Pemrosesan Citra Medis: Reduksi *Noise* pada Citra MRI Otak dengan *Convolutional Autoencoder*

I Gede Susrama Mas Diyasa

PT. GHANI PRESS GROUP

Pemrosesan Citra Medis: Reduksi Noise pada Citra MRI Otak dengan *Convolutional Autoencoder*

Penulis:
I Gede Susrama Mas Diyasa

Desain Cover:
Umar Khasan

Layouter:
Tyas Dzawil Istiqomah

Editor Naskah:
Dr. Yeni Kustiyahningsih, S.Kom., M.Kom.

14 x 21 cm, v-165
ISBN: 978-634-05-1621-0
Anggota IKAPI: No. 468/JTI/2026

Diterbitkan oleh: PT GHANI PRESS GROUP

Alamat:
Bumi Permata Raya Blok 8 No. 16 Tanjung, Lamongan
HP. 0896-5710-0105
Email: ptghanipress@gmail.com
Website: https://ghanipress.com/

Sanksi Pelanggaran Pasal 113
Undang-Undang No. 28 Tahun 2014 Tentang Hak Cipta

a. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp 100.000.000 (seratus juta rupiah).
b. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah).
c. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/atau pidana denda paling banyak Rp 1.000.000.000,00 (satu miliar rupiah).
d. Setiap Orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan/atau pidana denda paling banyak Rp 4.000.000.000,00 (empat miliar rupiah).

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas selesainya penyusunan buku yang berjudul "Pemrosesan Citra Medis: Reduksi Noise pada Citra MRI Otak dengan *Convolutional Autoencoder*". Kemajuan teknologi di bidang medis telah memungkinkan berbagai inovasi dalam metode diagnosis dan pengobatan.

Pembahasan diawali dengan menegaskan peran penting citra medis sebagai instrumen utama dalam proses diagnosis modern, untuk kemudian mengarah pada persoalan inti yang dihadapi pada pencitraan *Magnetic Resonance Imaging* (MRI) otak, yaitu keberadaan noise yang menurunkan kualitas visual citra dan berpotensi mengganggu ketepatan interpretasi klinis. Atas dasar tantangan tersebut, diuraikan motivasi pemilihan *Convolutional Autoencoder* (CAE) sebagai pendekatan reduksi noise.

Semoga buku ini dapat memberikan manfaat, memperluas wawasan, serta menjadi sumber referensi yang berguna bagi seluruh pembaca tentang reduksi noise pada citra MRI otak dengan *convolutional autoencoder*.

Surabaya, Juni 2026

Penulis

DAFTAR ISI

PT GHANI PRESS GROUP
Email: ptghanipress@gmail.com
Website: https://ghanipress.com/

BAB I
PENDAHULUAN

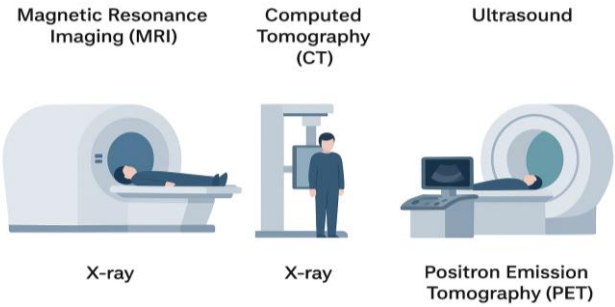
➢ Peran Penting Citra Medis dalam Diagnosis

Kemajuan teknologi di bidang medis telah memungkinkan berbagai inovasi dalam metode diagnosis dan pengobatan. Salah satu aspek penting dalam teknologi medis adalah pengolahan citra medis yang berperan besar dalam mendukung pengambilan keputusan klinis. Seiring dengan meningkatnya kebutuhan akan citra medis yang berkualitas tinggi, muncul pula tantangan baru terkait kualitas citra yang terganggu oleh *noise* dan artefak. Metode yang tepat diperlukan untuk meningkatkan kualitas citra guna memastikan hasil diagnosis yang lebih akurat

Peran penting citra medis dalam perawatan kesehatan, berfungsi sebagai alat utama untuk diagnosis dan analisis berbagai kondisi. Citra medis mengacu pada pola atau citra dua dimensi yang mewakili struktur internal tubuh manusia. Beberapa teknik digunakan untuk menangkap citra medis, salah satunya merupakan *Magnetic Resonance Imaging* (MRI) yang memanfaatkan medan magnet untuk menghasilkan citra rinci organ dan jaringan. Kualitas citra MRI terkadang dapat mengalami masalah kualitas karena *noise* dan artefak, yang dapat mempersulit proses diagnostik sehingga diperlukanya metode untuk meningkatkan kualitas citra MRI [1].

Selain MRI, terdapat berbagai modalitas citra medis lainnya seperti *Computed Tomography* (CT), *X-ray*, *Ultrasound*, dan *Positron Emission Tomography* (PET), yang masing-masing

memiliki karakteristik unik dalam memvisualisasikan struktur dan fungsi tubuh, seperti pada Gambar 1 Modalitas tersebut memiliki peran spesifik, mulai dari radiografi sederhana untuk mendeteksi fraktur tulang hingga pencitraan molekuler canggih untuk menilai metabolisme jaringan. Keberhasilan interpretasi citra medis sangat bergantung pada kualitas detail dan ketepatan representasi anatomi atau kelainan yang diamati, sehingga citra yang berkualitas rendah dapat menyebabkan kesalahan diagnosis, penundaan pengobatan, atau bahkan pemberian terapi yang tidak tepat. Dalam praktik klinis, kualitas citra sangat dipengaruhi oleh kondisi pasien, pengaturan alat, dan keterampilan operator, sehingga strategi peningkatan kualitas menjadi aspek yang tidak terpisahkan dari prosedur pencitraan.



Gambar 1. Modalitas Citra Medis

Untuk mengatasi masalah tersebut, para peneliti dan praktisi medis memanfaatkan berbagai teknik pemrosesan citra digital, mulai dari metode konvensional seperti *filtering* dan histogram *equalization* hingga metode canggih berbasis pembelajaran mesin dan kecerdasan buatan. *Filtering* digunakan untuk mengurangi *noise* yang mengganggu, sementara teknik seperti *Contrast Limited Adaptive Histogram Equalization* (CLAHE) dapat meningkatkan kontras citra tanpa

memperkuat *noise* secara berlebihan. Di sisi lain, pendekatan modern berbasis transformasi seperti *wavelet transform* dan *anisotropic diffusion* menawarkan kemampuan untuk menghilangkan *noise* sambil mempertahankan tepi objek, yang penting untuk menjaga kejelasan struktur anatomi.

Kemajuan terbaru di bidang kecerdasan buatan, khususnya deep learning, telah membuka peluang baru dalam peningkatan kualitas citra medis. *Convolutional Neural Networks* (CNN) digunakan untuk melakukan *denoising* pada citra MRI dan CT secara otomatis, sementara *Generative Adversarial Networks* (GAN) dimanfaatkan untuk menghasilkan citra resolusi tinggi dari data resolusi rendah atau terbatas. Pendekatan ini tidak hanya memperbaiki kualitas visual, tetapi juga meningkatkan nilai diagnostik citra, karena detail penting yang sebelumnya tereduksi dapat dipulihkan. Selain itu, teknik transfer learning memungkinkan penggunaan model yang telah dilatih pada domain lain untuk diterapkan pada pencitraan medis, sehingga mempercepat proses pengembangan model baru.

Integrasi pengolahan citra medis berbasis AI ke dalam sistem klinis semakin memperkuat perannya dalam mendukung pengambilan keputusan. Sistem *Computer-Aided Diagnosis* (CAD) kini dapat memberikan analisis kuantitatif yang membantu radiolog dalam mengidentifikasi lesi, mengukur ukuran tumor, atau menilai progres penyakit dari waktu ke waktu. Dengan dukungan algoritma yang mampu mengolah data citra dalam jumlah besar secara cepat dan konsisten, beban kerja tenaga medis dapat berkurang, sementara akurasi diagnosis dapat meningkat. Hal ini sangat penting di rumah sakit dengan volume pasien tinggi atau di daerah dengan keterbatasan tenaga spesialis radiologi.

Masa depan pengolahan citra medis akan mengarah pada integrasi multi-modalitas, di mana data dari berbagai teknik pencitraan seperti MRI, CT, PET, dan *ultrasound* digabungkan untuk memberikan gambaran komprehensif tentang kondisi pasien. Penggabungan ini tidak hanya memperkaya informasi diagnostik, tetapi juga memungkinkan pendekatan personalisasi pengobatan yang lebih tepat sasaran. Di sisi lain, pengembangan pencitraan kuantitatif berbasis biomarker visual akan memberikan kemampuan untuk mengukur parameter fisiologis secara langsung dari citra, sehingga diagnosis dapat dibuat tidak hanya berdasarkan pengamatan visual, tetapi juga data kuantitatif yang objektif.

Dengan demikian, pengolahan citra medis tidak hanya merupakan langkah teknis tambahan, tetapi sudah menjadi fondasi penting dalam praktik kedokteran modern. Kombinasi antara perangkat pencitraan yang semakin canggih, metode pemrosesan citra yang inovatif, serta integrasi kecerdasan buatan akan terus memperkuat peran citra medis dalam memberikan diagnosis yang akurat, cepat, dan aman bagi pasien. Perkembangan ini menunjukkan bahwa kolaborasi antara tenaga medis, ilmuwan komputer, dan insinyur merupakan kunci dalam mendorong inovasi yang dapat meningkatkan kualitas pelayanan kesehatan secara global, sekaligus mengurangi risiko kesalahan diagnosis yang dapat berdampak serius pada keselamatan pasien.

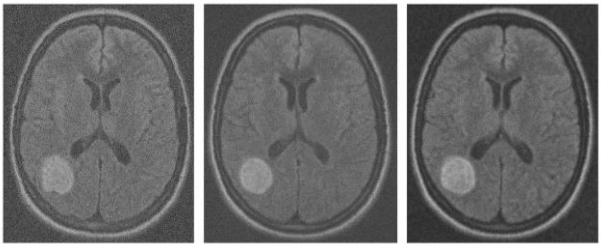
➢ Tantangan Utama: *Noise* pada Citra MRI Otak

Di bidang medis, terutama dengan pemindaian MRI, pemrosesan citra sangat penting. Pemrosesan citra mengambil citra dari dunia nyata dan mengubahnya menjadi format digital. Selama proses ini, beberapa kualitas dapat hilang, dan bagian yang tidak diinginkan yang disebut aretfak atau *noise*

dapat muncul. Dengan perbaikan citra digital, membuat citra lebih jelas dan lebih berguna sehingga dapat membantu dokter memahami tubuh manusia dan mendiagnosis penyakit secara akurat. Jika citra tidak jelas, dapat menyebabkan kesimpulan yang salah tentang kesehatan pasien [2].

Dalam konteks MRI otak, keberadaan *noise* menjadi salah satu tantangan paling krusial yang dihadapi oleh radiolog dan peneliti. *Noise* adalah gangguan acak yang menurunkan kualitas visual citra sehingga detail halus dari struktur otak menjadi sulit dikenali. Hal ini memiliki dampak besar, terutama dalam diagnosis penyakit neurologis seperti tumor otak, stroke, *multiple sclerosis*, atau penyakit *Alzheimer* yang membutuhkan visualisasi jaringan secara rinci. *Noise* dapat muncul dari berbagai sumber, mulai dari keterbatasan teknis alat, pergerakan pasien selama pemindaian, gangguan sinyal elektromagnetik, hingga kesalahan dalam proses rekonstruksi citra. Sifat sensitif MRI terhadap faktor lingkungan dan fisiologis membuat masalah *noise* menjadi kompleks dan memerlukan pendekatan multidisiplin untuk diatasi.

Jenis-jenis *noise* yang sering muncul pada citra MRI otak antara lain *Gaussian noise*, Rician *noise*, dan *Speckle noise*. *Gaussian noise* biasanya disebabkan oleh fluktuasi acak dalam sistem elektronik pemindai, seperti terlihat pada Gambar 2.



Gambar 2. Citra MRI otak Jenis (a) *Gaussian Noise*, (b) *Rician Noise*, dan (c) *Speckle Noise*

Rician *noise* sering muncul pada citra magnitude MRI dan cenderung mempengaruhi area dengan intensitas rendah, sehingga struktur halus otak menjadi kurang jelas. Sementara itu, *Speckle noise*, meskipun lebih umum pada citra berbasis gelombang ultrasonik, juga dapat terjadi pada MRI akibat fenomena interferensi gelombang. Keberadaan *noise* ini dapat mengaburkan batas antara jaringan normal dan patologis, yang pada akhirnya menurunkan akurasi diagnosis.

Dampak *noise* pada citra MRI otak tidak hanya terbatas pada penurunan kualitas visual, tetapi juga mempengaruhi analisis kuantitatif yang dilakukan oleh perangkat lunak pendukung diagnosis. Pembahasan *neuroimaging*, analisis statistik terhadap parameter seperti volume otak, ketebalan korteks, atau tingkat difusi air dalam jaringan sangat bergantung pada keakuratan data citra. Jika *noise* tidak dikelola dengan baik, hasil analisis bisa menyimpang, yang berisiko menghasilkan interpretasi yang keliru. Misalnya, keberadaan *noise* dapat menyebabkan *overestimasi* atau *underestimasi* ukuran lesi, sehingga mempengaruhi keputusan terapeutik.

Upaya untuk mengatasi *noise* pada citra MRI otak telah berkembang pesat. Secara tradisional, metode *filtering* seperti *Gaussian filter*, *median filter*, atau *anisotropic diffusion* digunakan untuk mereduksi *noise* sambil mempertahankan tepi objek. Namun, metode konvensional ini sering menghadapi dilema antara pengurangan *noise* dan pelestarian detail. Perkembangan teknologi komputasi membawa pendekatan baru berbasis domain frekuensi, seperti *discrete wavelet transform* (DWT) dan *Fourier transform*, yang memungkinkan penghapusan *noise* pada skala tertentu tanpa menghilangkan informasi penting.

Dalam satu dekade terakhir, kemajuan di bidang kecerdasan buatan, khususnya *deep learning*, telah merevolusi pendekatan perbaikan citra MRI. Model berbasis *Convolutional Neural Networks* (CNN) dan *Generative Adversarial Networks* (GAN) mampu mempelajari pola *noise* dan secara adaptif menghapusnya dari citra, bahkan menghasilkan detail yang mendekati kondisi ideal. Pendekatan ini tidak hanya meningkatkan kualitas visual, tetapi juga meningkatkan nilai diagnostik citra. Sebagai contoh, GAN dapat digunakan untuk merekonstruksi citra resolusi tinggi dari data resolusi rendah yang terpengaruh *noise*, sehingga memudahkan deteksi lesi kecil pada otak.

Selain itu, strategi pengurangan *noise* modern sering menggabungkan metode fisik dan komputasi. Dari sisi fisik, inovasi pada desain *coil* MRI, optimisasi sekuens pemindaian, dan pengurangan waktu akuisisi data membantu meminimalkan sumber *noise* sejak awal. Dari sisi komputasi, algoritma *denoising* adaptif, *non-local means filtering*, dan *sparse representation* digunakan untuk memproses data pascaakuisisi. Kolaborasi erat antara insinyur medis, ilmuwan komputer, dan tenaga medis menjadi kunci keberhasilan penerapan strategi ini di lingkungan klinis.

Pentingnya pengurangan *noise* pada citra MRI otak juga didorong oleh perkembangan teknik diagnosis presisi dan kedokteran personal. Di era ini, setiap detail citra menjadi sangat berharga karena dapat memandu pengambilan keputusan yang disesuaikan dengan kondisi unik setiap pasien. Oleh karena itu, keberhasilan mengatasi *noise* bukan hanya berdampak pada kualitas citra, tetapi juga pada peningkatan keselamatan pasien, efisiensi biaya, dan efektivitas terapi. Tantangan ini akan terus relevan di masa depan, terutama

dengan meningkatnya permintaan untuk pencitraan otak yang cepat, aman, dan akurat.

Dengan demikian, meskipun *noise* pada citra MRI otak merupakan tantangan teknis yang signifikan, kemajuan teknologi pencitraan dan pemrosesan citra digital, didukung oleh kecerdasan buatan, memberikan peluang besar untuk mengatasinya. Perbaikan berkelanjutan di bidang ini akan memastikan bahwa citra MRI otak dapat memberikan informasi yang tajam, akurat, dan andal bagi diagnosis penyakit neurologis, sehingga berkontribusi langsung pada peningkatan kualitas layanan kesehatan.

➢ Motivasi Penggunaan *Convolutional Autoencoder* (CAE)

Pembelajaran mesin pada teknik modern mengacu pada kemampuan sistem untuk belajar dari data pelatihan tertentu untuk mengotomatiskan pembuatan model analitik dan menyelesaikan tugas-tugas terkait. Penambahan *noise* pada dataset juga digunakan untuk meningkatkan kemampuan ekstraksi fitur model. Dengan melatih data yang memiliki *noise*, model dapat belajar untuk mengekstrak fitur-fitur penting yang tidak berubah terhadap *noise*, sehingga meningkatkan kinerja model. Jenis *noise* yang mungkin terjadi pada gambar adalah *Gaussian Noise*, *Salt* & *Pepper Noise*, *Poisson Noise*, *Speckle Noise* [3]. Penggunaan *Convolutional Neural Network* berupa *RicianNets* pada 2020 oleh menunjukkan hasil yang lebih baik dibandingkan dengan metode *denoising* konvensional [4]. Selain CNN arsitektur pembelajaran mesin yang populer digunakan untuk perbaikan citra dari *noise* adalah *Convolutional Autoencoder*.

Autoencoder memerlukan gambar yang bersih dan gambar yang memiliki *noise* [4]. Pada pembahasan yang menggunakan model *Convolutional Autoencoder* untuk

menghilangkan *noise* pada *X-Ray* dan *CT*. Berbagai jenis *noise* diselidiki, termasuk *Gaussian*, *Salt* and *Pepper*, dan *Speckle Noise* dengan varian yang berbeda. CAE berfungsi untuk mengurangi *noise* dan mencapai hasil yang baik dalam mengekstraksi fitur yang berbeda dari gambar medis. Hal ini terlihat pada nilai yang diperoleh dengan rata-rata nilai *Peak Rasio Signal-to-Noise* (PSNR) 31.66098 untuk Citra *X-Ray* dan 30.4040 untuk citra *CT Scan*. Diperoleh juga rata-rata nilai Indeks Kesamaan Struktural (SSIM) 0.8715 untuk Citra *X-Ray* dan 0.8698 untuk Citra *CT-Scan* [5].

Pada pembahasan tahun 2021 menggunakan metode menghilangkan *noise* pada citra MRI dengan menggabungkan teknik *advanced Non-Local Means* (NLM) yang dimodifikasi dan *Non-Subsampled Shearlet Transform* (NSST). Pembahasan ini meneliti dua jenis *noise* yaitu *Rician Noise* dan *Gaussian Noise*. Hasil eksperimen menunjukkan bahwa metode terbukti efektif dalam meningkatkan kualitas citra MRI dan menjaga informasi diagnostik penting. dengan peningkatan PSNR hingga 13.98% dan SSIM hingga 16.30% [6]. Berdasarkan pembahasan dan metode yang sudah dijelaskan sebelumnya, pembahasan ini akan membahas penggunaan aristektur *machine learning* tertentu yang memang dikhususkan untuk melakukan tugas secara spesifik salah satunya merupakan *Convolutional Autoencoder* (CAE) yang dapat digunakan untuk meng-hilangkan *noise* dari data *input* dengan melatih model untuk merekonstruksi data asli dari *input* yang sudah terkontaminasi *noise*. CAE digunakan karena kinerjanya lebih baik daripada metode *filter* linear dan diharapkan dapat memperoleh hasil yang lebih baik.

Pemrosesan citra medis, khususnya pada MRI otak, memegang peranan penting dalam dunia kesehatan modern. Citra MRI yang jernih dan minim *noise* menjadi kunci bagi

radiolog dan dokter untuk membuat diagnosis yang tepat. Namun, dalam praktiknya, citra MRI sering mengalami degradasi kualitas akibat keterbatasan waktu pemindaian, gerakan pasien, dan gangguan sinyal. Melalui buku ini, pembaca akan diperkenalkan pada konsep *Convolutional Autoencoder*, sebuah arsitektur *deep learning* yang dirancang untuk merepresentasikan data dalam bentuk yang lebih ringkas sambil mempertahankan informasi penting.

CAE bekerja melalui dua tahap utama: *encoder*, yang memampatkan citra menjadi representasi laten berukuran kecil, dan *decoder*, yang merekonstruksi citra dari representasi laten tersebut dengan kualitas yang lebih baik. Proses ini memungkinkan penghapusan *noise* sambil menjaga detail penting pada struktur otak. Untuk menilai efektivitas CAE dalam mereduksi *noise*, digunakan dua metrik utama:

***Mean Squared Error* (MSE)**

Di mana *I* adalah citra asli (*ground truth*), KKK adalah citra hasil rekonstruksi, dan m,nm, nm,n adalah ukuran citra. MSE yang lebih rendah menunjukkan kesalahan rata-rata yang lebih kecil antara citra asli dan hasil rekonstruksi.

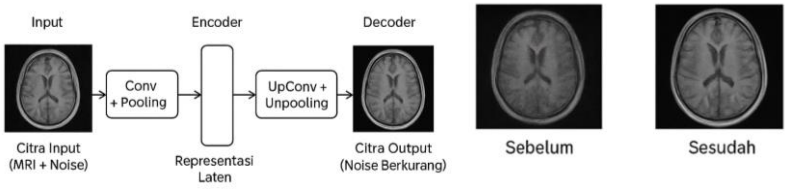
$$\mathrm{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

***Peak Signal-to-Noise Ratio* (PSNR)**

Di mana MAX_I adalah nilai piksel maksimum (misalnya 255 untuk citra 8-bit). PSNR yang lebih tinggi menunjukkan kualitas citra yang lebih baik.

$$\mathrm{PSNR} = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{\mathrm{MSE}} \right)$$

Interpretasi kedua metrik ini membantu pembaca memahami bagaimana performa CAE diukur secara objektif. Gambar 3 adalah ilustrasi konsep CAE dan Diagram Arsitektur CAE.



Arsitektur *Convolutional Autoencoder* | Perbandingan Citra MRI Sebelum dan Sesudah Pengurangan *Noise*

Gambar 3. Konsep CAE dan Diagram Arsitektur CAE.

Gambar 3 mengilustrasikan konsep dan arsitektur *Convolutional Autoencoder* (CAE) yang digunakan dalam proses pengurangan *noise* pada citra MRI otak. CAE terdiri dari dua komponen utama, yaitu *encoder* dan *decoder*. *Encoder* bertugas mengompres citra *input* yang mengandung *noise* melalui operasi konvolusi dan pooling sehingga menghasilkan representasi laten berupa fitur-fitur esensial dari citra tersebut. Representasi laten ini bersifat lebih ringkas namun tetap mempertahankan informasi struktural yang penting dari citra aslinya.

Selanjutnya, *decoder* menerima representasi laten tersebut dan melakukan rekonstruksi citra menggunakan operasi UpConv dan Unpooling, sehingga menghasilkan citra *output* yang telah tereduksi *noise*nya. Proses ini terlihat jelas pada bagian (b) gambar, yang menampilkan perbandingan citra MRI sebelum dan sesudah pengurangan *noise*. Citra sebelum pemrosesan tampak lebih kasar akibat gangguan *noise*, sedangkan citra sesudah pemrosesan menunjukkan struktur anatomi otak yang lebih bersih dan tajam. Pendekatan ini menjadikan CAE efektif sebagai metode preprocessing dalam analisis citra medis, khususnya untuk meningkatkan kualitas citra sebelum tahap segmentasi atau klasifikasi lebih lanjut.

