
1	val/MildDemented/mildDem83.jpg	0	MildDemented
2	val/MildDemented/mildDem117.jpg	0	MildDemented
3	val/MildDemented/mildDem202.jpg	0	MildDemented
4	val/MildDemented/30 (27).jpg	0	MildDemented

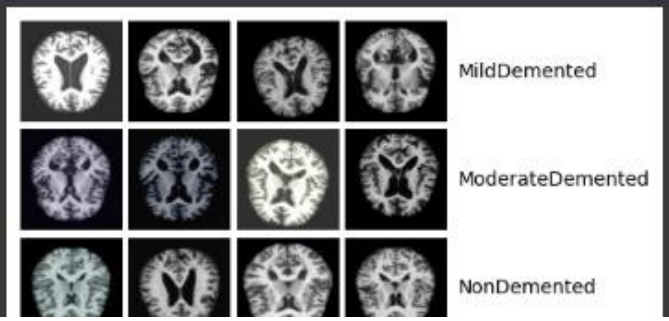
```
# prompt: images count in each categories of test dataset and total

# Calculate the number of images in each category for the test data
test_counts = test.groupby('category')['file'].count()

print(test_counts)
print(f"\nTotal images in test dataframe: {test_counts.sum()}")
```

```
category
MildDemented          896
ModerateDemented       64
NonDemented          3200
VeryMildDemented     2240
Name: file, dtype: int64

Total images in test dataframe: 6400
```

6. Demonstrating Example Images

```
def read_img(filepath, size):
    img = image.load_img(os.path.join(data_dir, filepath), target_size=size)
    img = image.img_to_array(img)
    return img
```

```
fig = plt.figure(1, figsize=(NUM_CATEGORIES, NUM_CATEGORIES))
grid = ImageGrid(fig, 111, nrows_ncols=(NUM_CATEGORIES, NUM_CATEGORIES), axes_pad=0.05)

i=0
for category_id, category in enumerate(CATEGORIES):
    for filepath in train[train['category'] == category]['file'].values[:NUM_CATEGORIES]:
        ax = grid[i]
        img = read_img(filepath, (WIDTH, HEIGHT))
        ax.imshow(img / 255.)
        ax.axis('off')
        if i % NUM_CATEGORIES == NUM_CATEGORIES - 1:
            ax.text(250, 112, filepath.split('/')[1], verticalalignment='center')
        i+=1

plt.show();
```



7. Keras ImageDataGenerator

```
datagen_train = ImageDataGenerator(rescale=1./255)
train_generator = datagen_train.flow_from_dataframe(dataframe=train,
                                                    directory="/content/MRI/data",
                                                    x_col="file",
                                                    y_col="category",
                                                    batch_size=64,
                                                    shuffle=True,
                                                    class_mode="categorical",
                                                    target_size=(HEIGHT, WIDTH));

validation_generator = datagen_train.flow_from_dataframe(dataframe=validation,
                                                    directory="/content/MRI/data",
                                                    x_col="file",
                                                    y_col="category",
                                                    batch_size=64,
                                                    shuffle=True,
                                                    class_mode="categorical",
                                                    target_size=(HEIGHT, WIDTH));
```

```
Found 25488 validated image filenames belonging to 4 classes.
Found 8496 validated image filenames belonging to 4 classes.
```

```
datagen_test = ImageDataGenerator(rescale=1./255)
test_generator = datagen_test.flow_from_dataframe(dataframe=test,
                                                  directory="/content/MRI/data",
                                                  x_col="file",
                                                  y_col="category",
                                                  batch_size=64,
                                                  shuffle=False,
                                                  class_mode="categorical",
                                                  target_size=(HEIGHT, WIDTH));
```

```
Found 6400 validated image filenames belonging to 4 classes.
```

8. Early Stopping

```
[ ] early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

9. Creating Model Based on ResNet152V2

```
#Building Model
def create_model():

    resnet_model = tf.keras.applications.resnet_v2.ResNet152V2(
        weights='imagenet',
        include_top = False,
        input_shape = (224, 224, 3)
    )

    for layers in resnet_model.layers[:100]:
        layers.trainable = False #freeze first number of layers of resnet
    for layers in resnet_model.layers[100:]:
        layers.trainable = True #unfreeze layers start from indexed number

    x = resnet_model.output
    x = tf.keras.layers.GlobalAveragePooling2D()(x)