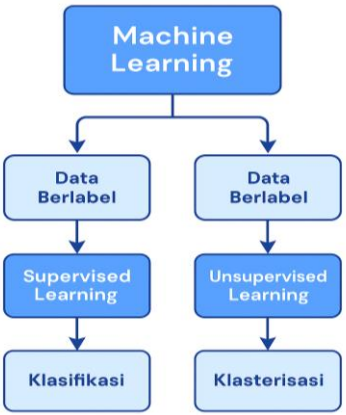
BAB II
DASAR TEORI DAN KONSEP KUNCI

➢ Pengantar *Machine Learning* dan *Deep Learning*

Machine learning dapat diartikan sebagai cabang dari ilmu kecerdasan buatan yang memiliki tujuan untuk memrogram komputer menyelesaikan masalah menggunakan data latih yang diberi dan mengevaluasi hasil belajar menggunakan data uji tanpa diprogram secara terperinci [1]. Berdasarkan teknik belajar, *machine learning* dapat dibagi menjadi dua tipe yaitu *supervised* dan *unsupervised*. *Supervised* dalam pembelajaran yang diawasi, model dilatih pada kumpulan data berlabel, yang berarti bahwa data *input* dipasangkan dengan *output* yang benar. Tujuannya adalah untuk mempelajari pemetaan dari *input* ke *output*, memungkinkan model untuk membuat prediksi pada data baru yang tidak terlihat. *Unsupervised learning* melibatkan model pelatihan pada data tanpa tanggapan berlabel. Tujuannya adalah untuk mengidentifikasi pola atau pengelompokan dalam data [6].

Denoising citra yang menggunakan pasangan data citra *noise* dan citra bersih merupakan bagian dari pembelajaran terawasi (*supervised learning*) dengan pendekatan regresi. Hal ini disebabkan karena model bertugas memprediksi nilai intensitas piksel bersih dari *input Noise*, bukan meng-klasifikasikan ke dalam kategori tertentu. Dengan demikian, *denoising* dalam konteks ini dapat dikategorikan sebagai tugas regresi citra.

Pada Gambar 4 dapat dilihat metode *supervised learning* melibatkan pelatihan model dengan menggunakan *input* berupa kumpulan data berlabel. *Output* dari supervised learning secara umum dibedakan menjadi dua jenis kategori utama, yaitu klasifikasi untuk menentukan label kelas diskret dan regresi yang berfungsi memprediksi nilai numerik kontinu secara akurat sesuai pola data. Sedangkan *unsupervised learning* adalah jenis pembelajaran mesin yang berhubungan pelatihan model dengan menggunakan *input* berupa kumpulan data tanpa *output* berlabel, *output* dari *supervised learning* secara umum berupa *clustering* atau pengelompokan [7].



Gambar 4. *Machine learning*

Setelah memahami pembagian utama *machine learning* ke dalam *supervised* dan *unsupervised learning*, perlu dipahami bahwa keberhasilan kedua pendekatan ini sangat dipengaruhi oleh ketersediaan data, kualitas *preprocessing*, serta pemilihan algoritma yang sesuai. Dalam *supervised learning*, data berlabel menjadi elemen penting. Namun, proses pelabelan data medis, seperti citra MRI otak, bukanlah tugas yang sederhana. Pelabelan memerlukan keahlian khusus dari radiolog atau dokter ahli, dan seringkali membutuhkan waktu yang lama

serta biaya yang tinggi. Oleh karena itu, salah satu tantangan besar dalam penerapan *machine learning* pada bidang medis adalah keterbatasan data berlabel. Hal ini kemudian mendorong lahirnya pendekatan semi-*supervised learning* dan *transfer learning*, di mana model dapat memanfaatkan data tanpa label atau pengetahuan yang sudah dipelajari dari domain lain untuk meningkatkan performa.

Unsupervised learning, meskipun tidak memerlukan label, menghadapi tantangan dalam hal interpretabilitas hasil. Misalnya, ketika sebuah algoritma klasterisasi menemukan pola baru dalam data MRI otak, interpretasi pola tersebut tetap memerlukan validasi klinis agar dapat dipastikan relevansinya terhadap kondisi medis tertentu. Oleh sebab itu, integrasi antara domain pengetahuan medis dengan kemampuan teknis *machine learning* sangat penting. Tanpa pemahaman medis yang mendalam, hasil dari *unsupervised learning* bisa saja tidak berguna atau bahkan menyesatkan.

Perkembangan *deep learning* membawa revolusi besar dalam dunia *machine learning*. Jika pada metode tradisional dibutuhkan *feature engineering* yang rumit, maka deep learning mampu mengekstraksi fitur secara otomatis dari data mentah. Arsitektur seperti *Convolutional Neural Network* (CNN) terbukti sangat efektif untuk pemrosesan citra, termasuk citra medis. CNN bekerja dengan menerapkan filter konvolusi pada citra, yang mampu mendeteksi pola sederhana seperti tepi dan sudut pada lapisan awal, hingga pola kompleks seperti struktur tumor pada lapisan yang lebih dalam. Secara matematis, operasi konvolusi dalam CNN dapat dijelaskan dengan persamaan:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) \cdot K(i-m, j-n)$$

Dimana *I* adalah citra *input*, *K* adalah *kernel* atau filter, dan S adalah hasil konvolusi. Proses ini memungkinkan CNN mengekstraksi fitur spasial dari citra MRI otak dengan sangat efisien.

Selain CNN, *Autoencoder* juga menjadi arsitektur penting, terutama untuk tugas *denoising*. *Autoencoder* terdiri dari dua bagian utama, yaitu *encoder* dan *decoder*. *Encoder* bertugas memampatkan data *input* ke dalam representasi laten berdimensi lebih rendah, sedangkan *decoder* bertugas merekonstruksi kembali data dari representasi laten tersebut. Untuk tugas *denoising*, *input* berupa citra ber*noise*, dan target *output* berupa citra bersih. Dengan cara ini, *Autoencoder* belajar membuang *noise* dan mempertahankan detail penting dari citra. Fungsi *Loss* yang umum digunakan dalam *denoising* adalah *Mean Squared Error* (MSE), yang didefinisikan sebagai:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Dimana y_i adalah nilai piksel asli, y_i adalah nilai piksel hasil prediksi, dan *N* adalah jumlah piksel.

Selain MSE, metrik lain yang sering digunakan adalah *Peak Signal-to-Noise Ratio* (PSNR), yang mengukur perbandingan antara daya maksimum sinyal dan daya *noise*. Rumusnya adalah:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

Dimana MAX_i adalah nilai intensitas maksimum piksel (misalnya 255 untuk citra 8-bit). PSNR yang lebih tinggi menunjukkan kualitas rekonstruksi citra yang lebih baik.

Dalam implementasi nyata, CNN dan CAE telah menunjukkan performa yang sangat baik untuk *denoising* citra MRI otak. Sebagai ilustrasi, perbandingan antara citra MRI dengan *Gaussian noise*, citra hasil *denoising* menggunakan filter

tradisional (misalnya median filter), dan citra hasil *denoising* menggunakan CAE akan menunjukkan perbedaan signifikan. Filter tradisional sering menghapus *noise* tetapi sekaligus mengaburkan detail penting, sedangkan CAE mampu mengurangi *noise* sekaligus mempertahankan struktur halus pada jaringan otak.

Selain aspek teknis, perlu juga dipahami bagaimana *machine learning* dan deep learning diintegrasikan dalam workflow klinis. Misalnya, radiolog dapat menggunakan sistem berbasis AI sebagai alat bantu diagnosis. Ketika citra MRI otak pasien diunggah ke sistem, model deep learning yang telah dilatih akan memberikan prediksi mengenai keberadaan tumor, ukuran, serta jenis jaringan yang mencurigakan. Radiolog kemudian memverifikasi hasil ini sebelum mengambil keputusan. Integrasi seperti ini tidak dimaksudkan untuk menggantikan dokter, melainkan untuk meningkatkan efisiensi dan akurasi diagnosis.

Di sisi lain, penggunaan *machine learning* dalam bidang medis juga menimbulkan tantangan etika dan regulasi. Pertanyaan seperti "siapa yang bertanggung jawab jika model AI salah mendiagnosis pasien?" atau "bagaimana cara memastikan bahwa data pasien tetap aman dan terlindungi?" menjadi isu penting yang harus dijawab. Oleh karena itu, pengembangan sistem berbasis *machine learning* di bidang medis harus memperhatikan aspek keamanan data, transparansi algoritma, serta validasi klinis yang ketat.

Sejalan dengan itu, interpretabilitas model juga menjadi perhatian utama. Model deep learning sering disebut sebagai "*black box*" karena sulit dipahami bagaimana model mengambil keputusan. Dalam bidang medis, interpretabilitas sangat penting, karena dokter harus dapat menjelaskan kepada pasien dasar dari setiap keputusan medis. Oleh sebab itu,

berkembanglah bidang *Explainable* AI (XAI), yang bertujuan untuk menjelaskan cara kerja model *machine learning* secara transparan.

Jika kembali pada diagram sederhana di Gambar 4, terlihat bahwa meskipun pembagian *machine learning* hanya ke dalam *supervised* dan *unsupervised*, perkembangan nyata di lapangan jauh lebih kompleks. Ada pula *reinforcement learning*, *semi-supervised learning*, hingga *self-supervised learning*, yang masing-masing memiliki peran khusus. Namun, kerangka dasar *supervised* dan *unsupervised* tetap relevan untuk menjelaskan fondasi awal *machine learning*.

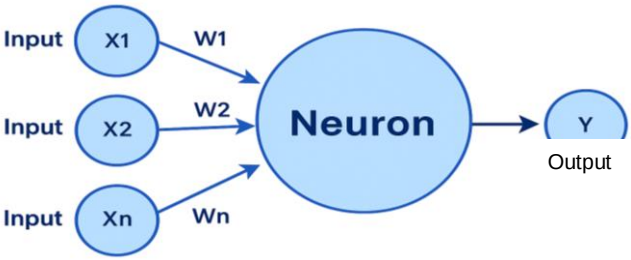
Dalam praktiknya, *machine learning* tidak hanya terbatas pada klasifikasi, regresi, atau klasterisasi. Ada berbagai variasi teknik dan pendekatan yang terus dikembangkan. Misalnya, dalam bidang medis, digunakan metode segmentasi semantik dengan *U-Net*, yang merupakan varian CNN khusus untuk citra medis. *U-Net* memiliki arsitektur *encoder-decoder* dengan skip *connections*, yang memungkinkan model mempertahankan detail spasial penting saat merekonstruksi citra.

U-Net dan variannya sangat efektif untuk segmentasi tumor otak, dengan akurasi yang jauh lebih tinggi dibandingkan metode tradisional. Hal ini memperlihatkan bagaimana *machine learning* dan deep learning tidak hanya menjadi teori, tetapi benar-benar memberikan dampak nyata pada perawatan kesehatan.

➢ Jaringan Saraf Tiruan (*Artificial Neural Network*)

Jaringan Saraf Tiruan (JST) adalah salah satu terobosan besar dalam bidang kecerdasan buatan yang terinspirasi dari sistem saraf biologis manusia. Konsep ini memungkinkan mesin untuk meniru sebagian kecil dari cara otak manusia belajar, mengenali pola, serta membuat keputusan berdasarkan data.

JST merupakan model komputasi yang memiliki tugas mengolah informasi menggunakan kumpulan *neuron* yang saling terhubung terinspirasi oleh jaringan saraf biologis yang ditemukan di otak manusia. JST terdiri dari *node* yang saling berhubungan, sering disebut sebagai *neuron*, yang diatur dalam tiap lapisan. Setiap *node* berfungsi mirip dengan *neuron* biologis, memproses *input* dan meneruskan *output* ke *node* lain melalui koneksi, mirip dengan sinapsis di otak. Koneksi antar *node* ditimbang, yang berarti bahwa beberapa koneksi memiliki pengaruh lebih besar pada *output* daripada yang lain. Dalam praktiknya, ini berarti bahwa selama pelatihan, bobot koneksi disesuaikan berdasarkan kinerja jaringan dalam mencapai hasil yang diinginkan [7]. JST bekerja dengan memanfaatkan neuron buatan yang saling terhubung dalam bentuk lapisan, di mana setiap neuron menerima masukan (*input*), memprosesnya dengan bobot tertentu, dan kemudian menghasilkan keluaran (*output*). Bobot tersebut akan diperbarui terus-menerus melalui proses pelatihan, sehingga jaringan dapat meningkatkan kemampuannya dalam memprediksi atau mengklasifikasikan suatu informasi. Dengan kata lain, semakin banyak data yang dipelajari, semakin baik pula kemampuan JST untuk melakukan generalisasi terhadap kasus baru, seperti terlihat pada Gambar 5.



Gambar 5. Jaringan Saraf Tiruan

Pada Gambar 5 menunjukkan struktur dasar dari sebuah *neuron* dalam JST.

a. *Input* adalah fitur-fitur atau variabel *input* yang masuk ke dalam *neuron*. Setiap fitur x mewakili suatu nilai dari data *input*.
b. w adalah bobot yang mengatur seberapa besar pengaruh suatu *input* terhadap neuron. Setiap *input* dikalikan dengan bobot yang relevan. Jika suatu *input* memiliki bobot yang lebih besar, itu berarti pengaruh *input* tersebut terhadap keputusan neuron lebih signifikan.
c. *Neuron* melakukan perhitungan matematis dengan menjumlahkan hasil perkalian antara *input* dengan bobotnya. Hasil penjumlahan ini kemudian diproses oleh fungsi aktivasi untuk menghasilkan *output*.
d. *Output* adalah hasil dari perhitungan dalam *neuron* setelah melewati fungsi aktivasi. *Output* digunakan untuk prediksi atau klasifikasi.

Sebagaimana ditunjukkan pada Gambar 5, struktur dasar neuron buatan dapat dilihat sebagai unit sederhana yang menerima sejumlah masukan $_1, x_2, \ldots, x_n$, masing-masing dikalikan dengan bobot w_1, w_2, ..., w_n. Hasil perkalian ini kemudian dijumlahkan bersama dengan sebuah bias (b) untuk menghasilkan sinyal total z, yang dirumuskan sebagai:

$$z = \sum_{i=1}^{n} (w_i \cdot x_i) + b$$

Nilai z ini kemudian dilewatkan ke sebuah fungsi aktivasi untuk menghasilkan keluaran akhir y. Fungsi aktivasi sangat penting karena menentukan bagaimana sinyal diteruskan, apakah dalam bentuk linier atau non-linier, sehingga jaringan dapat menangani pola-pola kompleks. Tanpa

fungsi aktivasi non-linier, jaringan saraf hanya akan menjadi model linier biasa yang terbatas kemampuannya.

Beberapa fungsi aktivasi populer antara lain sigmoid, tanh, ReLU (*Rectified Linear Unit*), dan *Softmax*. Fungsi sigmoid memetakan nilai masukan ke rentang antara 0 dan 1, sehingga sering digunakan dalam kasus probabilistik. Fungsi tanh memiliki karakteristik serupa tetapi memetakan nilai ke rentang -1 hingga 1, memberikan keseimbangan yang lebih baik dalam pembelajaran. Sementara itu, ReLU memberikan keluaran 0 jika masukan negatif dan nilai asli jika masukan positif, sehingga lebih efisien digunakan dalam jaringan dalam (*deep networks*). Untuk kasus klasifikasi multikelas, fungsi *Softmax* digunakan karena mampu mengubah keluaran menjadi distribusi probabilitas. Grafik pada ilustrasi di atas menunjukkan bagaimana bentuk kurva dari masing-masing fungsi aktivasi, sehingga memberikan gambaran visual tentang bagaimana *input* diubah menjadi *output*.

Dalam implementasinya, JST biasanya tersusun dari tiga jenis lapisan: lapisan *input*, lapisan tersembunyi (*hidden layer*), dan lapisan *output*. Lapisan *input* menerima fitur data mentah, lapisan tersembunyi memproses informasi melalui kombinasi linear dan fungsi aktivasi, sedangkan lapisan *output* menghasilkan hasil akhir prediksi atau klasifikasi. Misalnya, pada kasus citra medis MRI, lapisan *input* dapat berupa piksel-piksel citra, lapisan tersembunyi bertugas mengekstraksi pola dari struktur otak, dan lapisan *output* memberikan hasil apakah citra tersebut menunjukkan tumor atau kondisi normal.

Keunggulan JST adalah kemampuannya untuk melakukan pembelajaran representasi (*representation learning*). Artinya, jaringan dapat secara otomatis mengekstraksi fitur penting dari data tanpa harus dirancang secara manual oleh manusia. Sebagai contoh, dalam pengolahan citra medis, JST

dapat mengenali tepi, bentuk, hingga pola kompleks dari jaringan otak untuk mendeteksi adanya kelainan. Hal ini berbeda dengan metode klasik yang mengandalkan perhitungan matematis eksplisit yang dirancang khusus oleh pakar.

Namun, JST juga memiliki tantangan, seperti kebutuhan akan jumlah data yang besar agar dapat berfungsi optimal, serta risiko *overfitting* ketika jaringan terlalu menyesuaikan diri pada data latih dan gagal melakukan generalisasi. Untuk mengatasi hal ini, berbagai teknik regulasi seperti *dropout*, *batch normalization*, dan *early stopping* sering digunakan. Selain itu, optimisasi bobot dilakukan dengan algoritma *backpropagation* yang berbasis pada turunan gradien, memastikan bahwa setiap lapisan belajar secara bertahap menuju hasil yang lebih baik.

Dengan adanya JST, bidang medis, khususnya dalam pemrosesan citra MRI otak, mendapatkan manfaat besar. JST memungkinkan perbaikan kualitas citra, reduksi *noise*, segmentasi jaringan otak, hingga deteksi otomatis adanya kelainan seperti tumor. Hal ini menjadikan JST sebagai salah satu teknologi kunci yang mendukung perkembangan kecerdasan buatan dalam dunia kesehatan modern.

Dalam konteks pemrosesan citra medis, khususnya MRI otak, Jaringan Saraf Tiruan dapat diimplementasikan dalam berbagai bentuk arsitektur. Dua pendekatan dasar yang sering digunakan adalah *Multilayer Perceptron (MLP)* dan *Convolutional Neural Network (CNN)*.

Multilayer Perceptron (MLP) merupakan arsitektur JST klasik yang terdiri dari lapisan *input*, beberapa hidden *layer*, dan lapisan *output*. Pada kasus MRI otak, setiap piksel citra dapat dipandang sebagai *input* numerik yang masuk ke neuron pada lapisan *input*. Meskipun MLP dapat digunakan, pendekatan ini memiliki kelemahan karena citra medis

biasanya berukuran besar, sehingga jumlah *input* yang sangat banyak akan membuat jaringan menjadi kompleks dan sulit dilatih. Namun, MLP tetap bermanfaat untuk menjelaskan konsep dasar JST pada data citra.

Sebagai contoh, jika memiliki citra MRI otak dengan ukuran 28 × 28 piksel, maka lapisan *input* MLP akan memiliki 784 neuron. Setiap neuron ini terhubung dengan sejumlah neuron pada hidden *layer*, yang kemudian diproses melalui fungsi aktivasi, hingga akhirnya menghasilkan *output* berupa klasifikasi, misalnya "normal" atau "tumor". Persamaan dasar operasi pada setiap neuron tetap sama, yaitu:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Di sisi lain, *Convolutional Neural Network (CNN)* merupakan arsitektur JST yang lebih efisien dan efektif dalam menangani data citra medis. CNN bekerja dengan menggunakan lapisan konvolusi yang menerapkan filter (*kernel*) pada citra untuk mengekstraksi fitur penting, seperti tepi, bentuk, atau pola tekstur yang relevan dengan identifikasi tumor. CNN juga dilengkapi dengan lapisan pooling untuk mengurangi dimensi citra sekaligus mempertahankan fitur penting, sehingga mempercepat proses komputasi.

Dalam alur CNN sederhana untuk klasifikasi tumor otak pada MRI, tahapannya dapat dijelaskan sebagai berikut:

a. *Input Layer*: Citra MRI otak dimasukkan sebagai *input* (misalnya ukuran 128x128 piksel).
b. *Convolutional Layer*: Filter berukuran 3x3 atau 5x5 digeser melintasi citra untuk menghasilkan *feature maps*.
c. *Activation Layer* (ReLU): Hasil konvolusi dilewatkan ke fungsi aktivasi ReLU untuk memperkenalkan non-linearitas.

d. *Pooling Layer*: Operasi *max pooling* diterapkan untuk mereduksi dimensi citra sambil mempertahankan fitur utama.
e. *Fully Connected Layer*: Hasil ekstraksi fitur diratakan (*flattening*) dan dihubungkan ke lapisan dense (*fully connected*) untuk klasifikasi.
f. *Output Layer*: Menghasilkan prediksi akhir, misalnya tumor jinak, tumor ganas, atau citra normal.

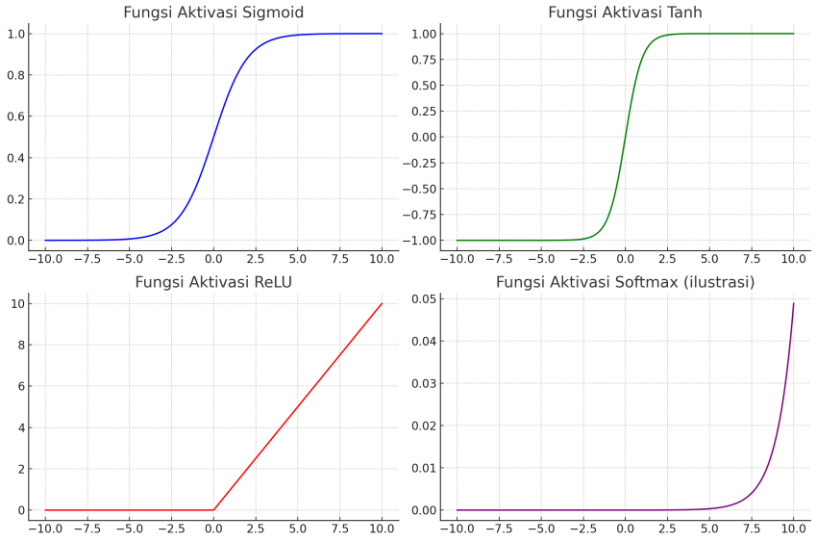
Alur CNN ini sering divisualisasikan dalam bentuk diagram bertingkat, di mana citra masukan secara bertahap diproses menjadi representasi fitur yang lebih abstrak hingga menghasilkan keputusan klasifikasi.

Dengan menggunakan CNN, model dapat mengenali pola kompleks dalam MRI otak yang mungkin sulit dilihat oleh mata manusia. Sebagai contoh, *noise* yang sering muncul pada citra MRI dapat difilter secara otomatis, dan jaringan dapat fokus pada area yang mengandung indikasi keberadaan tumor. Hal ini memberikan nilai tambah yang sangat signifikan dalam membantu radiolog mengambil keputusan.

Selain klasifikasi, JST berbasis CNN juga dapat digunakan untuk tugas segmentasi, yaitu membagi citra MRI ke dalam area-area spesifik, misalnya membedakan antara jaringan otak sehat, tumor, dan area lain yang terdampak. Model yang umum digunakan untuk segmentasi adalah *U-Net*, yang juga merupakan turunan dari arsitektur CNN.

Penerapan JST untuk MRI otak ini dapat ditingkatkan lebih lanjut dengan mengombinasikan teknik *denoising Autoencoder*. Dalam hal ini, citra MRI yang penuh dengan *noise* dapat dilewatkan ke jaringan *Autoencoder* untuk dipulihkan menjadi citra yang lebih bersih sebelum masuk ke tahap klasifikasi. Hal ini memastikan bahwa kualitas citra *input* yang

dianalisis oleh CNN jauh lebih baik, sehingga hasil diagnosis pun lebih akurat, seperti pada contoh Gambar 6.



Gambar 6. Empat Fungsi Utama Jaringan Saraf Tiruan

Gambar 6 menampilkan empat fungsi aktivasi utama dalam jaringan saraf tiruan: Sigmoid, Tanh, ReLU, dan *Softmax*. Fungsi aktivasi berfungsi untuk mengubah *input* linear dari neuron menjadi *output* non-linear, sehingga jaringan mampu mempelajari pola kompleks. Fungsi aktivasi pertama, Sigmoid, memiliki bentuk kurva berbentuk S yang mengubah *input* menjadi nilai antara 0 hingga 1. Secara matematis, fungsi ini didefinisikan sebagai:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Fungsi sigmoid sangat berguna dalam kasus klasifikasi biner karena dapat menginterpretasikan *output* sebagai probabilitas. Misalnya, dalam citra MRI otak, fungsi sigmoid

bisa digunakan untuk memutuskan apakah terdapat tumor (*output* mendekati 1) atau tidak (*output* mendekati 0). Namun, kelemahannya adalah rentan terhadap masalah *vanishing gradient*, yang membuat pelatihan jaringan menjadi lambat. Fungsi kedua adalah Tanh (*Hyperbolic Tangent*), yang mirip dengan sigmoid tetapi nilai *output* berada pada rentang -1 hingga 1. Rumus matematisnya:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Kelebihan tanh adalah *output*-nya terpusat pada nol, sehingga sering membuat pelatihan lebih stabil dibanding sigmoid. Dalam pemrosesan citra medis, tanh dapat membantu memetakan intensitas piksel yang memiliki nilai positif maupun negatif, sehingga distribusi data menjadi lebih seimbang.

Fungsi ketiga adalah ReLU (*Rectified Linear Unit*), yang didefinisikan sebagai:

$$f(x) = \max(0, x)$$

Artinya, semua nilai *input* negatif akan diubah menjadi nol, sedangkan nilai *input* positif tetap dipertahankan. Fungsi ini sangat sederhana namun efektif, sehingga banyak digunakan pada lapisan tersembunyi (*hidden layer*) jaringan modern. Keunggulannya adalah mempercepat pelatihan dan mengurangi masalah *vanishing gradient*. Dalam konteks MRI otak, ReLU dapat membantu jaringan fokus pada fitur-fitur penting seperti tepi tumor atau perbedaan tekstur jaringan.

Fungsi terakhir adalah *Softmax*, yang digunakan untuk klasifikasi multi-kelas. Fungsi ini mengubah *output* dari neuron menjadi distribusi probabilitas. Rumus matematisnya:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad i = 1, \ldots, K$$

Dengan ziz_izi adalah *output* linear dari neuron ke-iii dan KKK adalah jumlah kelas. *Output Softmax* selalu bernilai antara 0 hingga 1, dan totalnya berjumlah 1. Dalam aplikasi medis, *Softmax* sering dipakai untuk mengklasifikasikan jenis tumor otak, misalnya membedakan tumor glioma, meningioma, atau pituitari berdasarkan citra MRI.

Dengan demikian, fungsi aktivasi bukan hanya elemen teknis, tetapi merupakan komponen inti yang membuat jaringan saraf tiruan mampu bekerja pada masalah nyata. Dalam konteks citra medis, pemilihan fungsi aktivasi yang tepat dapat sangat memengaruhi akurasi diagnosis dan kecepatan model dalam mempelajari pola pada data.

➢ *Convolutional Neural Network* (CNN)

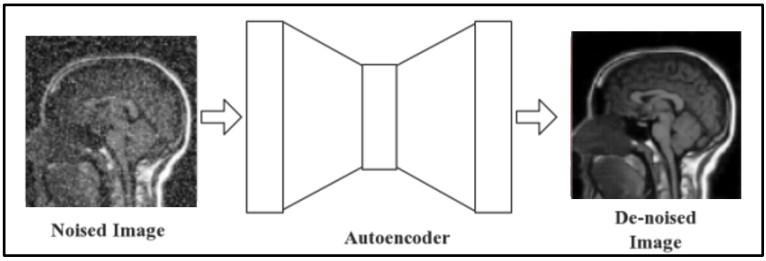
Convolutional Neural Networks (CNN) merupakan salah satu teknik *machine learning* yang paling populer, terutama untuk aplikasi yang melibatkan data visual, seperti klasifikasi gambar, deteksi objek, dan segmentasi. CNN dirancang untuk belajar representasi dari data yang berbentuk *grid*, yang merupakan struktur umum untuk gambar, di mana piksel diatur dalam grid. Kemampuan ini memungkinkan CNN untuk secara otomatis mengekstrak fitur-fitur relevan dari data *input* tanpa memerlukan rekayasa fitur secara manual [8].

CNN disusun dalam serangkaian lapisan, masing-masing dengan pekerjaan tertentu. Pada *layer* awal terdapat *input layer* dimana data *input* mentah dimasukkan ke dalam jaringan. Lapisan *input* tidak melakukan perhitungan apa pun tetapi mempersiapkan data untuk diproses oleh lapisan berikutnya. Pada lapisan berikutnuya terdapat lapisan konvolusional,

lapisan ini bertanggung jawab untuk menemukan fitur dalam gambar [9]. Misalnya, lapisan pertama mungkin mencari hal-hal sederhana seperti tepi atau warna, sementara lapisan yang lebih dalam dapat mengenali bentuk yang lebih kompleks. Setelah itu, lapisan *downsampling* mengecilkan ukuran data untuk menyoroti informasi penting, sementara lapisan aktivasi membantu jaringan belajar pola yang lebih rumit [10]. CNN belajar menggunakan metode *backpropagation*, yang menyesuaikan bobot dan bias kernel untuk meningkatkan kemampuan pengenalan fitur. Lapisan pooling digunakan untuk mengurangi data, menjaga informasi yang relevan dan mencegah *overfitting*. Fungsi aktivasi, seperti ReLU, menambahkan non-linearitas ke model untuk memungkinkan pengenalan pola yang lebih kompleks, dan fungsi *Softmax* mengubah skor mentah menjadi probabilitas untuk menentukan kelas objek dalam gambar [11].

➢ *Autoencoder* dan *Convolutional Autoencoder*

Autoencoder adalah jenis jaringan saraf buatan yang digunakan untuk pembelajaran tanpa pengawasan, terutama dalam konteks *denoising* gambar. *Autoencoder* terdiri dari dua bagian utama yaitu *Encoder* dan *Decoder*. *Encoder* mengambil *input* yang telah diberi *noise* dan mengubahnya menjadi representasi yang lebih kompak dalam bentuk kode. Di tengah-tengah jaringan terdapat *Bottleneck*, yaitu titik kompresi di mana data direduksi menjadi bentuk paling padat yang disebut representasi latent, yang diharapkan mengandung fitur utama dari data asli. *Decoder* kemudian mengambil kode yang dihasilkan oleh *encoder* dan mencoba merekonstruksi *input* asli dari representasi latent ini, menggunakan beberapa lapisan neuron yang secara bertahap meningkatkan dimensi data hingga kembali mendekati bentuk aslinya [12].



Gambar 7. *Autoencoder*

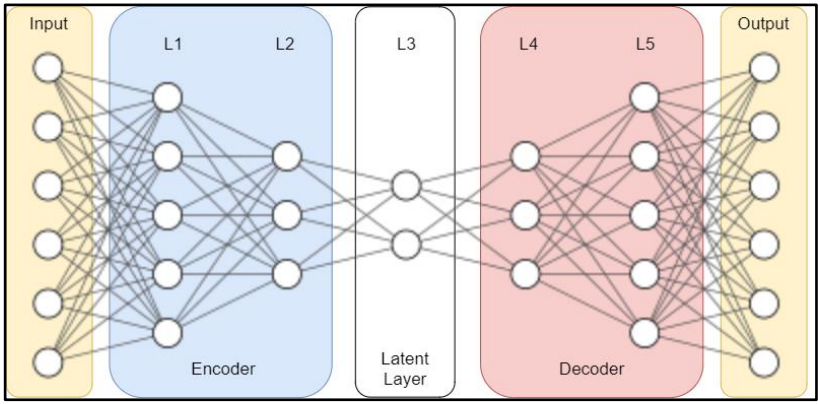
Pada Gambar 7 menggambarkan proses kerja dari sebuah *Denoising Autoencoder*. Di sisi kiri, terdapat gambar yang terpengaruh oleh *noise*, yang disebut sebagai *Noise*d *Image*. Gambar ini kemudian dimasukkan ke dalam *Autoencoder*, yang merupakan jaringan saraf dengan arsitektur simetris yang terdiri dari *encoder* dan *decoder*. *Encoder* berfungsi untuk mengompresi informasi dari gambar yang mengandung *noise* ke dalam bentuk yang lebih sederhana di lapisan tersembunyi, sementara *decoder* bertugas untuk merekonstruksi gambar kembali dari representasi kompresi tersebut. Hasil dari proses ini adalah De*noise*d *Image* di sisi kanan, yang merupakan gambar versi bersih dengan *noise* yang diminimalkan

➢ Arsitektur CAE: *Encoder, Decoder*, dan Fungsi Aktivasi

Arsitektur CAE terdiri dari *encoder* dan *decoder*. *Encoder* mengompresi gambar *input* menjadi representasi dimensi rendah menggunakan lapisan konvolusi, yang efektif dalam menangkap hierarki spasial dalam gambar. *Decoder* kemudian merekonstruksi gambar dari representasi terkompresi ini, sering menggunakan lapisan konvolusi yang ditransposikan untuk mengambil sampel data kembali ke dimensi aslinya [12]. *Convolutional Autoencoder* merupakan gabungan dari prinsip kerja *Autoencoder* dan CNN. CAE dirancang untuk mempelajari pemetaan dari citra yang terdistorsi menjadi citra yang bersih

melalui proses pelatihan yang melibatkan *input* berupa citra yang telah ditambahkan *noise*. Arsitektur CAE biasanya mencakup lapisan seperti lapisan *input* gambar, lapisan konvolusi, normalisasi *Batch*, fungsi aktivasi (seperti *ReLU*). Proses pelatihan melibatkan pengenalan *noise* acak (misalnya, *Noise Gaussian*) ke gambar *input*.

Teknologi *Convolutional Autoencoder* atau CAE saat ini memiliki cakupan penerapan yang sangat luas dan beragam, terutama di berbagai bidang profesional yang memang sangat bergantung pada tingkat kualitas visual sebuah citra, contohnya seperti pada prosedur pencitraan medis. Dalam konteks medis tersebut, kemunculan gangguan atau *noise* pada hasil gambar tentu saja dapat mengaburkan berbagai detail kecil namun sangat penting yang memiliki sifat diagnostik bagi pasien. Oleh karena itu, kemampuan luar biasa dari sistem CAE ini dalam mempertahankan seluruh karakteristik visual penting secara akurat, sambil sekaligus secara signifikan mengurangi tingkat *noise* yang mengganggu, menjadi sebuah hal yang sangat krusial bagi tenaga medis modern. Kemampuan luar biasa ini menjadikan teknologi CAE sebagai salah satu pendekatan yang sangat efektif dalam upaya peningkatan kualitas visual citra medis, sekaligus memperkuat keandalan proses diagnosis yang berbasis citra tersebut.[13].



Gambar 8. *Convolutional Autoencoder*

Pada Gambar 8 menunjukkan arsitektur *Convolutional Denoising Autoencoder* (CAE) yang terdiri dari tiga bagian utama seperti *Encoder*, *Latent layer*, dan *Decoder*. Bagian *Encoder* (ditandai dengan warna biru) bertugas untuk mengompresi data *input* menjadi representasi yang lebih sederhana di lapisan laten (L3). Proses ini melibatkan beberapa lapisan, seperti konvolusi, normalisasi, dan *pooling*, untuk mengekstraksi fitur penting dari citra yang mengandung *noise*. *Latent layer* (warna putih) adalah representasi terkompresi yang menyimpan informasi esensial dari citra dalam bentuk yang lebih padat dan ringkas.

Selanjutnya, bagian *Decoder* (warna merah) berfungsi untuk merekonstruksi kembali citra dari representasi laten tersebut. *Decoder* menggunakan lapisan seperti *Conv2D Transpose* dan *UpSampling* untuk mengembalikan data ke dimensi asli dan menghilangkan *noise*, menghasilkan *output* rekonstruksi yang mendekati citra *input*. Arsitektur ini dirancang untuk membersihkan citra yang terdegradasi oleh *noise* dengan mempelajari pola kompresi dan rekonstruksi secara efektif

➢ Fungsi *Loss*: MSE dan PSNR

Fungsi *Loss* MSE

MSE mengukur rata-rata dari kuadrat perbedaan antara nilai prediksi dan nilai sebenarnya. MSE menekankan penalti lebih besar pada kesalahan yang lebih besar karena menggunakan kuadrat perbedaan. Oleh karena itu, kesalahan yang lebih besar memiliki dampak lebih besar pada nilai MSE [14].

$$MSE = \frac{1}{w \times h} \sum_{i-0}^{w-1} \sum_{j-0}^{h-1} (I(i,j) - K(i,j))^2$$

Persamaan 2.1 digunakan untuk mengukur rata-rata kuadrat selisih antara dua citra, yaitu citra asli dan citra hasil rekonstruksi atau citra yang mengalami degradasi. Perhitungan dilakukan dengan mengambil selisih antara setiap piksel dari kedua citra, kemudian mengkuadratkan hasil selisih tersebut untuk menghilangkan nilai negatif dan memperbesar dampak perbedaan yang lebih besar. Setelah itu, semua hasil kuadrat selisih dijumlahkan dan dibagi dengan jumlah total piksel dalam citra yang dihitung dengan mengkalikan tinggi dan lebar citra dalam satuan piksel untuk mendapatkan nilai rata-rata kesalahan kuadrat. Semakin kecil nilai MSE, semakin kecil perbedaan kuadrat rata-rata antara nilai piksel citra asli dan citra hasil rekonstruksi, yang menunjukkan bahwa model *denoising* mampu memulihkan detail spasial citra MRI secara lebih akurat dengan tingkat kesalahan rekonstruksi piksel yang lebih rendah.

Fungsi *Loss* PSNR

PSNR mengukur rasio antara kekuatan maksimum sinyal (dalam hal ini, gambar asli) dan kekuatan korupsi *noise* yang mempengaruhi kesetiaan representasinya. Nilai PSNR yang

lebih tinggi menunjukkan kualitas gambar yang lebih baik, karena menandakan distorsi yang lebih sedikit dibandingkan dengan gambar asli. *Peak Signal-to-Noise Ratio* (PSNR) penting digunakan untuk mengevaluasi kualitas gambar [14]. PSNR digunakan untuk mengukur kualitas rekonstruksi gambar atau video yang telah terkompresi dibandingkan dengan versi aslinya.

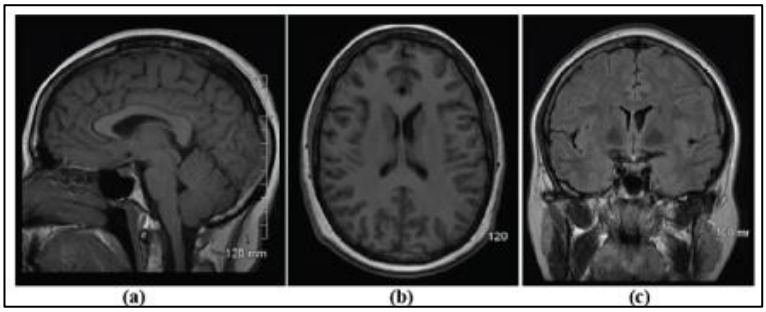
$$PSNR = 10 \times log_{10} \frac{(data\ range)^2}{MSE}$$

Persamaan 2.3 yang ditunjukkan adalah persamaan untuk menghitung *Peak Signal-to-Noise Ratio* (PSNR). *Data range* adalah rentang nilai maksimum piksel (misalnya, 1.0 untuk gambar yang dinormalisasi ke rentang 0-1, atau 255 untuk gambar 8-*bit*. Nilai MSE menunjukkan rata-rata kesalahan kuadrat antara piksel-piksel gambar asli dan gambar yang terkompresi. PSNR diukur dalam satuan *decibel* (dB), yang memungkinkan pengukuran rasio antara kekuatan gambar asli dan *noise* kesalahan dalam skala logaritmik. Penggunaan satuan dB ini memudahkan pemahaman tentang kualitas gambar, di mana nilai PSNR yang lebih tinggi menunjukkan kualitas visual yang lebih baik. Nilai di atas 30 dB menunjukkan kualitas yang sangat baik sesuai untuk diagnosis medis, sementara nilai antara 20–30 dB masih dapat diterima meskipun mulai terlihat penurunan kualitas. di bawah 20 dB menunjukkan distorsi yang dapat mengganggu interpretasi citra [15] [16].

BAB III
CITRA MRI OTAK DAN JENIS *NOISE*

➢ Pengantar MRI Otak: Fungsi dan Struktur

Magnetic Resonance Imaging (MRI) adalah teknik pencitraan yang kuat yang telah mengubah diagnostik dan pembahasan medis, terutama dalam ilmu saraf. MRI beroperasi dengan memanfaatkan urutan pulsa spesifik yang membuat kontras dalam gambar peka terhadap berbagai parameter fisik ansambel *spin proton*. MRI sangat efektif dalam membedakan antara berbagai jenis jaringan, seperti materi putih dan abu-abu di otak. Salah satu keuntungan signifikan MRI adalah kemampuannya menghasilkan gambar resolusi tinggi dengan kontras jaringan lunak yang sangat baik tanpa radiasi pengion. Teknologi ini memanfaatkan medan magnet kuat dan gelombang radio untuk membangkitkan sinyal dari inti hidrogen dalam jaringan, sehingga menghasilkan citra anatomi yang sangat detail. MRI menjadi pilihan yang lebih disukai untuk mendiagnosis kondisi seperti penyakit *neurodegeneratif*, tumor otak, dan stroke. Peran MRI fungsional (fMRI) dalam memahami fungsi otak melalui perubahan metabolisme dan aliran darah. MRI anatomi telah memfasilitasi pembahasan perkembangan otak, penuaan, dan gangguan neurologis [17].



Gambar 9. MRI Otak (a) Bidang Sagital (b) Bidang Aksial (c) Bidang Koronal

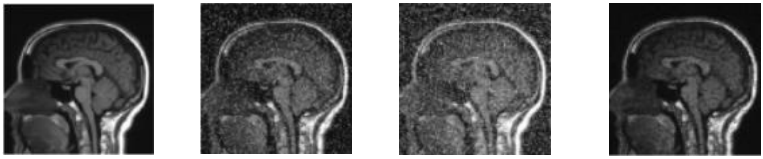
Pada Gambar 9 terlihat pemindai MRI dapat menghasilkan tiga jenis orientasi kepala manusia. Bidang dasar MRI dari atas ke bawah (bidang aksial), dari depan ke belakang (bidang koronal), dan dari samping ke samping (bidang sagital) [18].

- **Jenis *Noise* pada Citra Medis:**

Artefak dalam gambar mengacu pada gangguan yang muncul sebagai bintik-bintik, sering disebabkan oleh kotoran atau kotoran yang menempel pada gambar [19]. Bintik-bintik ini bukan bagian dari konten gambar yang sebenarnya tetapi diperkenalkan karena berbagai alasan. Artefak dapat timbul tidak hanya dari ketidaksempurnaan selama pengambilan gambar atau transmisi tetapi juga dari kotoran yang ada di dalam gambar itu sendiri. Artefak dalam konteks gambar MRI mengacu pada gangguan yang tidak diinginkan yang bermanifestasi sebagai bintik-bintik atau bintik pada gambar. Ini bukan bagian dari konten gambar yang sebenarnya tetapi diperkenalkan karena berbagai faktor, seperti ketidaksempurnaan dalam akuisisi gambar atau proses transmisi [20].

Salt and Pepper

Salt and Pepper Noise atau *noise* garam dan merica merupakan salah satu jenis *noise* impulsif yang muncul pada citra digital dalam bentuk piksel yang tersebar secara acak, tampak sebagai bintik-bintik hitam dan putih yang kontras. *Noise* ini terjadi akibat adanya gangguan pada sensor pencitraan atau kesalahan dalam proses transmisi data selama akuisisi citra. Secara spesifik, *noise* ini timbul ketika beberapa piksel dalam citra mengambil nilai intensitas ekstrem, di mana piksel putih biasanya merepresentasikan *Salt*, sedangkan piksel hitam melambangkan *pepper* dalam skala keabuan 8-*bit* [21] seperti yang dapat dilihat pada Gambar 10. (b) *Salt and Pepper*.



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

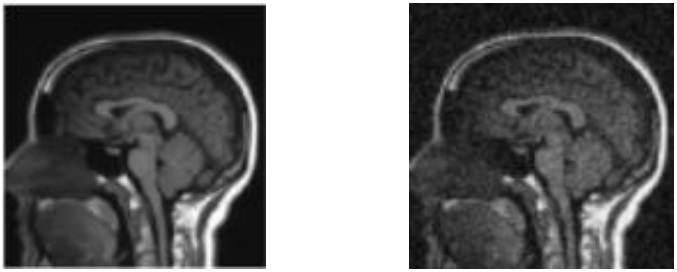
Gambar 10. *Salt* & *Pepper*

Salt and Pepper Noise dapat diuraikan lebih lanjut menjadi dua komponen utama, yaitu *Salt Noise* terlihat pada Gambar (c) dan *pepper noise* terlihat pada Gambar (d), yang masing-masing memiliki karakteristik berbeda dalam memengaruhi citra digital. *Salt noise* merujuk pada gangguan berupa kemunculan piksel putih yang muncul secara acak di berbagai lokasi dalam citra. *Noise* ini memberikan kesan bintik terang yang menyimpang dari pola visual sebenarnya, sehingga dapat menutupi detail penting pada area yang terdampak. Sebaliknya, *pepper noise* merupakan *noise* yang ditandai dengan munculnya piksel hitam yang juga tersebar secara acak. Kemunculan *Noise Salt and Pepper* dapat

disebabkan oleh berbagai faktor, termasuk kerusakan atau ketidaksempurnaan sensor, serta gangguan dalam transmisi data. Dalam kondisi tertentu, transmisi data melalui saluran yang bising dapat menghasilkan lonjakan atau penurunan intensitas piksel yang tidak semestinya, sehingga memicu efek garam dan merica pada citra digital [22]. Hal ini dapat berdampak negatif terhadap kualitas citra diagnostik yang dihasilkan, sehingga berpotensi mengganggu proses identifikasi dan analisis kondisi medis pasien [23].