    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    x = tf.keras.layers.Dense(1024, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001))(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    x = tf.keras.layers.Dense(512, activation='relu')(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    x = tf.keras.layers.Dense(256, activation='relu')(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    x = tf.keras.layers.Dense(128, activation='relu')(x)
```

```
    x = tf.keras.layers.Dropout(0.3)(x)
    x = tf.keras.layers.Dense(128, activation='relu')(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    # output layer
    predictions = tf.keras.layers.Dense(4, activation='softmax')(x)

    res_model = tf.keras.Model(inputs=resnet_model.input, outputs=predictions)

    # Compiling the model
    res_model.compile(loss='categorical_crossentropy', optimizer=Adamax(learning_rate = 0.001), metrics=['accuracy'])
    return res_model
```

```
[ ] res_model = create_model()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet152v2_weights_tf_dim_ordering_tf_kernels_notop.h5
234545216/234545216 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

```
[ ] res_model.summary()
```

```
Model: "functional"
```

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,472	conv1_pad[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_conv[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_preact_bn (BatchNormalization)	(None, 56, 56, 64)	256	pool1_pool[0][0]
conv2_block1_preact_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_preact_b…

10. Training the Model

```
[ ] history = res_model.fit(train_generator,
                            epochs=50,
                            steps_per_epoch=100,
                            validation_data=validation_generator,
                            callbacks=[early_stopping],
                            batch_size=48)
```

```
Epoch 1/50
100/100 ━━━━━━━━━━━━━━━━━━━━ 322s 2s/step - accuracy: 0.3922 - loss: 2.6849 - val_accuracy: 0.4401 - val_loss: 3.6083
Epoch 2/50
100/100 ━━━━━━━━━━━━━━━━━━━━ 230s 2s/step - accuracy: 0.6409 - loss: 2.0102 - val_accuracy: 0.6463 - val_loss: 1.9703
Epoch 3/50
100/100 ━━━━━━━━━━━━━━━━━━━━ 175s 2s/step - accuracy: 0.7403 - loss: 1.7431 - val_accuracy: 0.7652 - val_loss: 1.6007
Epoch 4/50
100/100 ━━━━━━━━━━━━━━━━━━━━ 174s 2s/step - accuracy: 0.8170 - loss: 1.5437 - val_accuracy: 0.8265 - val_loss: 1.4604
Epoch 5/50
```

10. Evaluation

```
[ ] valid_loss, valid_accuracy = res_model.evaluate(validation_generator)

    print(f'\nTraining loss: {valid_loss:.2f}')
    print(f'Training Accuracy: {valid_accuracy*100:.2f} %')
```

```
 77/133 ━━━━━━━━━━━━━━━━━━━━ 23s 416ms/step - accuracy: 0.9770 - loss: 0.0983
```

```
[ ] loss, accuracy = res_model.evaluate(test_generator)

    print(f'\nTest loss: {loss:.2f} ')
    print(f'Test Accuracy: {accuracy*100:.2f} %')
```

```
[ ] y_predict = res_model.predict(test_generator)
    prediction = np.argmax(y_predict,axis=1)
    labels = (train_generator.class_indices)
    labels = dict((v,k) for k,v in labels.items())
    prediction = [labels[k] for k in prediction]

    y_test = list(test.category)

    report = classification_report(y_test, prediction, output_dict=True)
    df = pd.DataFrame(report).transpose()
    df
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc)+1)

#Plotting training and validation accuracy graph
plt.plot(epochs, acc, 'b', label='Training Accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation Accuracy')
plt.title('Training and Validation Accuracy', x='Epochs', y='Accuracy')
plt.legend()

#Plotting training and validation loss graph
plt.figure()
plt.plot(epochs, loss, 'b', label='Training Loss')
plt.plot(epochs, val_loss, 'r', label='Validation Loss')
plt.title('Training and Validation Loss', x='Epochs', y='Loss')
plt.legend()

plt.show()
```

```
[ ] #Test data confusion matrix
    def create_classification_report(model_name):
      y_pred = model_name.predict(test_generator)
      y_pred = np.argmax(y_pred, axis=1)
      y_true = test_generator.classes

      cm = confusion_matrix(y_true, y_pred)
      class_names = list(test_generator.class_indices.keys())

      plt.figure(figsize=(8, 6), dpi=100)
      sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
      plt.xlabel('Predicted Labels')
      plt.ylabel('True Labels')
      plt.title('Confusion Matrix')
      #plt.savefig('confusion_matrix.png')
```