Speckle

Speckle Noise merupakan jenis *noise* multiplicatif yang umum dijumpai dalam citra medis, khususnya pada modalitas pencitraan seperti *ultrasonografi* (USG) dan MRI [24]. *Noise* ini muncul dalam bentuk pola granular atau bintik-bintik acak yang tersebar di seluruh citra, sehingga memberikan tampilan kasar atau berbintik pada permukaan citra. *Speckle Noise* bukan berasal dari gangguan eksternal, melainkan merupakan hasil dari fenomena interferensi koheren antara gelombang pantul yang berasal dari berbagai permukaan mikro dalam jaringan biologis [25].



(a) (b)

Gambar 11. *Speckle*

➢ Dampak *Noise* Terhadap Diagnosis

Dalam konteks pencitraan medis, keberadaan *Speckle Noise* merupakan salah satu tantangan utama yang secara langsung memengaruhi kualitas citra yang dihasilkan. Fenomena ini muncul akibat interferensi gelombang koheren pada sistem pencitraan berbasis ultrasonik maupun optik, sehingga menghasilkan pola granular yang tersebar di seluruh permukaan citra. Seperti yang dapat diamati pada Gambar 11, pola tersebut tampak sebagai variasi intensitas acak yang menyelimuti area citra secara merata.

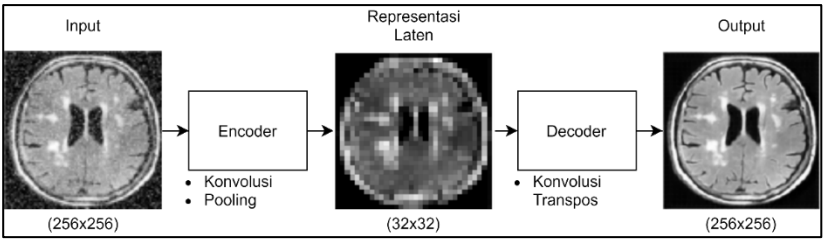
Keberadaan *Speckle Noise* berdampak signifikan terhadap kemampuan sistem maupun tenaga medis dalam menginterpretasikan informasi visual. *Noise* jenis ini terbukti mampu menurunkan kontras citra secara keseluruhan, sehingga menyulitkan identifikasi struktur halus maupun tepi objek yang seharusnya tampak jelas. Kondisi ini pada akhirnya menghambat proses diagnosis yang akurat, terutama pada kasus klinis yang memerlukan presisi tinggi dalam pembacaan citra.

Aspek yang menjadikan *Speckle Noise* lebih kritis dibandingkan jenis gangguan lainnya adalah sifat pembentukannya yang bersifat multiplicative, yakni *noise* ini tidak sekadar menambahkan nilai gangguan secara independen, melainkan menyatu dan berkembang bersama informasi asli dalam citra. Karakteristik inilah yang menyebabkan *Speckle Noise* jauh lebih sulit untuk dieliminasi dibandingkan *noise* aditif konvensional, sehingga dibutuhkan pendekatan pemrosesan citra yang lebih kompleks dan adaptif untuk penanganannya [26].

BAB IV
TEKNIK REDUKSI *NOISE* DENGAN CAE

➢ Prinsip Kerja CAE untuk *Denoising*

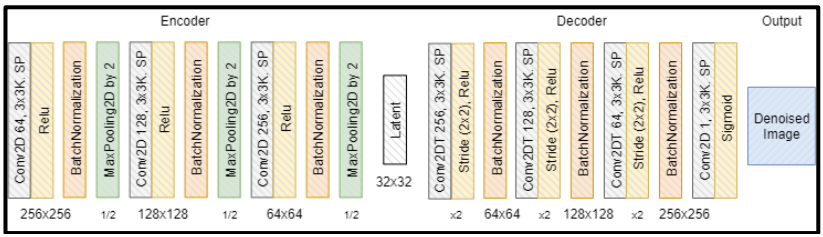
Perancangan model dalam pembahasan ini berfokus pada pengembangan (CAE) yang bertujuan untuk meng-hilangkan *noise* dari citra MRI otak. CAE ini terdiri dari dua komponen utama yaitu *encoder* dan *decoder*. *Encoder* berfungsi untuk mereduksi dimensi dan mengekstraksi fitur penting dari citra *input*, *bottleneck* menyimpan representasi paling *compact* dari citra, sementara *decoder* bertugas untuk merekonstruksi citra dari representasi yang telah direduksi oleh *encoder* seperti pada Gambar 12 Model ini dirancang untuk menerima citra MRI otak yang terdegradasi oleh *noise* dan menghasilkan citra yang lebih bersih dan mendekati aslinya [12].



Gambar 12. Model CAE

➢ Arsitektur Model: Desain *Encoder-Decoder*

CAE dirancang dengan memanfaatkan lapisan konvolusi dan konvolusi *Transpose* untuk merekonstruksi citra dari representasi yang terkompresi. Pada bagian *encoder*, model menggunakan beberapa lapisan konvolusi dengan ukuran *kernel* tertentu untuk mengekstraksi fitur dari citra *input*. Kemudian, data tersebut direduksi dimensinya melalui *max-pooling* dan disimpan dalam ruang laten. Pada bagian *decoder*, lapisan *Conv2DTranspose* digunakan untuk mengembalikan data ke dimensi aslinya, menghasilkan citra yang direkonstruksi tanpa *noise*.



Gambar 13. Perancangan CAE

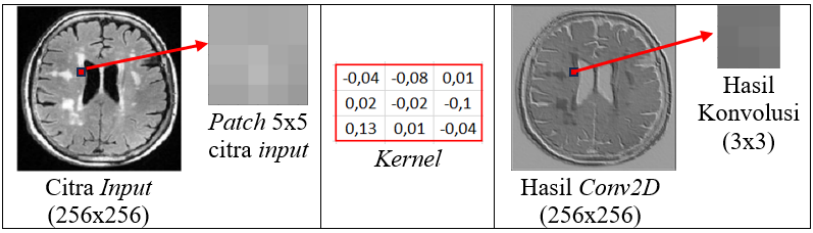
Pada Gambar 13 dapat dilihat model CAE yang terlihat pada gambar tersebut terdiri dari dua bagian utama, yaitu *encoder* dan *decoder*, yang bekerja untuk memproses dan menghasilkan citra yang telah melalui proses *denoising*. Pada bagian *encoder*, dimulai dengan lapisan konvolusi pertama menggunakan *kernel* berukuran 3x3 dengan jumlah *filter* 64, yang diikuti oleh fungsi aktivasi *ReLU*. Setelah itu, lapisan *BatchNormalization* diterapkan untuk menstabilkan distribusi *input* dari *layer* berikutnya. Kemudian, dilakukan *MaxPooling2D* dengan faktor pengurangan 2x2 untuk mengurangi dimensi spasial dari citra. Proses ini berlanjut melalui beberapa lapisan konvolusi dengan peningkatan

jumlah *filter*, yaitu 128 dan 256, serta diteruskan dengan proses *MaxPooling2D* lagi, yang akhirnya menghasilkan representasi dengan dimensi 32x32 yang merepresentasikan fitur-fitur penting dari citra yang telah diolah. Bagian *decoder* dimulai dengan melakukan transposisi konvolusi menggunakan *stride* 2x2 untuk meningkatkan dimensi spasial citra, yang diawali dengan konvolusi dengan *filter* berukuran 256. Proses ini terus berlanjut dengan lapisan-lapisan konvolusi yang berfungsi untuk memperbesar citra hingga mencapai dimensi 256x256. Setelah beberapa tahap transposisi dan normalisasi, proses berakhir dengan lapisan konvolusi akhir yang menghasilkan citra keluaran dengan dimensi yang sama dengan citra *input* awal, yaitu 256x256, yang kemudian diproses lebih lanjut dengan fungsi aktivasi *Sigmoid* untuk menghasilkan citra yang telah di-*denoise*, yakni citra bebas dari gangguan *noise*.

- **Lapisan Konvolusi dan Konvolusi *Transpose***

Perhitungan *Conv2D*

Misalkan pada Gambar 14 diambil *patch* piksel berukuran 5x5 dari citra "*N_109.jpg*" dengan koordinat x (baris) mulai dari 99 hingga 103 dan y (kolom) mulai dari 100 hingga 104. *Patch* ini menunjukkan area yang diambil dari citra asli dengan ukuran 5x5 piksel. Proses ini dilakukan dengan memilih bagian citra yang berada pada rentang indeks baris dan kolom yang kemudian digunakan sebagai *input* untuk operasi konvolusi dengan *kernel* tertentu.



Gambar 14. *Patch Conv2D*

Citra *Input Patch* (5x5): $\begin{bmatrix} 0{,}67 & 0{,}67 & 0{,}65 & 0{,}66 & 0{,}66 \\ 0{,}67 & 0{,}67 & 0{,}65 & 0{,}66 & 0{,}67 \\ 0{,}69 & 0{,}69 & 0{,}71 & 0{,}65 & 0{,}64 \\ 0{,}67 & 0{,}67 & 0{,}72 & 0{,}67 & 0{,}63 \\ 0{,}64 & 0{,}64 & 0{,}68 & 0{,}64 & 0{,}60 \end{bmatrix}$

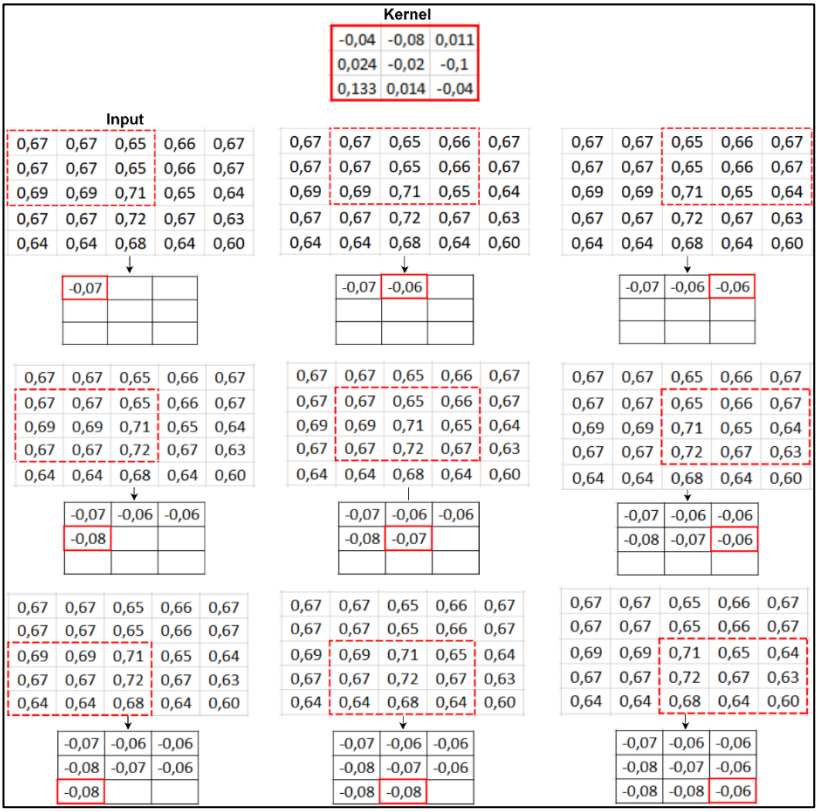
Langkah selanjutnya adalah melakukan operasi konvolusi, yaitu dengan mengalikan setiap elemen dalam *patch* 5x5 dengan elemen yang sesuai dalam *kernel* 3x3, dan kemudian menjumlahkan hasil perkalian tersebut. Misalnya, untuk elemen pertama dalam hasil konvolusi, kita melakukan operasi sebagai berikut:

$$(0{,}67 \times -0{,}04) + (0{,}67 \times -0{,}08) + (0{,}65 \times 0{,}011) + (0{,}69 \times 0{,}024) + (0{,}69 \times -0{,}02) + (0{,}71 \times -0{,}1) + (0{,}67 \times 0{,}133) + (0{,}67 \times 0{,}014) + (0{,}65 \times -0{,}04) = (-0{,}07)$$

Hasil dari operasi ini akan memberikan satu nilai yang mewakili kontribusi dari *patch* pertama terhadap *output* konvolusi pada posisi tersebut. Proses ini akan diulang untuk seluruh area citra sesuai dengan ukuran *stride* yang digunakan. Setelah semua hasil perkalian dan penjumlahan dilakukan, kita akan mendapatkan hasil konvolusi dengan ukuran yang lebih kecil, yaitu 3x3. Seperti yang dapat dilihat pada Gambar 14 dan Gambar 15 yang merupakan *patch* piksel berukuran 3x3 hasil dari proses *Conv2D* pada citra "*N_109.jpg*" dengan koordinat x

(baris) mulai dari 100 hingga 102 dan y (kolom) mulai dari 100 hingga 102.

Hasil Konvolusi (3x3) = $\begin{bmatrix} -0{,}07 & -0{,}06 & -0{,}06 \\ -0{,}08 & -0{,}07 & -0{,}06 \\ -0{,}08 & -0{,}08 & -0{,}06 \end{bmatrix}$

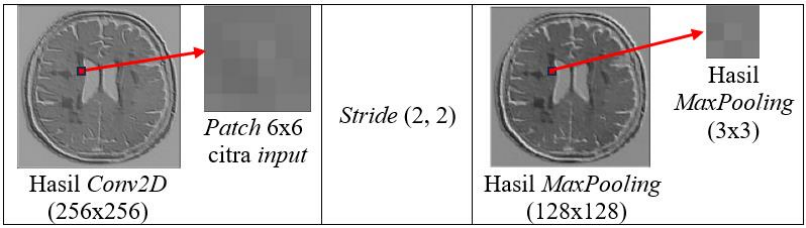


Gambar 15. Perhitungan *Conv2D*

Perhitungan *MaxPooling*

Misalkan pada Gambar 15 diambil *patch* piksel berukuran 6x6 dari citra hasil *Conv2D* "*N_109.jpg*" dengan koordinat x (baris) mulai dari 99 hingga 103 dan y (kolom) mulai dari 100 hingga

104. Proses ini dilakukan dengan memilih bagian citra yang berada pada rentang indeks baris dan kolom tertentu, yang selanjutnya digunakan sebagai *input* untuk operasi *MaxPooling*. Dalam proses *MaxPooling*, setiap blok berukuran 2x2 piksel dari matriks asli diambil secara sistematis, kemudian nilai maksimum dari masing-masing blok tersebut dipilih sebagai representasi fitur yang dominan. Pendekatan ini bertujuan untuk mereduksi dimensi spasial citra sekaligus mempertahankan informasi yang paling signifikan dari setiap wilayah lokal citra yang telah diproses.



Gambar 16. *Patch MaxPooling*

Patch hasil *Conv2D* 6x6:

$$\begin{bmatrix} -0{,}08 & -0{,}08 & -0{,}08 & -0{,}08 & -0{,}07 & -0{,}07 \\ -0{,}08 & -0{,}07 & -0{,}07 & -0{,}07 & -0{,}08 & -0{,}07 \\ -0{,}08 & -0{,}08 & -0{,}07 & -0{,}06 & -0{,}08 & -0{,}07 \\ -0{,}08 & -0{,}09 & -0{,}08 & -0{,}07 & -0{,}07 & -0{,}08 \\ -0{,}08 & -0{,}09 & -0{,}09 & -0{,}08 & -0{,}07 & -0{,}07 \\ -0{,}07 & -0{,}07 & -0{,}08 & -0{,}08 & -0{,}07 & -0{,}07 \end{bmatrix}$$

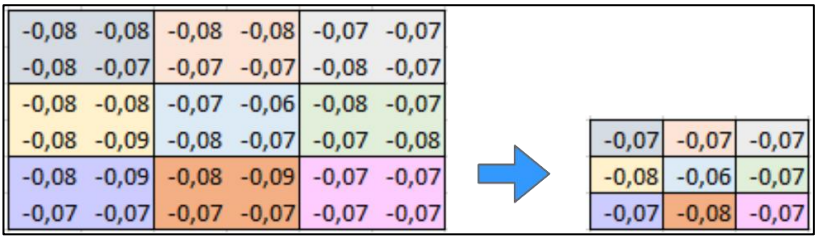
Patch 6x6 tersebut kemudian dibagi menjadi beberapa kecil berukuran 2x2. Pembagian ini dilakukan untuk mempersiapkan proses operasi *MaxPooling*, di mana tiap blok akan dipilih nilai maksimum dari empat elemen dalam blok tersebut proses ini diulang untuk seluruh blok 2x2. Setelah seluruh blok 2x2 diproses, kita akan memperoleh matriks hasil *pooling* yang lebih kecil, yakni berukuran 3x3, yang terdiri dari

nilai maksimum dari setiap blok 2x2. Hasil ini merupakan citra tereduksi yang lebih kecil. Berikut perhitungan proses *pooling* pada matriks 6x6.

$$\text{Blok 1 } Max(-0{,}08;\ -0{,}08;\ -0{,}08;\ -0{,}07) = -0{,}07$$
$$\text{Blok 2 } Max(-0{,}08;\ -0{,}08;\ -0{,}07;\ -0{,}07) = -0{,}07$$
$$\text{Blok 3 } Max(-0{,}07;\ -0{,}07;\ -0{,}08;\ -0{,}07) = -0{,}07$$
$$\text{Blok 4 } Max(-0{,}08;\ -0{,}08;\ -0{,}08;\ -0{,}09) = -0{,}08$$
$$\text{Blok 5 } Max(-0{,}07;\ -0{,}06;\ -0{,}08;\ -0{,}08) = -0{,}06$$
$$\text{Blok 6 } Max(-0{,}08;\ -0{,}07;\ -0{,}07;\ -0{,}08) = -0{,}07$$
$$\text{Blok 7 } Max(-0{,}08;\ -0{,}09;\ -0{,}07;\ -0{,}07) = -0{,}07$$
$$\text{Blok 8 } Max(-0{,}09;\ -0{,}08;\ -0{,}08;\ -0{,}08) = -0{,}08$$
$$\text{Blok 9 } Max(-0{,}07;\ -0{,}07;\ -0{,}07;\ -0{,}07) = -0{,}07$$

$$\text{Hasil } \textit{MaxPooling} \text{ (3x3)} = \begin{bmatrix} -0{,}07 & -0{,}07 & -0{,}07 \\ -0{,}08 & -0{,}06 & -0{,}07 \\ -0{,}07 & -0{,}08 & -0{,}07 \end{bmatrix}$$

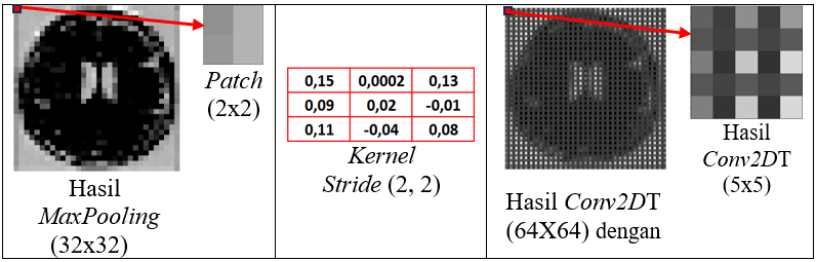
Dalam proses *MaxPooling* seperti yang ditunjukkan pada Gambar 17, citra masukan dibagi menjadi beberapa blok berukuran 2×2 tanpa tumpang tindih, sesuai dengan *stride* 2. Untuk setiap blok tersebut, nilai maksimum dari empat elemen di dalamnya dipilih sebagai representasi untuk blok tersebut di *feature* map *output*. Contohnya, dari blok berwarna abu-abu yang terdiri dari nilai (-0,08; -0,08; -0,08; -0,07) nilai maksimum yaitu -0,07 diambil dan dimasukkan ke dalam posisi yang sesuai pada hasil pooling, dan proses ini akan diulang ke semua blok.



Gambar 17. Perhitungan *MaxPooling*

Perhitungan *Conv2D Transpose*

Misalkan pada Gambar 3.18 diambil *patch* piksel berukuran 2x2 dari citra "*N_109.jpg*" saat berada pada *bottleneck* dengan koordinat x (baris) mulai dari 1 hingga 2 dan y (kolom) mulai dari 1 hingga 2. *Patch* ini menunjukkan area yang diambil dari citra asli dengan ukuran 2x2 piksel. Proses ini dilakukan dengan memilih bagian citra yang berada pada rentang indeks baris dan kolom yang kemudian digunakan sebagai *input* untuk operasi konvolusi *Transpose*.



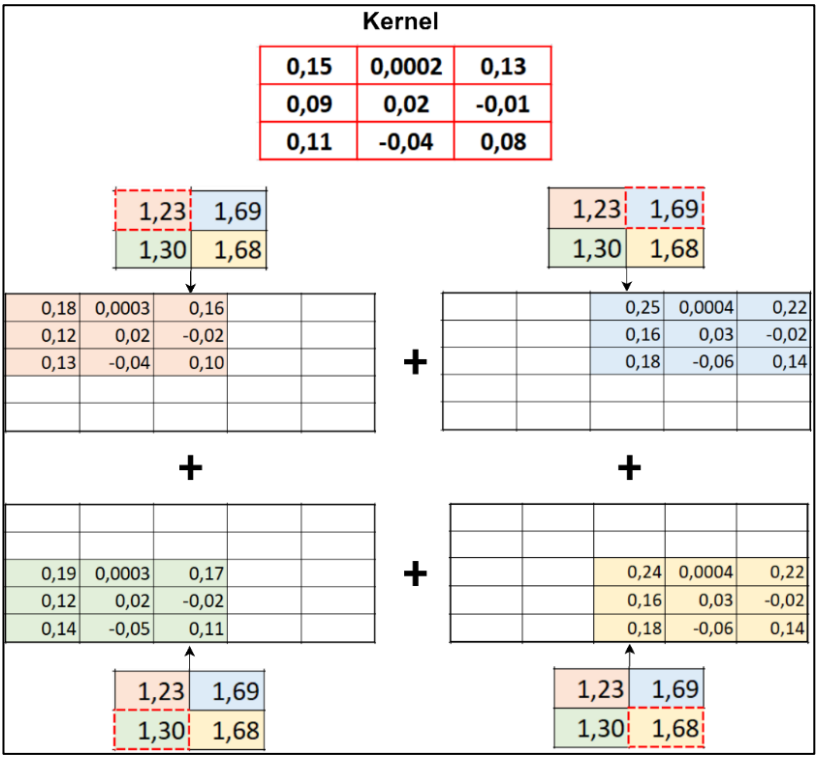
Gambar 18. *Patch Conv2D Transpose*

Patch hasil *MaxPooling* (2x2): $\begin{bmatrix} 1{,}23 & 1{,}69 \\ 1{,}30 & 1{,}68 \end{bmatrix}$

Pada *Conv2D Transpose*, diambil *patch* (2x2) setiap piksel dari *input* seperti (32×32) ditempatkan pada *grid output* dengan jarak tertentu sesuai dengan nilai *stride*. Dalam hal ini, dengan *stride* = 2, jarak antar piksel yang dihasilkan menjadi dua kali

lebih jauh, sehingga ukuran awalnya diperluas secara spasial. *Kernel* berukuran 3×3 kemudian diaplikasikan untuk menghasilkan nilai piksel *output* berdasarkan konvolusi terbalik, di mana bobot *kernel* disebarkan ke lokasi yang berdasarkan nilai *stride* pada citra *output*. Hasil akhir dari proses ini adalah citra berukuran 64×64 yang diperoleh dari citra 32×32, dengan peningkatan resolusi spasial melalui pembelajaran parameter *kernel*.

Dapat dilihat pada Gambar 19 Setiap nilai dalam *patch input* seperti 1,23; 1,69; 1,30; dan 1,68 digunakan untuk mengalikan seluruh *kernel* 3×3. Misalnya, nilai 1,23 (berwarna jingga) dengan *kernel*, menghasilkan matriks baru yang kemudian diletakkan pada posisi yang sesuai pada *output*, sesuai dengan posisi asli nilai *input* dalam *patch*, tetapi diperbesar sesuai dengan *stride*.



Gambar 19. Perhitungan *Conv2DTranspose*

Hal ini dilakukan untuk keempat nilai dalam *patch*, masing-masing dengan warna berbeda pada gambar (jingga, biru, hijau, dan kuning), menghasilkan empat buah hasil konvolusi. Setelah itu, semua hasil ini dijumlahkan, jika beberapa hasil *kernel overlap*, maka hasilnya akan dijumlahkan. Hasil *Conv2DT* =

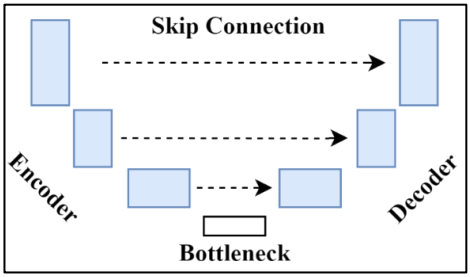
$$\begin{bmatrix} 0{,}18 & 0{,}0003 & (0{,}16+0{,}25) & 0{,}0004 & (0{,}22+\dots) \\ 0{,}12 & 0{,}02 & (-0{,}02+0{,}16) & (0{,}03) & (-0{,}02+\dots) \\ (0{,}13+0{,}19) & (-0{,}04+0{,}0003) & (0{,}1+0{,}18+0{,}24+0{,}17) & (-0{,}06+0{,}0004) & (0{,}14+\ 0{,}22+\dots) \\ 0{,}12 & 0{,}02 & (-0{,}02\ +\ 0{,}16) & 0{,}03 & (-0{,}02+\dots) \\ (0{,}14+\dots) & (-0{,}05+\dots) & (0{,}11\ +\ 0{,}18+\dots) & (-0{,}06+\dots) & (0{,}14+\dots) \end{bmatrix}$$

Hasil *Conv2DT* =

$$\begin{bmatrix} 0{,}18 & 0{,}0003 & 0{,}41 & 0{,}0004 & (0{,}22+\dots) \\ 0{,}12 & 0{,}02 & 0{,}14 & 0{,}03 & (-0{,}02+\dots) \\ 0{,}32 & -0{,}04 & 0{,}69 & -0{,}06 & (0{,}36+\dots) \\ 0{,}12 & 0{,}02 & 0{,}14 & 0{,}03 & (-0{,}02+\dots) \\ (0{,}14+\dots) & (-0{,}05+\dots) & (0{,}29+\dots) & (-0{,}06+\dots) & (0{,}14+\dots) \end{bmatrix}$$

➢ Perbandingan dengan Model *U-Net*

Dalam proses pembahasan ini peneliti juga telah melakukan pengkajian terhadap metode lain yang memiliki arsitektur serupa dengan *Autoencoder*, yaitu *U-Net*, sebagai pembanding dalam proses *denoising* citra. *U-Net* dan *Autoencoder* pada dasarnya memiliki prinsip dasar yang serupa, yakni keduanya merupakan CNN yang dirancang untuk melakukan proses *encoding* dan *decoding* terhadap data citra. Keduanya bekerja dengan cara mengekstraksi fitur penting dari citra *input* (pada tahap *encoder*) dan merekonstruksi citra pada tahap *decoder*.

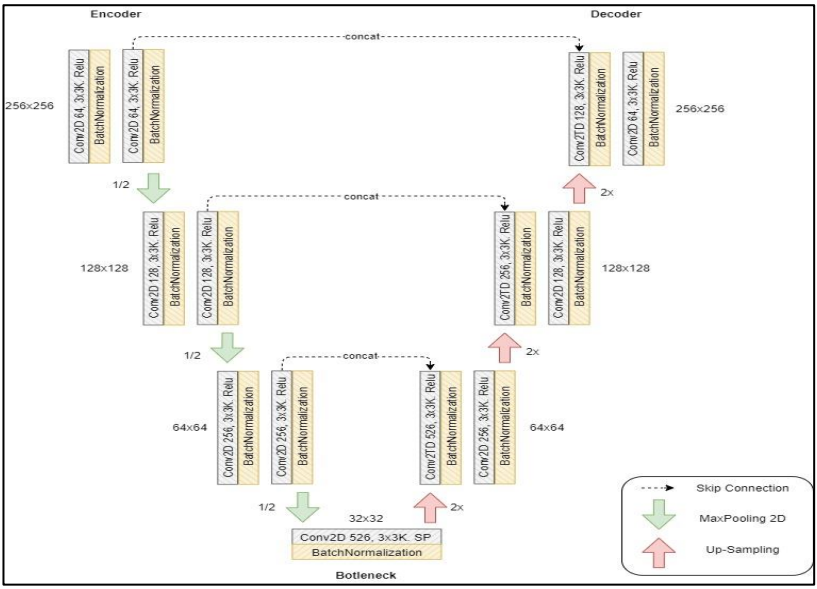


Gambar 20. *U-Net*

Terdapat perbedaan mendasar dalam arsitektur dan tujuan desain antara *U-Net* dan *Autoencoder*. *U-Net* dirancang memiliki struktur simetris dengan jalur *skip connection* yang menghubungkan setiap *layer encoder* dengan *layer decoder* yang bersesuaian seperti yang diilustrasikan pada Gambar 20 *Skip connection* ini memungkinkan informasi spasial dari tahap *encoder* untuk langsung ditransfer ke tahap *decoder*, sehingga

mampu mempertahankan detail resolusi tinggi yang penting dalam proses rekonstruksi. Sebaliknya, *Autoencoder* standar umumnya tidak memiliki jalur tersebut, sehingga dapat mengalami kehilangan informasi spasial selama proses *encoding*. Dalam konteks tugas *denoising*, penggunaan *U-Net* memungkinkan model untuk mempelajari representasi yang lebih kontekstual dan mempertahankan detail lokal secara lebih baik dibandingkan *Autoencoder* konvensional. Oleh karena itu, meskipun kedua metode memiliki kemiripan dalam pendekatan pembelajaran representasi, *U-Net* menawarkan keunggulan struktural yang berpotensi memberikan hasil rekonstruksi yang lebih presisi terutama pada citra dengan *noise* yang kompleks.

Pada Gambar 21, peneliti merancang arsitektur *U-Net* yang memiliki kemiripan setinggi mungkin dengan arsitektur *Autoencoder* yang telah dikembangkan sebelumnya. Upaya ini dilakukan untuk memastikan bahwa perbandingan performa antara kedua model dilakukan secara adil dan terkontrol, dengan meminimalkan variabel bebas yang tidak relevan. Dalam hal ini, *filter* pada lapisan konvolusional, ukuran *kernel*, fungsi aktivasi, dan parameter pelatihan seperti *learning rate* dan *Batch size* dibuat serupa. Perbedaan utama yang tetap dipertahankan adalah keberadaan jalur *skip connection* pada *U-Net*, yang secara struktural menjadi ciri khas dan keunggulan dari arsitektur tersebut.

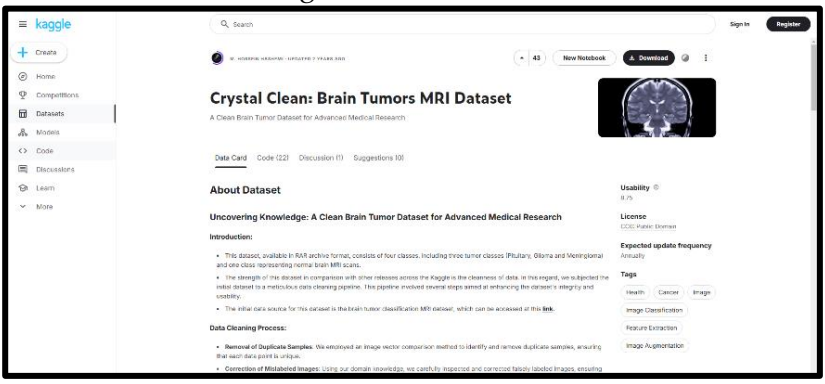


Gambar 21. Arsitektur *U-Net*

BAB V
DATASET DAN TEKNIK *PREPROCESSING*

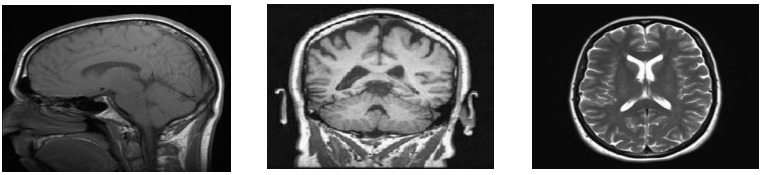
➢ Sumber dan Deskripsi Dataset MRI

Proses pengumpulan data dilakukan melalui website *Kaggle*, menggunakan dataset bernama "*Crystal Clean: Brain Tumors MRI Dataset*" yang dapat dilihat pada Gambar 22 Dataset ini terdiri dari dua kategori utama, yaitu citra MRI otak normal dan citra MRI otak dengan tumor.



Gambar 22. Dataset

Dalam pembahasan ini, data yang digunakan adalah citra MRI otak normal sebagai data sekunder. Untuk keperluan pelatihan model, terkumpul sebanyak 434 citra MRI otak normal dalam format "*jpg*" yang digunakan sebagai dataset.

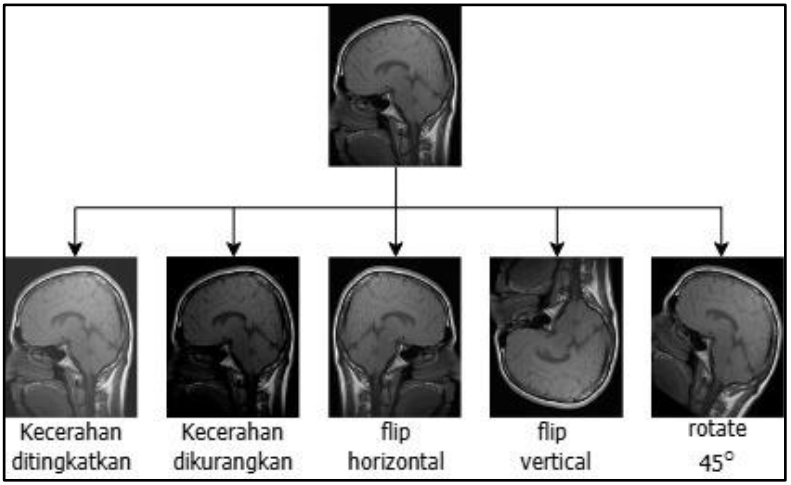


Gambar 23. *Sample* Data MRI Otak

Pada Gambar 23 merupakan contoh data MRI otak yang akan digunakan dalam pembahasan ini [27] terdapat 39 citra bidang Koronal, 91 citra bidang Sagital, dan 304 bidang Aksial. Data yang dikumpulkan ini disiapkan untuk proses lebih lanjut dalam tahap praproses dan pelatihan model CAE.

➢ Teknik Augmentasi Citra

Augmentasi data merupakan salah satu langkah penting dalam pembahasan ini untuk meningkatkan varian data pelatihan tanpa perlu mengumpulkan data tambahan. Proses ini dilakukan dengan memodifikasi citra MRI otak menggunakan berbagai teknik augmentasi, sehingga model dapat lebih baik dalam menangani berbagai variasi pada data *input*. Augmentasi ini bertujuan untuk meningkatkan generalisasi model dengan mensimulasikan kondisi nyata yang bervarian. Beberapa teknik augmentasi yang digunakan seperti *Brightness Adjustment*, *Darkness Adjustment*, *Horizontal / Vertical Flip*, dan *Rotation*. Setelah dilakukan augmentasi menggunakan lima metode, jumlah data set menjadi 2604 gambar MRI dalam format "*jpg*".

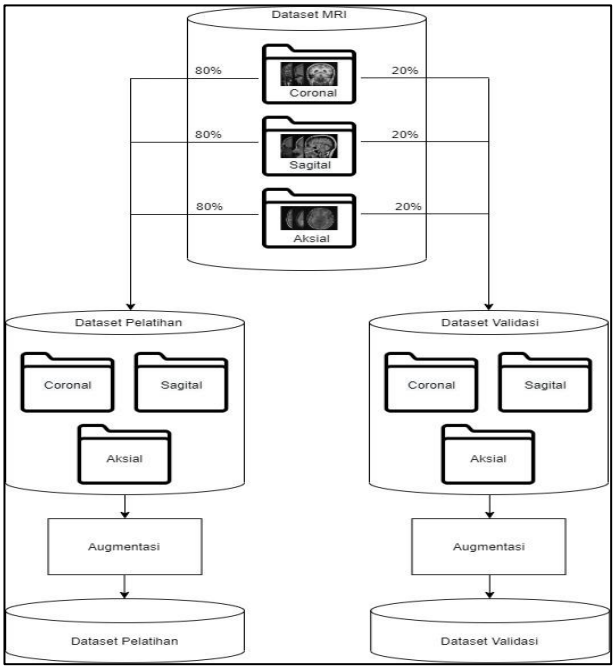


Gambar 24. Augmentasi

Gambar 24 menampilkan contoh dataset citra asli yang diolah menjadi lima variasi baru melalui serangkaian teknik augmentasi citra. Teknik-teknik yang diterapkan meliputi peningkatan kecerahan (*brightness increase*), pengurangan kecerahan (*brightness decrease*), pembalikan horizontal (*horizontal flip*), pembalikan vertikal (*vertical flip*), serta rotasi citra sebesar 45°. Setiap teknik menghasilkan representasi visual yang berbeda dari citra sumber yang sama, sehingga menghasilkan keragaman data yang lebih kaya tanpa memerlukan pengambilan citra baru.

Proses augmentasi ini dilakukan secara khusus untuk memperbanyak jumlah data dalam dataset pelatihan dengan tetap mempertahankan label kelas yang relevan [28]. Strategi ini umum diterapkan dalam pengembangan model berbasis *deep learning*, terutama ketika ketersediaan data asli terbatas. Dengan menghadirkan variasi transformasi geometris dan fotometris, model diharapkan mampu belajar dari distribusi data yang lebih beragam, sehingga meningkatkan kemampuan

generalisasi terhadap citra yang belum pernah ditemui sebelumnya.



Gambar 25. Alur Augmentasi

Proses dimulai dari data MRI yang memiliki tiga kelas, setiap kelas data dibagi menjadi dua subset, yakni 80% untuk dataset pelatihan dan 20% untuk dataset validasi. Pembagian ini dilakukan agar tiap kelas tetap terdistribusi secara seimbang di kedua subset tersebut. Setelah proses pembagian, tahap selanjutnya adalah augmentasi data, yang dilakukan secara terpisah baik pada dataset pelatihan maupun validasi. Augmentasi ini mencakup berbagai Transformasi citra seperti penambahan kecerahan, pengurangan kecerahan, *flip* vertikal, *flip horizontal*, dan rotasi. Seluruh Transformasi ini dilakukan secara eksplisit melalui manipulasi matriks piksel dengan

menggunakan pustaka *NumPy* dan *OpenCV*, tanpa menggunakan pustaka augmentasi eksternal Hasil akhir dari proses ini adalah dataset *training* dan daset valiadasi yang telah siap digunakan dalam pelatihan dan validasi model *denoising* seperti yang dapat dilihat pada Gambar 25.

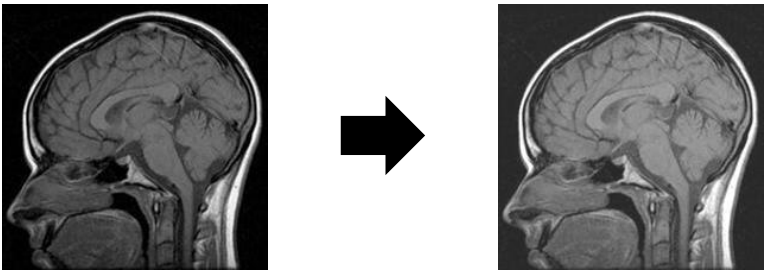
Setiap kelas melalui proses pembagian data 80:20. Selanjutnya setiap data akan diaugmentasikan dengan lima metode seperti peningkatan kecerahan, pengurangan kecerahan, *flip horizontal*, *flip* vertikal, dan rotasi. Proses ini dapat dilihat melalui kode 4.1 sampai kode 4.5.

Kode Program 4.1 Peningkatan Kecerahan

```
1  def brighten_image(image_np):
2      w, h, c = image_np.shape
3      brightening = np.zeros((w, h, 3), dtype=int)
4      for i in range(w):
5          for j in range(h):
6              for k in range(c):
7                  temp = image_np[i, j, k] + 40
8
9                  # Clipping
10                 if temp > 255:
11                     brightening[i, j, k] = 255
12                 else:
13                     brightening[i, j, k] = temp
14     return brightening
```

Kode 4.1 berfungsi untuk mencerahkan gambar dengan menambahkan nilai tertentu pada setiap piksel dalam gambar. Fungsi *brighten_image*(*image_np*) menerima *input* berupa *array NumPy* yang merepresentasikan gambar (*image_np*) dengan dimensi (w, h, c), di mana w adalah lebar gambar, h adalah tinggi gambar, dan c adalah jumlah saluran warna (biasanya 3 untuk RGB). Di dalam fungsi, setiap nilai piksel pada gambar

ditingkatkan sebesar 40, yang dilakukan dengan iterasi melalui semua piksel di setiap saluran warna (R, G, B). Setelah peningkatan nilai, dilakukan *clipping* untuk memastikan bahwa nilai piksel tidak melebihi batas maksimum 255, Jika hasil peningkatan melebihi 255, nilai piksel tersebut diubah menjadi 255.



Input *Output*

Gambar 26. Augmentasi Peningkatan Kecerahan

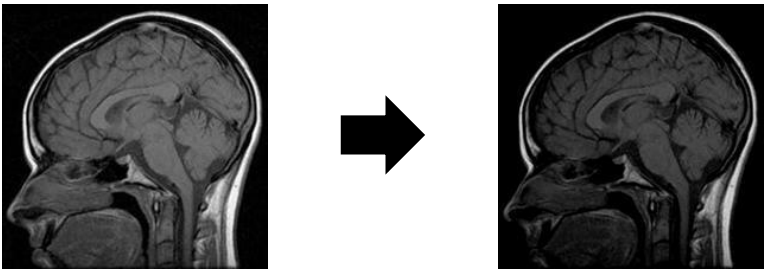
Gambar yang sudah diproses ini kemudian dikembalikan sebagai gambar yang lebih terang seperti yang dapat dilihat pada Gambar 26. Setelah melalui proses augmentasi dengan cara peningkatan kecerahan, proses dilanjutkan dengan mengaugmentasikan citra dengan metode *Darkness Adjustment* seperti yang terletak pada Kode 4.2.

Kode Program 4.2 Pengurangan Kecerahan

```
1  def darken_image(image_np):
2      w, h, c = image_np.shape
3      darkening = np.zeros((w, h, 3), dtype=int)
4
5      for i in range(w):
6          for j in range(h):
7              for k in range(c):
8                  temp = image_np[i, j, k] - 40
9
```

```
10          # Clipping
11              if temp < 0:
12                  darkening[i, j, k] = 0
13              else:
14                  darkening[i, j, k] = temp
15      return darkening
```

Kode 4.2 berfungsi untuk menggelapkan gambar dengan mengurangi nilai intensitas pikselnya. Fungsi *darken_image* (*image_np*) menerima *input* berupa *array NumPy* yang mewakili gambar (*image_np*) dengan dimensi (w, h, c), di mana w adalah lebar gambar, h adalah tinggi gambar, dan c adalah jumlah saluran warna (biasanya 3 untuk RGB). Fungsi ini mengurangi setiap nilai piksel gambar sebesar 40 untuk menghasilkan efek penggelapan. Selama proses ini, dilakukan *clipping* untuk memastikan bahwa nilai piksel tidak turun di bawah 0, nilai piksel minimum adalah 0. Jika hasil pengurangan piksel kurang dari 0, maka nilai piksel tersebut diubah menjadi 0.



Input *Output*

Gambar 27. Augmentasi Pengurangan Kecerahan

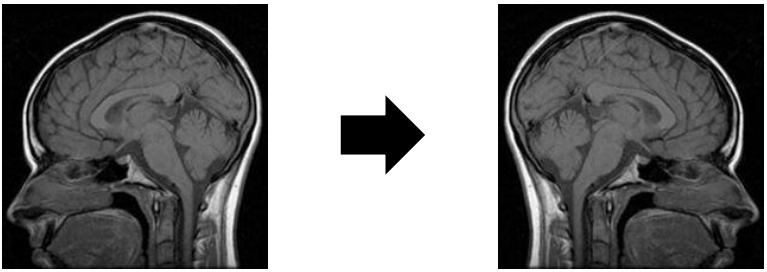
Hasil akhirnya adalah gambar yang lebih gelap sesuai dengan tingkat pengurangan yang ditentukan seperti yang dapat dilihat pada Gambar 27. Tahap berikutnya adalah melakukan proses augmentasi citra dengan menerapkan

transformasi flip horizontal guna memperluas variasi data latih model, sebagaimana ditunjukkan pada Kode 4.3.

Kode Program 4.3 Flip Horisontal

```
1  def flip_horizontal(image_np):
2      w, h, c = image_np.shape
3      flip = np.zeros((w, h, 3), dtype=int)
4
5      for i in range(w):
6          for j in range(h):
7              flip[i][h - 1 - j] = image_np[i][j]
8
9      return flip
```

Transformasi *flip horizontal* diterapkan sebagai bagian lanjutan dari proses augmentasi citra, yang disajikan pada Kode 4.3. dimana berfungsi untuk membalikkan gambar secara *horizontal* (*flip* cermin) dengan menggunakan *array NumPy*. Fungsi *flip_horizontal*(*image_np*) menerima gambar dalam bentuk *array NumPy* dengan dimensi (w, h, c), di mana w adalah lebar gambar, h adalah tinggi gambar, dan c adalah jumlah saluran warna (biasanya 3 untuk RGB). Fungsi ini kemudian membuat *array* kosong *flip* dengan dimensi yang sama untuk menampung hasil gambar yang telah dibalik. Di dalam dua *loop* bertingkat, yang pertama untuk baris (i) dan yang kedua untuk kolom (j), fungsi ini memetakan piksel gambar pada posisi (i, j) ke posisi baru (i, h - 1 - j) di mana kolomnya dibalik. Hasil akhirnya gambar terbalik di mana sisi kiri gambar akan menjadi sisi kanan dan sebaliknya.



Input *Output*

Gambar 28. Augmentasi *Flip Horizontal*

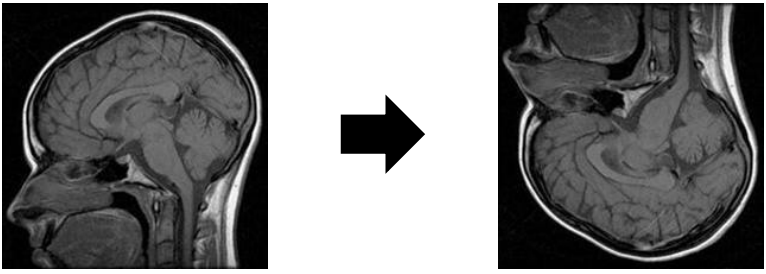
Gambar yang telah melalui proses tersebut kemudian dikembalikan sebagai keluaran berupa gambar yang telah terbalik secara horizontal, sebagaimana ditampilkan pada Gambar 28. Hasil transformasi ini memperlihatkan perubahan orientasi spasial citra secara simetris terhadap sumbu vertikal. Selanjutnya, proses augmentasi citra dilanjutkan dengan penerapan transformasi flip vertikal untuk memperoleh variasi orientasi tambahan, yang implementasinya dapat dilihat pada Kode 4.4.

Kode Program 4.4 Flip Vertikal

```
1   def flip_vertical(image_np):
2       w, h, c = image_np.shape
3       flip = np.zeros((w, h, 3), dtype=int)
4
5       # Proses flip vertikal (membalikkan dari atas ke bawah)
6       for i in range(w):
7           for j in range(h):
8               flip[w - 1 - i][j] = image_np[i][j]
9
10      return flip
```

Kode 4.4 berfungsi untuk membalikkan gambar secara vertikal (*flip* vertikal) menggunakan *array NumPy*. Fungsi *flip_vertical*(*image_np*) menerima gambar dalam bentuk *array NumPy* dengan dimensi (*w, h, c*), di mana w adalah lebar gambar, h adalah tinggi gambar, dan c adalah jumlah saluran warna (biasanya 3 untuk *RGB*). Fungsi ini membuat *array* kosong *flip* dengan dimensi yang sama untuk menampung hasil gambar yang telah dibalik. Dalam dua *loop* bertingkat, yang pertama untuk baris (*i*) dan yang kedua untuk kolom (j), fungsi ini memetakan setiap piksel dari posisi (*i, j*) ke posisi baru (*w - 1 - i, j*), yang mengubah urutan baris gambar dari atas ke bawah.

Hasil akhirnya adalah gambar yang terbalik secara vertikal, di mana bagian atas gambar menjadi bagian bawah dan sebaliknya. Transformasi ini menghasilkan representasi citra yang mencerminkan perubahan orientasi secara simetris terhadap sumbu horizontal. Setiap piksel yang semula berada di baris teratas akan dipindahkan ke posisi baris terbawah, sehingga struktur spasial citra mengalami inversi penuh dari atas ke bawah. Hasil visualisasi dari proses augmentasi flip vertikal ini dapat dilihat pada Gambar 29.



Input *Output*

Gambar 29. Augmentasi *Flip* Vertikal

Kode Program 4.5 Rotasi

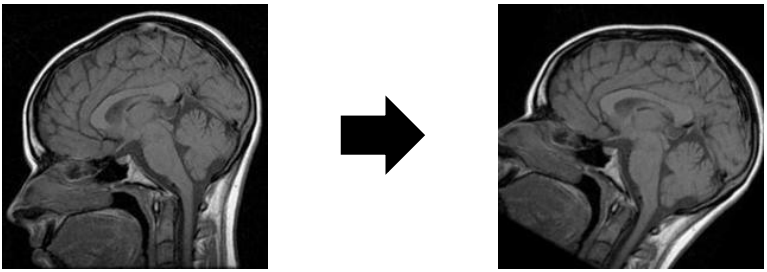
```
# Rotation
def rotate_image(image, angle=45):
    (h, w) = image.shape[:2]
    center = (w // 2, h // 2)
    # Matriks rotasi
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    # Hitung ukuran gambar baru setelah rotasi
    cos = np.abs(M[0, 0])
    sin = np.abs(M[0, 1])
    new_w = int((h * sin) + (w * cos))
    new_h = int((h * cos) + (w * sin))

    # Update translasi pada matriks rotasi agar gambar tetap di tengah
    M[0, 2] += (new_w / 2) - center[0]
    M[1, 2] += (new_h / 2) - center[1]

    # Rotasi dengan interpolasi bilinear dan latar belakang hitam
    rotated = cv2.warpAffine(image, M, (new_w, new_h), flags=cv2.INTER_LINEAR, borderValue=(0, 0, 0))
    return rotated
```

Transformasi rotasi diterapkan sebagai bagian lanjutan dari proses augmentasi citra, yang disajikan pada Kode 4.5. Kode ini berfungsi untuk memutar gambar dengan sudut tertentu menggunakan rotasi. Fungsi rotate_*image*(*image*, *angle*) menerima gambar dan sudut rotasi (dalam derajat) sebagai *input*. Pertama, sudut rotasi dikonversi dari derajat ke radian, dan nilai cosinus dan sinus dari sudut tersebut dihitung. Ukuran gambar asli (h, w) digunakan untuk menghitung dimensi gambar baru setelah rotasi, agar seluruh gambar dapat

ditampung. *Array* kosong *rotated_image* kemudian dibuat untuk menampung hasil rotasi.



Input *Output*

Gambar 30. Augmentasi Rotasi

Pusat gambar asli dan gambar yang diputar dihitung untuk menjaga rotasi tetap terpusat. Dalam dua *loop* bertingkat, fungsi ini memetakan setiap piksel pada posisi (*x, y*) dalam gambar asli ke koordinat baru (*new_x, new_y*) dalam gambar yang telah diputar, dengan memperhitungkan pergeseran berdasarkan pusat gambar. Setiap piksel dipindahkan ke lokasi baru yang sesuai jika koordinat tersebut berada dalam batas gambar hasil rotasi. Hasil rotasi akan seperti pada Gambar 30.

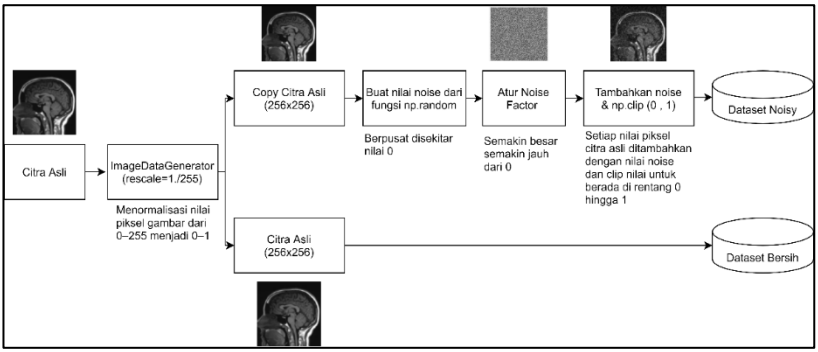
➢ Teknik Pemberian *Noise* Sintetis

Pada tahap ini, citra MRI otak yang digunakan sebagai data latih dan data validasi diberikan *noise* secara sengaja untuk mensimulasikan kondisi nyata di mana data sering kali terpengaruh oleh gangguan. Dikarenakan sulitnya mendapatkan citra MRI dengan *noise* dilakukan penambahan *noise* untuk menciptakan versi citra yang terkena *noise*, sehingga model dapat belajar menghilangkan *noise* tersebut [12]. Penambahan *noise* yang digunakan dalam pembahasan ini adalah *Salt Noise*, *Pepper Noise*, *Salt* & *Pepper Noise*, dan *Speckle*

noise, yang menyerupai artefak pada Citra MRI. Hasil dari tahap ini adalah dua set data. Set pertama berisikan data *training* bersih dan *training noise* dan set kedua berisikan data *validation* bersih dan *validation noise* yang masing-masing digunakan sebagai *input* ke model.

Induksi *Noise Speckle*

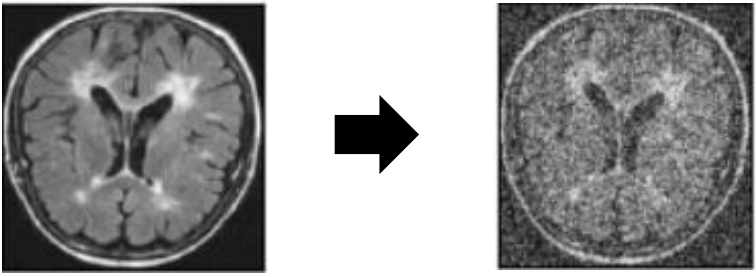
Dalam pembahasan ini, *Speckle Noise* ditambahkan ke dalam citra dataset dengan tujuan mensimulasikan di mana citra MRI sering mengalami gangguan. Dengan demikian, model yang dilatih diharapkan mampu melakukan generalisasi dan meningkatkan kemampuan dalam menghilangkan *noise* jenis ini dari citra MRI otak. Pertama citra melalui *ImageDataGenerator* (*rescale*=1./255), yang berfungsi untuk menormalkan nilai piksel ke rentang [0, 1]. *Speckle Noise* ditambahkan dengan cara membuat gangguan acak yang mengikuti distribusi normal, di mana setiap nilai *noise* dibangkitkan secara acak kemudian dikalikan dengan intensitas piksel asli sehingga menghasilkan pola derau yang proporsional terhadap kecerahan citra. Proses penambahan *noise* ini divisualisasikan seperti pada Gambar 31. *Output* dari tahap ini menghasilkan dua jalur pemrosesan yang berbeda, yakni jalur untuk dataset bersih tanpa gangguan dan jalur untuk dataset bising yang telah diberi *noise*.



Gambar 31. Penambahan *Speckle*

Kode Program Induksi *Speckle*	
1	def add_*Speckle_Noise*(*image*, *Noise_factor*):
2	*Noise* = *np.random.normal*(loc=0.0, *scale*=*Noise_factor*, *size*=*image*.shape)
3	*noisy_image* = *image* + *Noise*
4	return *np.clip*(*noisy_image*, 0., 1.)

Kode 4.8 memiliki fungsi untuk menambahkan *noise* jenis "*Speckle*" pada sebuah citra gambar. Fungsi ini menerima dua argumen: *image* (gambar *input*) dan *Noise_factor* (faktor yang mengatur intensitas *noise*). Pertama, fungsi menghasilkan *Noise Gaussian* menggunakan *np.random.normal*(), dengan mean (lokasi rata-rata) 0 dan deviasi standar (*scale*) sesuai dengan *Noise_factor*. *Noise* ini memiliki ukuran yang sama dengan citra *input* (*image*.shape). Kemudian, *noise* tersebut ditambahkan langsung ke citra *input* untuk menghasilkan citra dengan *noise*. Terakhir, gambar yang telah diberi *noise* dibatasi agar nilai pikselnya tetap berada dalam rentang [0, 1] dengan menggunakan *np.clip* dan hasilnya dapat dilihat pada Gambar 4.6 yang merupakan contoh penambahan *noise* dengan jenis *Speckle* terhadap salah satu citra dataset digunakan oleh model CAE belajar menghilangkan *noise* pada citra MRI otak.

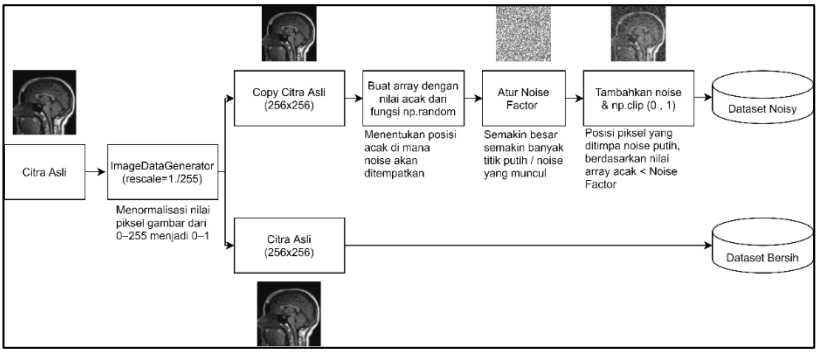


Input *Output*

Gambar 32. Induksi *Noise Speckle*

Induksi *Noise Salt*

Noise Salt ditambahkan ke dalam citra dataset dengan tujuan mensimulasikan kondisi nyata di mana citra MRI dapat mengalami gangguan berupa piksel-piksel acak yang sangat terang. Dengan demikian, model yang dilatih diharapkan mampu menghilangkan *noise* jenis ini dari citra MRI otak. Pertama citra melalui *ImageDataGenerator* (*rescale*=1./255), yang berfungsi untuk menormalkan nilai piksel ke rentang [0, 1]. Untuk menghasilkan efek ini, *Noise Salt* ditambahkan dengan cara mengganti nilai piksel secara acak menjadi nilai maksimum sehingga menghasilkan efek bintik putih seperti pada Gambar 33. *Output* dari tahap ini menghasilkan dua jalur pemrosesan, yakni jalur untuk dataset bersih dan jalur untuk dataset bising.

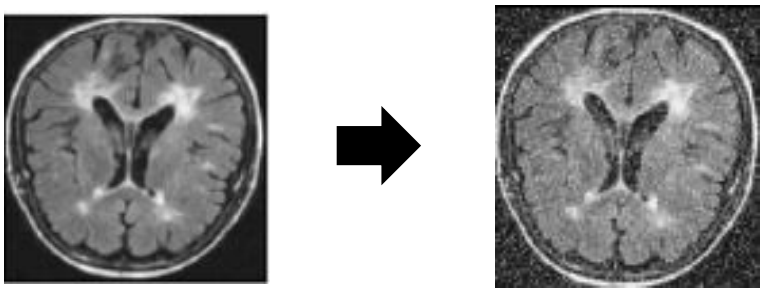


Gambar 33. Penambahan *Salt*

Kode Program 4.9 Induksi *Salt*	
1	def add_*Salt_Noise*(*image*, *Noise_factor*):
2	*noisy_image* = *np.copy*(*image*)
3	# *Salt* (white) *Noise*
4	*Salt_mask* = *np.random.random*(*size*=*image*.shape) < *Noise_factor*
	noisy_image[*Salt_mask*] = 1
5	
6	return *np.clip*(*noisy_image*, 0., 1.)
7	

Pada Kode 4.9 fungsi untuk menambahkan *noise* jenis *Salt* (garam) pada sebuah citra gambar. Fungsi ini menerima dua argumen yaitu *image* (gambar *input* yang akan diberi *noise*) dan *Noise_factor* (faktor yang menentukan intensitas *noise*). Pertama, fungsi membuat salinan gambar menggunakan *np.copy*(*image*). Kemudian, dengan menggunakan *np.random.random*(), dibuat sebuah *mask*er acak (*Salt_mask*) yang menghasilkan nilai *boolean* dengan probabilitas berdasarkan nilai *Noise_factor*. Pada setiap posisi yang memiliki nilai *True* dalam *masker*, piksel gambar tersebut diganti dengan nilai 1 (putih). Akhirnya, gambar yang telah diberi *noise* dipastikan berada dalam rentang [0, 1] menggunakan *np.clip*(), untuk menghindari nilai piksel yang

tidak valid, hasil dapat dilihat pada Gambar 4.7 yang merupakan contoh penambahan *noise* dengan jenis *Salt* terhadap salah satu citra dataset untuk nantinya digunakan oleh model CAE belajar menghilangkan *noise* pada citra MRI otak.

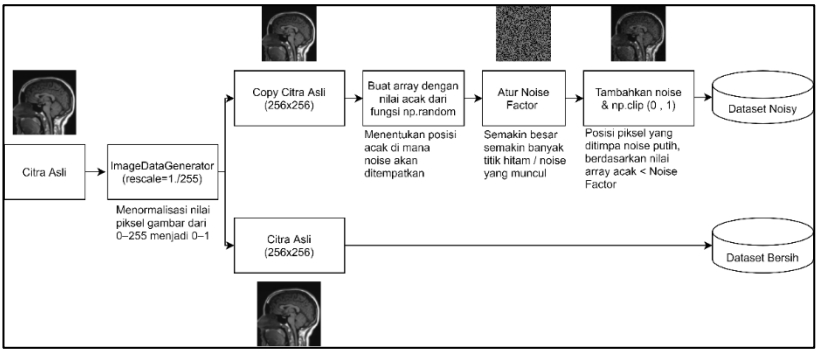


Input *Output*

Gambar 34. Induksi *Noise Salt*

Induksi *Noise Pepper*

Pepper Noise ditambahkan ke citra dataset untuk mensimulasikan gangguan pada citra MRI berupa piksel acak yang sangat gelap. Citra diproses melalui ImageDataGenerator (rescale=1./255) untuk menormalkan nilai piksel ke rentang [0,1]. Selanjutnya, Pepper *Noise* diterapkan dengan mengganti piksel secara acak menjadi nilai minimum (0), menghasilkan efek bintik hitam seperti pada Gambar 35. *Output* menghasilkan dua jalur, yakni dataset bersih dan dataset noisy untuk pelatihan model *denoising*.

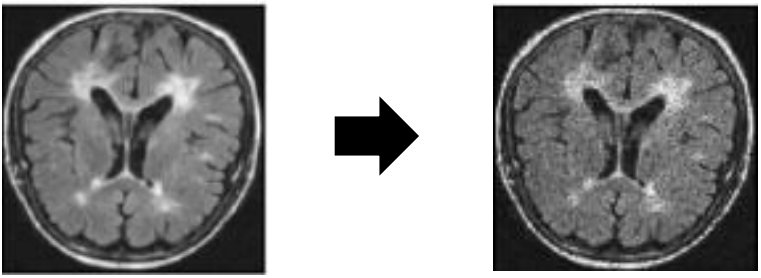


Gambar 35. Penambahan *Pepper*

Kode Program 4.10 Induksi *Pepper*	
1	def add_*Pepper_Noise*(*image*, *Noise_factor*):
2	*noisy_image* = *np.copy*(*image*)
3	
4	# *Pepper* (black) *Noise*
5	*Pepper_mask* = *np.random.random*(*size*=*image*.shape) <
6	*Noise_factor*
7	
8	*noisy_image*[*Pepper_mask*] = 0
9	return *np.clip*(*noisy_image*, 0., 1.)

Kode 4.10 adalah fungsi untuk menambahkan *Noise* jenis "*Pepper*" (hitam) pada sebuah citra gambar. Fungsi ini menerima dua argumen: *image* (gambar *input*) dan *Noise_factor* (faktor yang menentukan intensitas *Noise*). Pertama, fungsi membuat salinan gambar dengan *np.copy*(*image*). Kemudian, menggunakan *np.random.random*(), dibuat sebuah *mask*er acak (*Pepper_mask*) yang menghasilkan nilai *boolean* (*True or False*) dengan probabilitas sesuai dengan *Noise_factor*. Pada setiap posisi yang memiliki nilai *True* dalam *mask*er, piksel gambar tersebut diganti dengan nilai 0 (hitam). Akhirnya, gambar yang telah diberi *Noise* dipastikan berada dalam rentang [0, 1]

menggunakan *np.clip()*, untuk menghindari nilai piksel yang tidak valid, hasil dapat dilihat pada Gambar 36 yang merupakan contoh penambahan *noise* dengan jenis *pepper* terhadap salah satu citra dataset untuk nantinya digunakan oleh model CAE belajar menghilangkan *noise* pada citra MRI otak.



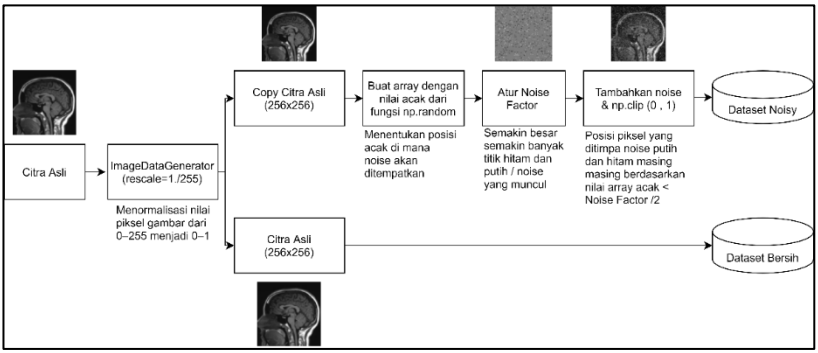
Input *Output*

Gambar 36. Induksi *Noise Pepper*

Induksi *Noise Salt and papper*

Noise Salt and Pepper ditambahkan ke dalam citra dataset dengan tujuan mensimulasikan kondisi nyata di mana citra MRI dapat mengalami gangguan berupa piksel-piksel acak yang sangat terang (putih) atau sangat gelap (putih). Pertama citra melalui *ImageDataGenerator* (*rescale=1./255*), yang berfungsi untuk menormalkan nilai piksel ke rentang [0, 1]. *Salt and Pepper Noise* ditambahkan dengan cara mengganti nilai piksel secara acak menjadi nilai minimum (0) yang menciptakan efek bintik hitam atau maksimum (1) yang menciptakan efek bintik putih, sehingga menghasilkan efek bintik atau *noise* hitam-putih yang menyerupai gangguan digital. Jika *noise_factor* = 0.2, maka ~10% piksel menjadi putih, dan ~10% menjadi hitam Seperti yang dapat dilihat pada Gambar 3.10. *Output* dari tahap

ini menghasilkan dua jalur pemrosesan, yakni jalur untuk dataset bersih dan jalur untuk dataset *noisy*.

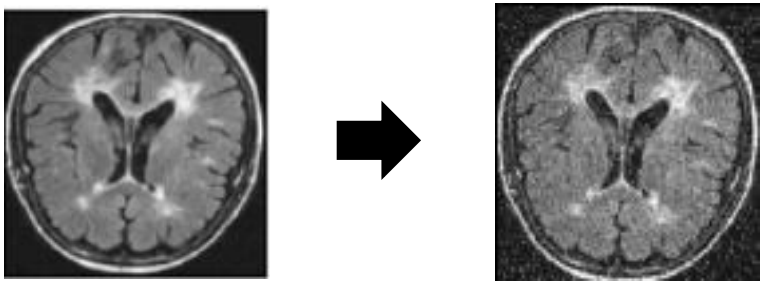


Gambar 37. Penambahan *Salt* & *Pepper*

Kode Program 4.11 Induksi *Salt and Papper*	
1	def add_*Salt*_and_*Pepper_Noise*(*image*, *Noise_factor*):
2	*noisy_image* = *np.copy*(*image*)
3	
4	# *Salt* (white) *Noise*
5	*Salt_mask* = *np.random.random*(*size*=*image*.shape) < (*Noise_factor*/2)
6	*noisy_image*[*Salt_mask*] = 1
7	
8	# *Pepper* (black) *Noise*
9	*Pepper_mask* = *np.random.random*(*size*=*image*.shape) < (*Noise_factor*/2)
10	*noisy_image*[*Pepper_mask*] = 0
11	
12	return *np.clip*(*noisy_image*, 0., 1.)

Pada Kode 4.11 Fungsi add_*Salt*_and_*Pepper_Noise* menambahkan *noise Salt and Pepper* pada gambar. Fungsi ini menerima *input* gambar (*image*) dan tingkat *Noise* (*Noise_factor*). Pertama, gambar disalin ke variabel baru untuk menjaga data

asli tetap utuh. Kemudian, *Noise Salt* (berwarna putih) ditambahkan dengan membuat *mask*er acak dan mengganti piksel yang dipilih menjadi 1. *Noise Pepper* (berwarna hitam) ditambahkan dengan cara serupa, tetapi mengganti piksel menjadi 0. Hasil akhirnya dikembalikan setelah nilai piksel dijepit dalam rentang [0, 1], dapat dilihat pada Gambar 38 yang merupakan contoh penambahan *noise* dengan jenis *Salt and Pepper* terhadap salah satu citra dataset untuk nantinya digunakan oleh model CAE belajar menghilangkan *noise* pada citra MRI otak.



Input *Output*

Gambar 38. Induksi *Noise* Salt and Pepper

Langkah selanjutnya adalah membentuk dua set data utama, yaitu data bersih dan data bising (*noisy*). Penambahan *noise* dilakukan secara otomatis saat proses pemuatan data menggunakan *generator*, sehingga setiap *Batch* citra yang dimuat akan dikombinasikan dengan versi bisingnya, dapat dilihat pada Kode 4.12. *generator* ini menghasilkan pasangan citra, satu mengandung *noise* dan satu lagi sebagai citra referensi tanpa gangguan. Dengan membagi data menjadi dua kelompok ini, model dapat dilatih untuk mempelajari pola perbedaan antara citra bising dan citra bersih, yang sangat penting dalam pengembangan model *denoising*. Proses ini

memungkinkan model untuk belajar secara langsung dari gangguan yang disimulasikan, serta mengevaluasi kemampuan generalisasi terhadap data validasi dalam kondisi bersih maupun terkorupsi.

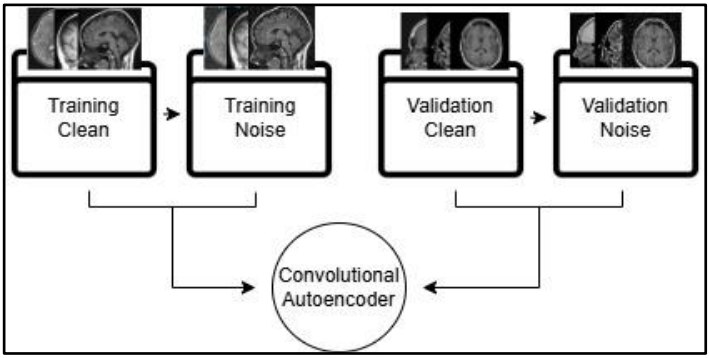
Kode Program 4.12 Dua Set Data

```
Batch_size = 8
target_size = (256, 256)
train_generator = load_images_from_directory(train_dir,
Batch_size, target_size)
validation_generator =
load_images_from_directory(validation_dir, Batch_size,
target_size)

def generate_noisy_images(generator, Noise_factor=0.15):
   while True:
      Batch = next(generator)
      noisy_Batch = add_Speckle_Noise(Batch, Noise_factor)
      yield noisy_Batch, Batch

noisy_train_generator = generate_noisy_images(train_generator)
noisy_validation_generator =
generate_noisy_images(validation_generator)
```

Hasil dari Kode 4.12 adalah dua kelompok data. Kelompok pertama terdiri dari data pelatihan bersih (*clean training data*) dan data pelatihan bising (*noisy training data),* sedangkan kelompok kedua mencakup data validasi bersih (*clean validation data*) dan data validasi bising (*noisy validation data*).



Gambar 39. Dua Set Data

Setiap subset data berfungsi sebagai masukan bagi model, sehingga memungkinkan model untuk belajar secara efektif dalam mengembangkan kemampuan *denoising* (penghilangan *noise*) [29] [30]. Pendekatan ini memungkinkan model CAE untuk mempelajari hubungan antara citra rusak dan versi aslinya secara berpasangan, sehingga model dapat mengembangkan kemampuan dalam menghilangkan *noise* (*denoising*) dengan akurasi yang tinggi. Pelibatan data bising dan bersih baik dalam pelatihan maupun validasi juga mendorong peningkatan kemampuan generalisasi model, pendekatan ini mendukung pengembangan model pelatihan model pada data terkorupsi dan tidak terkorupsi untuk meningkatkan efektivitas model dalam tugas restorasi citra citra medis, sebagaimana diilustrasikan dalam Gambar 39.

- **Pembagian Data dan Pemuatan Dataset**

Setelah melalui proses augmentasi, data akan dibagi (*splitting*) untuk memastikan bahwa model CAE dilatih dan dievaluasi dengan data yang berbeda, guna menghindari *overfitting* dan memastikan generalisasi model terhadap data baru [31]. Dataset setelah melalui augmentasi terdiri dari 2605 citra MRI

otak normal, yang kemudian dibagi dengan perbandingan 80:20. Sebanyak 80% dari data, yaitu 2.082 citra, digunakan sebagai *training* set, yang berfungsi untuk melatih model agar mampu mempelajari pola dan fitur dari data masukan. Sedangkan, 20% sisanya, yaitu 522 citra, digunakan sebagai *valdation* set. Data validasi ini digunakan untuk mengevaluasi kinerja model selama proses pelatihan, sehingga dapat diketahui seberapa baik model mampu menangani data yang tidak terlihat sebelumnya. Pembagian data dilakukan pada setiap kelas dalam dataset. Tujuan dari pendekatan ini adalah untuk memastikan bahwa baik data pelatihan maupun data validasi mencakup representasi dari seluruh kelas yang terdapat dalam citra MRI otak seperti yang terlihat pada Tabel 1.

Tabel 1. Pembagian Data

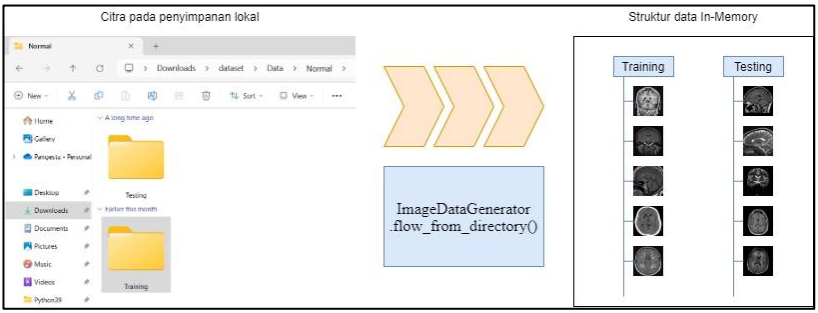
Jenis Data	Bidang MRI Otak	Sebelum Augmentasi	Setelah Augmentasi
Data Latih	Sagital	73	438
	Koronal	31	186
	Aksial	243	1458
Data Validasi	Sagital	18	108
	Koronal	8	48
	Aksial	61	366

Dengan begitu Model CAE memiliki kesempatan untuk mempelajari pola dan karakteristik dari setiap bagian otak dan kemampuan generalisasi model, sehingga model mampu menghasilkan performa yang baik saat diterapkan pada data baru dengan distribusi kelas yang serupa. Terlihat pada Tabel 1, jumlah data citra MRI otak setelah proses augmentasi

dilakukan pada data latih dan data validasi. Augmentasi data bertujuan untuk meningkatkan jumlah dan keragaman data pelatihan untuk memperkuat kemampuan generalisasi model. Setelah augmentasi, jumlah data latih mengalami peningkatan signifikan menjadi 438 citra sagittal, 186 citra koronal, dan 1.458 citra aksial. Namun demikian, tetap terlihat adanya ketidak-seimbangan kelas antar bidang citra, di mana citra aksial mendominasi jumlah data.

Pemuatan Data

Setelah melalui proses augmentasi dan pembagian data (data *splitting*), langkah berikutnya dalam alur pemodelan adalah memuat data ke dalam lingkungan pemrograman, seperti *Jupyter Notebook*, untuk keperluan pelatihan model. Proses ini diawali dengan menentukan direktori penyimpanan data latih dan data validasi, yang kemudian direpresentasikan dalam bentuk path pada variabel tertentu.



Gambar 40. Pemuatan Data

Untuk mempermudah pemrosesan data citra, digunakan pustaka *TensorFlow*, khususnya modul *ImageDataGenerator*, yang menyediakan metode efisien untuk memuat dan memroses data gambar secara otomatis. Salah satu metode penting yang digunakan adalah *flow_from_directory*(), yang memungkinkan pemuatan gambar langsung dari direktori

penyimpanan serta melakukan pra-pemrosesan seperti normalisasi nilai piksel (*rescaling*) dan pengubahan ukuran citra sesuai kebutuhan model.

Selain itu, metode ini juga mendukung pengacakan data (*shuffling*) dan dapat dikonfigurasi untuk memuat data dalam format tertentu, seperti skala abu-abu. Dengan pendekatan ini, data citra yang tersimpan dalam sistem file dapat langsung dimanfaatkan sebagai *input* model dalam format yang telah diproses secara konsisten, sehingga mendukung efisiensi dan keakuratan dalam pelatihan model pembelajaran mesin seperti yang diilustrasikan pada Gambar 40.

Setelah melalui augmentasi dan data splitting, data akan dimuat pada *Jupyter Notebook* yang nantinya akan digunakan sebagai bahan pelatihan model. Seperti yang dapat di lihat pada Kode 4.6 dimana variable *train_dir* menyimpan value berupa path untuk mengakses data traing. Sedangkan variable *validation* _dir menyimpan value berupa path untuk mengakses data validasi.

Kode Program 4.6 Pemuatan Data	
1	*train_dir* = "C:\\Users\\Pangestu
2	Siagian\\Downloads\\dataset\\Data\\Normal*Training* "
3	*validation*_dir = "C:\\Users\\Pangestu
4	Siagian\\Downloads\\dataset\\Data\\Normal\\Testing"

Selain itu, dengan memanfaatkan *library* milik *TensorFlow* dan menggunakan fitur *ImageDataGenerator* data yang memungkinkan pemuatan dan pemrosesan gambar secara efisien. Fungsi *flow_from_directory()* seperti yang dapat dilihat pada Kode 4.6 memungkinkan pemuatan gambar langsung dari direktori pada *disk*. Dengan melakukan pemuatan data, langkah-langkah praproses data yang dilakukan selanjutnya dapat berjalan dengan lebih mudah. Setelah menentukan

direktori penyimpanan data pelatihan dan validasi, tahap selanjutnya adalah memuat data tersebut ke dalam program untuk diproses lebih lanjut. Fungsi *ImageDataGenerator* memungkinkan pemuatan gambar secara otomatis dari direktori, sekaligus melakukan beberapa tahap pra-pemrosesan seperti normalisasi piksel dan pengubahan ukuran gambar. Pada implementasi di Kode 4.7 fungsi *load_images_from_directory()* digunakan untuk memuat gambar-gambar dari sebuah direktori.

Kode Program 4.7 *Image Data Generator*

```
1  def load_images_from_directory(directory, Batch_size,
2  target_size):
3
4     datagen = ImageDataGenerator(rescale=1./255)
5     return datagen.flow_from_directory(
6        directory,
7        target_size=target_size,
8        Batch_size=Batch_size,
9        color_mode='grayscale',
10       class_mode=None,
11       shuffle=True,
12       seed=42
13    )
```

Output:

```
Found 2082 images belonging to 3 classes.
Found 523 images belonging to 3 classes.
```

Fungsi *load_images_from_directory*() pada Kode 4.7 bertujuan untuk memuat gambar dari suatu direktori dalam *Batch* dengan ukuran yang ditentukan, sekaligus melakukan pra-pemrosesan seperti *rescaling*. Di dalam fungsi, *ImageDataGenerator* digunakan untuk menormalisasi nilai piksel gambar dengan membagi setiap nilai piksel dengan 255

(*rescale*=1./255). Metode *flow_from_directory* dari *ImageDataGenerator* membaca gambar dari direktori yang diberikan, mengubah ukurannya sesuai dengan *target_size*, dan mengatur ukuran *Batch* menggunakan *Batch_size*. Gambar diproses dalam skala abu-abu (*color_mode*='*grayscale*') tanpa label kelas (*class_mode*=*None*), dengan pengacakan diaktifkan (*shuffle*=*True*) dan seed tetap (seed=42) untuk hasil yang konsisten. Fungsi ini menghasilkan *Batch* data gambar yang siap digunakan untuk pelatihan atau pengujian model.

Pentingnya keseimbangan kelas

Keseimbangan kelas (*class balance*) pada model *Convolutional Autoencoder* (CAE) untuk reduksi *noise* citra MRI menjadi penting terutama jika dataset Anda mencakup tiga bidang irisan (sagittal, coronal, dan axial). Meskipun CAE bersifat *unsupervised* dalam konteks reduksi *noise* (tidak meng-klasifikasikan label), distribusi yang tidak seimbang antar-bidang irisan tetap dapat memengaruhi performa model, karena:

Representasi Fitur yang Adil

Setiap bidang irisan (*sagittal, coronal, axial*) memiliki pola dan struktur anatomi yang berbeda. Jika satu bidang mendominasi jumlah data (misalnya sagittal jauh lebih banyak), CAE akan lebih "terbiasa" mempelajari pola pada bidang tersebut, sehingga kemampuannya menghilangkan *noise* pada bidang yang jarang muncul menjadi lebih buruk.

Generalitas Model

Model yang dilatih dengan proporsi seimbang akan mempelajari fitur spasial dan tekstur dari semua bidang secara proporsional. Ini penting untuk memastikan model bekerja baik pada seluruh variasi irisan otak, bukan hanya bidang yang paling sering dilihat saat pelatihan.

Kualitas Rekonstruksi yang Konsisten

Pada reduksi *noise*, tujuan CAE adalah menghasilkan rekonstruksi yang mempertahankan detail anatomi sambil menghilangkan gangguan. Jika kelas (bidang irisan) tidak seimbang, kualitas rekonstruksi antar-bidang akan berbeda bidang yang jarang dilatih berisiko kehilangan detail atau menghasilkan artefak.

Mencegah *Bias Latent Space*

CAE membangun representasi laten (*latent space*) dari citra. Ketidakseimbangan data dapat membuat ruang laten lebih "berorientasi" pada bidang dominan, sehingga pemetaan citra dari bidang minor menjadi kurang akurat, yang akhirnya menurunkan efektivitas *denoising*.

BAB VI
PELATIHAN DAN EVALUASI MODEL

➢ Strategi Pelatihan Model CAE

Arsitektur model CAE yang ada pada Kode 4.13 terdiri dari dua bagian utama, yaitu *encoder* dan *decoder*, yang dirancang untuk menghasilkan rekonstruksi citra setelah mengalami proses *encoding* dan *decoding*. Proses ini bertujuan untuk menghilangkan *noise* atau gangguan pada citra *input*. Bagian *Encoder* dimulai dengan lapisan *input* yang memiliki ukuran sesuai dengan citra target yang akan diproses dalam bentuk citra *grayscale*. Citra kemudian diproses melalui tiga lapisan konvolusi berturut-turut. Pada setiap lapisan konvolusi pertama, jumlah *filter* yang digunakan adalah 64, diikuti dengan lapisan *BatchNormalization* untuk menstabilkan pembelajaran dan mengurangi *overfitting*. Lapisan *MaxPooling2D* dengan ukuran *pool* (2, 2) digunakan setelah setiap lapisan konvolusi untuk mengurangi dimensi spasial citra. Setelah lapisan pertama, model melanjutkan dengan dua lapisan konvolusi tambahan dengan *filter* 128 dan 256, masing-masing dilanjutkan dengan *BatchNormalization* dan diakhiri dengan *MaxPooling2D*, yang menghasilkan representasi fitur yang lebih padat dengan dimensi akhir 32x32.

Kode Program 4.13 Arsitektur *Autoencoder*

```
# Membangun Model Autoencoder
input_img = Input(shape=(target_size[0], target_size[1], 1))
x = Conv2D(64, (3, 3), padding='same')(input_img)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(128, (3, 3), activation='ReLU', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(256, (3, 3), activation='ReLU', padding='same')(x)
x = BatchNormalization()(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)
# Decoder
x = Conv2DTranspose(256, (3, 3), activation='ReLU', padding='same')(encoded)
x = BatchNormalization()(x)
x = Conv2DTranspose(128, (3, 3), activation='ReLU', padding='same')(x)
x = BatchNormalization()(x)
x = Conv2DTranspose(64, (3, 3), activation='ReLU', padding='same')(x)
x = BatchNormalization()(x)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)
Autoencoder = Model(input_img, decoded)
Autoencoder.compile(optimizer='adam', Loss='mse')
Autoencoder.summary()
```

Bagian *decoder* yang terdapat pada Kode 4.13 dimulai dengan lapisan *Conv2DTranspose,* yang berfungsi untuk memperbesar dimensi spasial citra yang telah terkompresi dalam ruang laten. Lapisan pertama menggunakan 256 *filter* dengan ukuran *kernel* 3x3 dan *padding 'same',* diikuti oleh *BatchNormalization* untuk normalisasi. Proses ini dilanjutkan dengan dua lapisan *Conv2DTranspose* berikutnya dengan

jumlah *filter* yang berkurang, yaitu 128 dan 64, masing-masing diikuti oleh *BatchNormalization* untuk meningkatkan stabilitas model. Setiap lapisan transposisi konvolusi menggunakan *ReLU* sebagai fungsi aktivasi, yang membantu dalam rekonstruksi citra. Lapisan Conv2D dengan satu filter dan aktivasi sigmoid digunakan setelah lapisan transposisi konvolusi terakhir untuk menghasilkan citra *output* grayscale yang berukuran sama dengan *input*, dengan nilai piksel dalam rentang [0,1]. Lalu model dikompilasi dengan *optimizer* ADAM dan fungsi *Loss* MSE. Tujuan dari model ini adalah untuk mempelajari citra *input* dan mengembalikannya menjadi citra yang telah diperbaiki.

- ***Hyperparameter Tuning***

Pelatihan model dilakukan dengan menggunakan data yang telah diberikan *noise* sebagai *input*, sementara citra aslinya digunakan sebagai *target* untuk proses pembelajaran. Model CAE dilatih selama beberapa *Epoch* dengan menggunakan *Batch size* yang telah ditentukan untuk mempercepat proses konvergensi. Selama proses pelatihan, model meminimalkan *Loss* yang diukur menggunakan MSE antara citra yang direkonstruksi dan citra aslinya.

Tabel 2. Parameter *Compile* Model

No	Jenis Parameter	Nilai
1	*Optimizer*	*Adam*
2	*Loss*	*MSE*
3	*Epochs*	20
4	*Batch Size*	8

Serangkaian uji coba dilakukan untuk mendapatkan nilai *hyparameter* pelatihan yang paling optimal untuk digunakan dalam melatih model CAE yang dapat dilihat pada Tabel 2

Nilai-nilai *hyperparameter* yang akan diuji coba berdasarkan beberapa pembahasan-pembahasan terdahulu yang melatih model CAE untuk melakukan *denoising* pada citra. Uji Coba meliputi *Epoch*, *Batch Size*, dan *Optimizer*. Dalam menentukan *Epoch*, uji coba dengan 15, 20, dan 25 *Epoch* dilakukan dan ditemukan *Epoch* 20 memberikan hasil yang paling baik diantara ketiga nilai. Lalu dalam menentukan nilai *Batch Size*, uji coba dengan nilai 4, 8, dan 16 dilakukan dan ditemukan *Batch size* dengan nilai 8 memberikan hasil yang paling baik dari ketiga nilai yang di uji coba. Ujicoba *optimizer* juga dilakukan antara SGD dan ADAM, *optimizer* ADAM memberikan hasil yang jauh lebih baik.

Pada Tabel 2 dapat dilihat model akan di-*compile* dengan beberapa nilai parameter, diantaranya ADAM untuk *optimizer*, MSE untuk metrik evaluasi, dengan 20 kali perulangan dan 8 Batch size. Penggunaan *Optimizer* ADAM dapat mengatasi keterbatasan yang disebabkan oleh variabilitas *noise* pada gambar medis, karena tingkat pembelajaran adaptif ADAM dapat membantu CAE menyesuaikan diri dengan berbagai tingkat dan jenis *noise*, sehingga meningkatkan kinerja denosing *noise* [32]. MSE adalah fungsi *Loss* yang sederhana dan efisien secara komputasi yang umumnya digunakan dalam pelatihan tugas-tugas *denoising*. Fungsi ini secara efektif mengukur perbedaan kuadrat rata-rata antara nilai piksel yang diprediksi dan yang sebenarnya, sehingga cocok untuk kinerja denosing *noise* [33]. *Epoch* yang dapat digunakan sebanyak 20 dan Batch size 8. Pada pembahasan *denoising* menggunakan *Autoencoder* [34] setelah membandingkan hasil yang di-dapatkan ditemukan bahwa *Epoch* dengan nilai 25 sangat cocok terhadap pembahasan tersebut dengan hasil nilai PSNR di atas 35 dB.

➢ Skema Pengujian dengan Variasi *Noise*

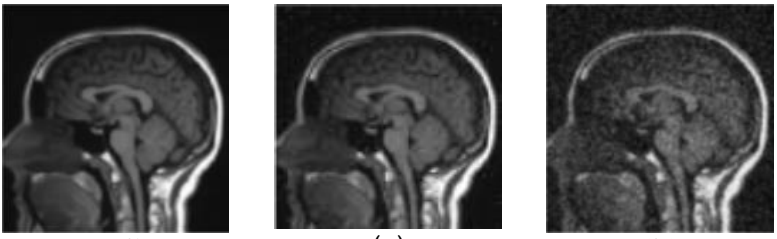
Pengujian model dilakukan untuk memastikan bahwa CAE yang dikembangkan dapat men-*denoise* citra MRI otak. Skenario pengujian menggunakan metrik seperti PSNR dan MSE. Model diuji menggunakan data validasi yang berisi citra MRI otak dengan *noise*, dan hasil rekonstruksinya dibandingkan dengan citra MRI asli.

Tabel 3. Skenario Pengujian

Metode	Parameter	Varian	Metrik Evaluasi	
Convolutional Autoencoder	*Speckle*	0,025	MSE	PSNR
		0,05		
		0,075		
		0,1		
		0,125		
		0,15		
		0,175		
		0,2		
	Salt	0,025		
		0,05		
		0,075		
		0,1		
		0,125		
		0,15		
		0,175		
	Salt	0,2		
	Pepper	0,025	MSE	PSNR
		0,05		
		0,075		
		0,1		
		0,125		
		0,15		

Metode	Parameter	Varian	Metrik Evaluasi	
Convolutional Autoencoder		0,175		
		0,2		
	Salt & *Pepper*	0,025		
		0,05		
		0,075		
		0,1		
		0,125		
		0,15		
		0,175		
		0,2		

Pada Tabel 3 terlihat skenario pengujian dimana model CAE diuji dengan empat jenis *noise* yang berbeda, yaitu *Speckle Noise*, *Salt Noise*, *Pepper Noise*, *Salt* & *Pepper Noise*. Setiap jenis *noise* dimasukan pada citra MRI, setiap metode diuji dengan tiga nilai parameter yang berbeda, yaitu 0,025; 0,05; 0,075; 0,1; 0,125; 0,15; 0,175; dan 0,2. Parameter ini mewakili tingkat *noise* atau gangguan yang diterapkan pada data atau gambar yang diuji. Dalam pembahasan terdahulu nilai-nilai varians tersebut menunjukkan proporsi piksel yang terpengaruh oleh korupsi *noise*, berkisar antara 2,5%; 5%; 7,5%; 10%; 12,5%; 15%; 17,5%; dan 20% dari total piksel citra [46]. Untuk *Noise Speckle*, parameter tersebut merepresentasikan intensitas *noise* multiplikatif yang diterapkan pada gambar asli [5].



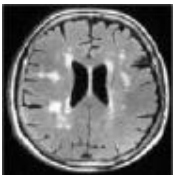
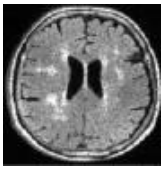
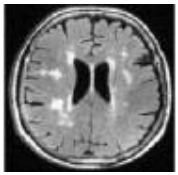
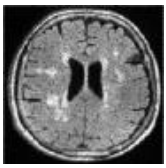
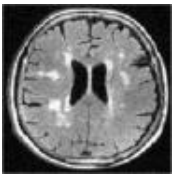
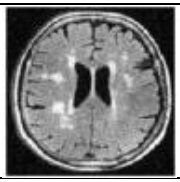
Asli (a) (b)

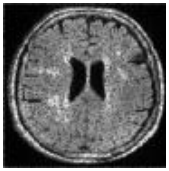
Gambar 41. Varian *Noise*

Rentang 0,025; 0,05; 0,075; 0,1; 0,125; 0,15; 0,175; dan 0,2 cukup mewakili kondisi varian *noise* tanpa menyebabkan citra medis terlalu rusak sehingga tidak relevan lagi untuk diagnosis, seperti yang digunakan pada pembahasan-pembahasan *denoising* MRI lainnya [5] [46]. *Noise* dengan varian 0,01 seperti pada Gambar 41 (a) tidak digunakan dikarenakan *noise* yang terlihat tidak signifikan dan untuk *noise* varian di atas 0,2 tidak digunakan karena citra dinilai terlalu rusak sehingga ketika di rekonstruksi ulang terlalu banyak detail penting yang hilang yang dapat dilihat pada Gambar 41 (b).

Pada, pengujian dilakukan dengan menggunakan empat varian *noise* yang disimulasikan ke dalam citra, yaitu *Salt*, *Pepper*, kombinasi *Salt* dan *Pepper*, serta *Speckle*. Setiap jenis *noise* diterapkan dengan tiga tingkat intensitas yang berbeda, yaitu (0,025; 0,05; 0,075; 0,1; 0,125; 0,15; 0,175; dan 0,2) dapat dilihat pada Tabel 4 Variasi ini bertujuan untuk mengevaluasi performa model *denoising* dalam menghadapi beragam kondisi degradasi citra MRI, mencakup tiga jenis gangguan utama yaitu Gaussian *Noise*, Salt & Pepper *Noise*, serta *Speckle Noise* yang secara umum sering ditemukan pada proses akuisisi citra medis.

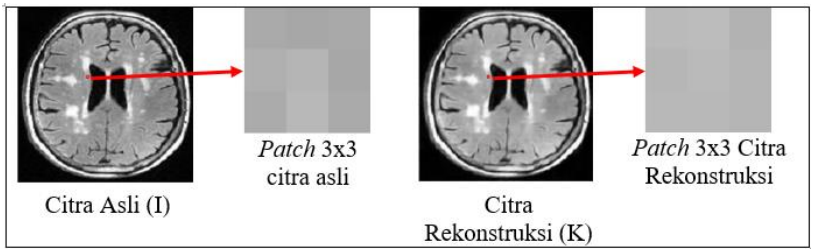
Tabel 4. *Sample* Varian *Noise*

Varian *Noise*	Jenis *Noise*			
	Salt	*Pepper*	*Salt* & *Pepper*	*Speckle*
0,025				
0,05				
0,075				
0,1				
0,125				
0,15				

Varian *Noise*	Jenis *Noise*			
	Salt	*Pepper*	*Salt & Pepper*	*Speckle*
0,175				
0,2				

➢ Evaluasi Performa dengan MSE dan PSNR

Model dilatih untuk merekonstruksi citra bersih dari versi yang telah diberi *noise*. Kualitas rekonstruksi hasil model dinilai menggunakan dua metrik, yaitu MSE dan PSNR, untuk setiap jenis *noise*. MSE dapat dihitung dengan menjumlahkan kuadrat selisih antara setiap piksel dari kedua citra, kemudian membaginya dengan jumlah total piksel dalam citra. Misalkan pada Gambar 42 diambil *patch* piksel 3x3 dari citra "*N_109.jpg*" dengan koordinat x (baris) sama dengan 100 hingga 103 dan y (kolom) sama dengan 100 hingga 103, dimana citra sudah diubah menjadi *grayscale* dan dinormalisasi ke rentang 0 - 1, yaitu citra asli lalu dihitung dengan citra hasil *denoisingnya* menggunakan persamaan (2.3).



Gambar 42. Perhitungan MSE dan PSNR

$$\text{Citra (I)} : \begin{bmatrix} 0{,}667 & 0{,}655 & 0{,}663 \\ 0{,}690 & 0{,}710 & 0{,}655 \\ 0{,}671 & 0{,}722 & 0{,}667 \end{bmatrix}$$

$$\text{Citra (K)} : \begin{bmatrix} 0{,}727 & 0{,}735 & 0{,}699 \\ 0{,}714 & 0{,}727 & 0{,}711 \\ 0{,}721 & 0{,}721 & 0{,}711 \end{bmatrix}$$

Langkah pertama berdasarkan persamaan 2.2 yaitu dengan menghitung selisih antara setiap piksel dengan posisi yang sama antara citra asli (l) dan citra hasil *denoising* (K) dan mengkuadratkan hasilnya.

$$\begin{aligned}
i=1, j=1 &: (0{,}667-0{,}727)^2 = (-0{,}060)^2 = 0{,}0036 \\
i=1, j=2 &: (0{,}655-0{,}735)^2 = (-0{,}080)^2 = 0{,}0064 \\
i=1, j=3 &: (0{,}663-0{,}699)^2 = (-0{,}036)^2 = 0{,}001296 \\
i=2, j=1 &: (0{,}690-0{,}714)^2 = (-0{,}024)^2 = 0{,}000576 \\
i=2, j=2 &: (0{,}710-0{,}727)^2 = (-0{,}017)^2 = 0{,}000289 \\
i=2, j=3 &: (0{,}655-0{,}711)^2 = (-0{,}056)^2 = 0{,}003136 \\
i=3, j=1 &: (0{,}671-0{,}721)^2 = (-0{,}050)^2 = 0{,}0025 \\
i=3, j=2 &: (0{,}722-0{,}721)^2 = (0{,}001)^2 = 0{,}000001 \\
i=3, j=3 &: (0{,}667-0{,}711)^2 = (-0{,}044)^2 = 0{,}001936
\end{aligned}$$

Langkah selanjutnya jumlahkan semua kuadrat selisih :

$$0{,}0036 + 0{,}0064 + 0{,}001296 + 0{,}000576 + 0{,}000289 + 0{,}003136 + 0{,}0025 + 0{,}000001 + 0{,}001936 = 0{,}019734$$

Bagi dengan jumlah total piksel dalam citra. Karena citra ini berukuran 3x3, maka jumlah piksel adalah 9.

$$MSE = \frac{0{,}019734}{9} = 0{,}002193$$

Setelah memperoleh nilai MSE, langkah berikutnya adalah menentukan data range, yang dalam citra pada pembahasan ini bernilai 1 karena citra telah dinormalisasi ke rentang 0-1. Selanjutnya, PSNR dihitung menggunakan persamaan (2.3), persamaan logaritmik berbasis 10 yang membandingkan kuadrat data range dengan MSE. Perhitungan ini bertujuan untuk mengekspresikan rasio antara sinyal dan *Noise* dalam satuan desibel (dB).

$$PSNR = 10 \times log_{10}\left(\frac{(1)^2}{0{,}002193}\right)$$

$$PSNR = 10 \times log_{10}\left(\frac{1}{0{,}002193}\right)$$

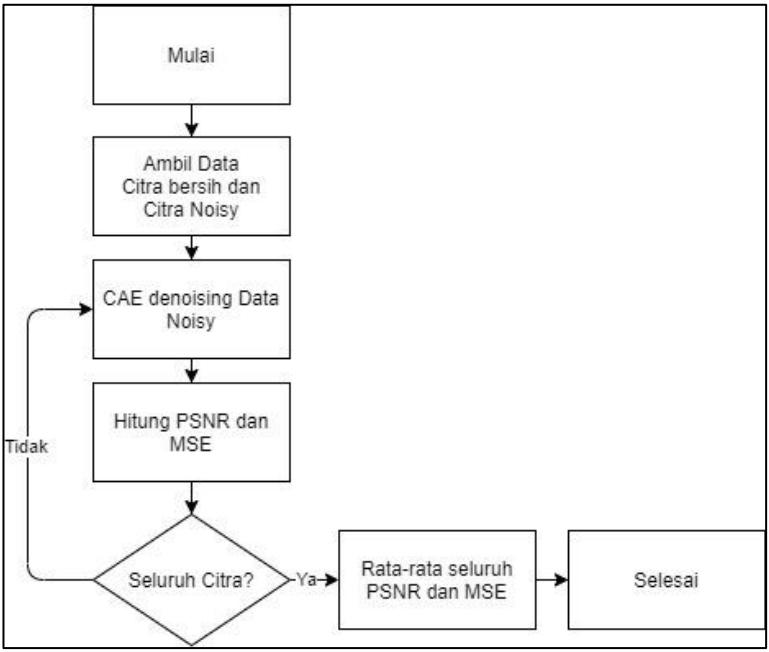
$$PSNR = 10 \times log_{10}(445{,}79)$$

$$PSNR = 10 \times 2{,}65$$

$$PSNR = 26{,}5 \text{ dB}$$

Setelah menghitung dengan persamaan PSNR, hasilnya adalah 26,5 dB. MSE mengukur rata-rata kesalahan kuadrat antara dua citra yaitu citra (I) dan citra yang dihasilkan (K). Semakin kecil nilai MSE, semakin sedikit perbedaan antara citra asli dan citra yang diproses oleh model, yang berarti model

menghasilkan citra yang lebih mirip dengan citra asli. Nilai PSNR yang lebih tinggi menunjukkan bahwa citra hasil rekonstruksi atau citra. Logaritma dalam persamaan PSNR memperkuat dampak dari perubahan MSE. Semakin kecil MSE, nilai dalam logaritma akan meningkat, yang menyebabkan PSNR menjadi lebih tinggi. Kedua metrik ini membantu mengukur seberapa baik model dapat menghilangkan masing-masing jenis *noise*, sehingga dapat dinilai kekuatan dan kelemahan CAE dalam menangani *noise* tertentu. Model CAE yang efektif dalam menghilangkan *noise* akan menunjukkan MSE rendah dan PSNR tinggi, yang menandakan bahwa citra hasil model sangat mendekati citra asli dan *noise* telah diminimalkan.



Gambar 43. Alur Evaluasi Performa

Gambar 43 merupakan diagram alur evaluasi performa model *denoising* berbasis CAE dengan menggunakan metrik PSNR dan MSE. Evaluasi dimulai dengan mengambil data citra bersih dan citra ber*noise*, kemudian citra ber*noise* diproses menggunakan model CAE untuk menghasilkan citra hasil *denoising*. Setelah itu, dilakukan perhitungan nilai PSNR dan MSE untuk setiap pasang citra bersih dan noisy. Proses ini dilakukan secara berulang hingga seluruh citra dalam dataset selesai diproses. Selanjutnya, seluruh nilai PSNR dan MSE yang telah dihitung akan dirata-ratakan untuk memperoleh nilai performa akhir dari model.

Evaluasi performa pada CAE penting dilakukan untuk mengukur sejauh mana model mampu merekonstruksi citra *input* secara akurat, terutama setelah mengalami gangguan seperti *noise*. Dengan menggunakan metrik evaluasi seperti MSE dan PSNR, dengan melihat hasil dari metrik evaluasi performa model dapat dinilai tingkat kesalahan rekonstruksi dan kualitas visual yang dihasilkan.

Kode Program 4.14 Evaluasi Performa

```
1  def evaluate_model(Autoencoder, generator, steps):
2      mse_values = []
3      psnr_values = []
4
5      for _ in range(steps):
6          noisy_Batch, clean_Batch = next(generator)
7          predicted_Batch = Autoencoder.predict(noisy_Batch)
8          for i in range(len(clean_Batch)):
9              clean_image = clean_Batch[i].flatten()
10             predicted_image = predicted_Batch[i].flatten()
11             mse_value = mse(clean_image, predicted_image)
12             psnr_value = psnr(clean_image, predicted_image,
13 data_range=1.0)
```

Kode Program 4.14 Evaluasi Performa

```
            mse_values.append(mse_value)
            psnr_values.append(psnr_value)

    avg_mse = np.mean(mse_values)
    avg_psnr = np.mean(psnr_values)

    return avg_mse, avg_psnr

# Evaluate on validation data
validation_steps = len(validation_generator)
avg_mse, avg_psnr = evaluate_model(Autoencoder,
noisy_validation_generator, validation_steps)
print(f"Average MSE: {avg_mse}")
print(f"Average PSNR: {avg_psnr} dB")
```

Output :

```
...
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
Average MSE: 0.0011215686099603772
Average PSNR: 29.92626413048818 dB
```

Kode program 4.14 bertujuan untuk mengevaluasi kinerja model *Autoencoder* dalam melakukan *denoising* citra dengan menggunakan metrik kuantitatif yaitu MSE dan PSNR. Fungsi *evaluate_model* menerima tiga parameter utama, yaitu model *Autoencoder* yang telah dilatih, *generator* data citra bising dan bersih, serta jumlah langkah evaluasi. Untuk setiap langkah, fungsi akan mengambil sepasang *Batch data* bersih dan bising (*noisy*), melakukan prediksi menggunakan *Autoencoder*, kemudian menghitung nilai-nilai MSE dan PSNR dan dirata-ratakan untuk memberikan gambaran menyeluruh terhadap performa model pada data validasi. Penggunaan metrik PSNR memberikan informasi tentang kualitas visual rekonstruksi, sedangkan MSE mengukur kesalahan numerik antara citra prediksi dan citra asli. Hasil evaluasi ditampilkan dalam bentuk nilai rata-rata MSE dan PSNR untuk seluruh *Batch* pada data validasi seperti pada contoh *output*.

BAB VII
DENOISING, NOISE DAN KURVA *LOSS*

➢ Evaluasi PSNR dan MSE

Evaluasi kuantitatif terhadap kinerja *denoising* model mengungkap pola yang berbeda-beda di antara berbagai jenis *Noise* seperti *Salt*, *Pepper*, *Salt* & *Pepper*, dan *Speckle* serta tingkat intensitasnya yaitu (0,025; 0,05; 0,075; 0,1; 0,125; 0,15; 0,175; dan 0,2). Analisis ini didasarkan pada dua metrik yaitu PSNR yang diukur dalam desibel (dB) dan MSE yang ditunjukkan pada Tabel 4 dan Tabel 6 Hasil evaluasi menunjukkan bahwa kinerja model cenderung menurun seiring meningkatnya intensitas *noise*.

Tabel 5. Evaluasi PSNR model CAE

Metrik Evaluasi	Varian *Noise*	Jenis *Noise*			
		Salt	*Pepper*	*Salt* & *Pepper*	*Speckle*
PSNR (dB)	0,025	29,9262	31,1651	29,9518	30,5091
	0,05	28,2765	30,7346	28,8030	28,4495
	0,075	28,8862	30,7521	29,1296	27,7341
	0,1	27,5998	29,4030	27,7834	27,6893
	0,125	28,6794	30,4348	28,3477	27,7458
	0,15	28,1191	28,9051	28,7272	26,4614
	0,175	27,5775	28,1476	28,2832	25,8317
	0,2	27,3503	27,9315	27,8364	25,8930

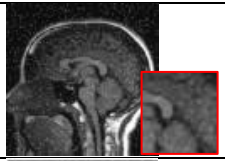
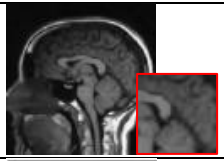
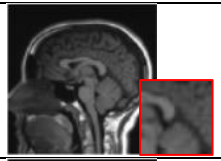
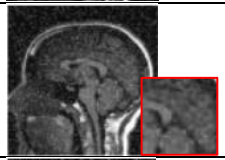
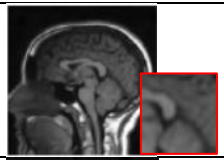
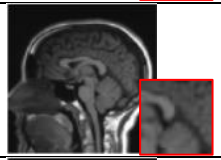
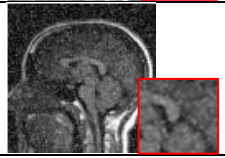
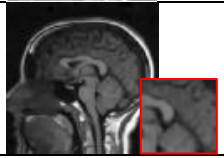
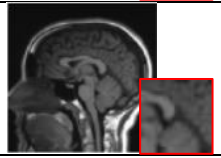
Tabel 6. Evaluasi MSE model CAE

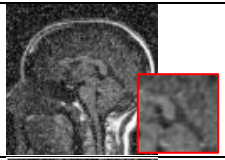
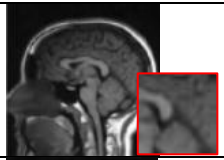
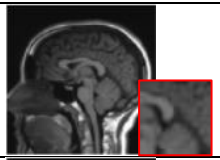
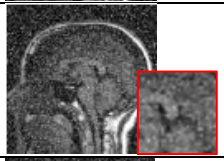
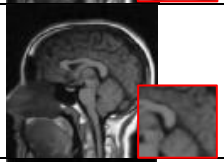
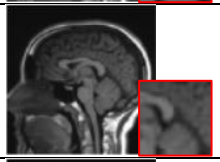
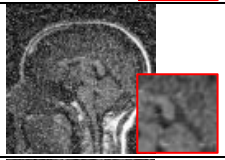
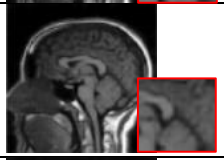
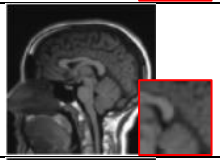
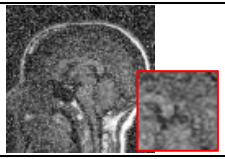
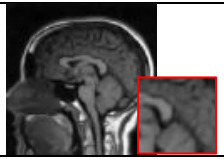
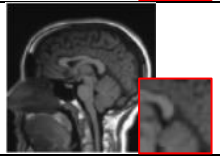
Metrik Evaluasi	Varian *Noise*	Jenis *Noise*			
		Salt	*Pepper*	*Salt & Pepper*	*Speckle*
MSE	0,025	0,0011215	0,0008604	0,0011086	0,0009864
	0,05	0,0016673	0,0009467	0,0015677	0,0016020
	0,075	0,0014397	0,0009485	0,0013337	0,0017770
	0,1	0,0019890	0,0012410	0,0018213	0,0018411
	0,125	0,0014924	0,0010262	0,0015897	0,0017837
	0,15	0,0016260	0,0013972	0,0014664	0,0024591
	0,175	0,0019489	0,0016852	0,0016371	0,0027707
	0,2	0,0019931	0,0017866	0,0017607	0,0027850

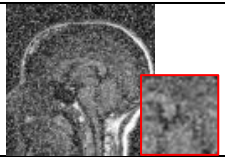
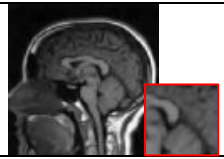
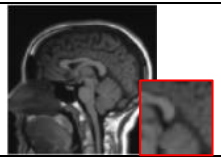
➢ Hasil *Denoising Salt*

Seperti yang ditunjukkan pada Tabel 7, model CAE menunjukkan performa yang cukup baik dalam mengeliminasi *noise* bertipe *Salt*. Meskipun terjadi sedikit fluktuasi pada nilai PSNR, secara keseluruhan PSNR tetap berada pada kisaran tinggi, yaitu antara 29,61 dB hingga 27,35 dB. Karateristik *Noise Salt* yang bersifat impulsif yang memungkinkan model dengan cepat mengenali *noise*.

Tabel 7. Hasil *Denoising* Noise Salt

Varian *Noise*	Performa Rekonstruksi		Citra		
	MSE	PSNR (dB)	*Noisy*	Rekonstruksi	Asli
0,025	0,00112	29,926			
0,05	0,00167	28,276			
0,075	0,00144	28,886			

Varian *Noise*	Performa Rekonstruksi		Citra		
	MSE	PSNR (dB)	*Noisy*	Rekonstruksi	Asli
0,1	0,00199	27,599			
0,125	0,00149	28,679			
0,15	0,00163	28,119			
0,175	0,00195	27,577			

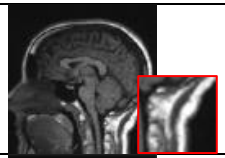
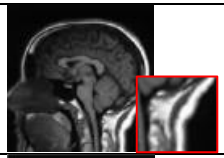
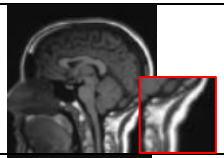
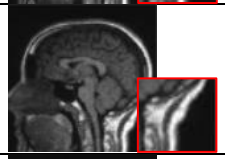
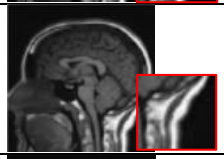
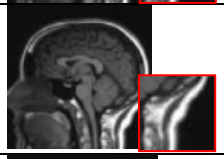
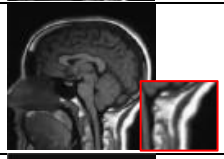
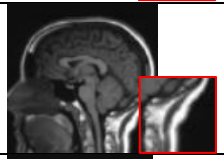
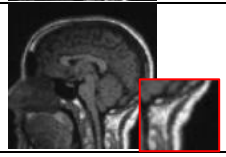
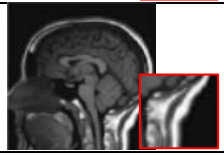
Varian *Noise*	Performa Rekonstruksi		Citra		
	MSE	PSNR (dB)	*Noisy*	Rekonstruksi	Asli
0,2	0,00199	27,350			

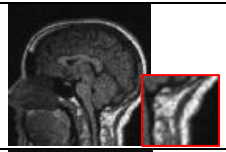
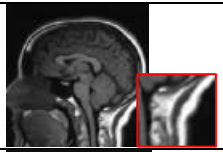
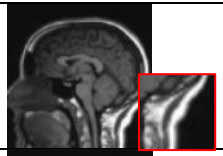
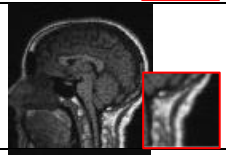
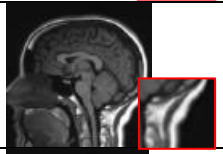
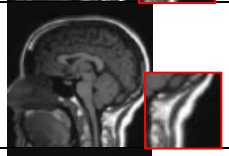
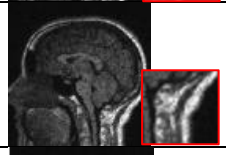
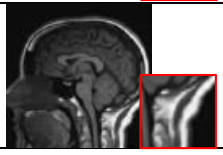
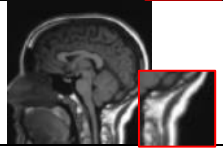
Berdasarkan hasil yang diperbesar dan ditampilkan pada Tabel 7, model CAE menunjukkan performa yang konsisten dalam mereduksi *noise* bertipe *Salt* di kedelapan tingkat varians. Pada varian *noise* sebesar 0,025 memiliki gangguan pada citra yang tidak terlalu signifikan, sehingga model mampu melakukan rekonstruksi dengan tingkat ketelitian yang tinggi, terlihat dari kemiripan hasil rekonstruksi dengan citra asli. Seiring dengan meningkatnya varians, hasil tidak mengalami degradasi yang signifikan. Pada tingkat varians tertinggi, yaitu 0,2 dimana penyebaran *noise* berdampak lebih besar, hasil rekonstruksi tetap menunjukkan bahwa model mampu mempertahankan bentuk citra asli. Dapat disimpulkan bahwa model CAE memiliki kapabilitas yang baik dalam mengatasi *noise* bertipe *Salt* hingga tingkat korupsi 20% yang terjadi pada citra.

- **Hasil *Denoising Pepper***

Pada baris Pepper di Tabel 8, model menunjukkan performa model tercatat paling optimal di antara semua jenis *noise* yang diuji. Nilai PSNR tertinggi diperoleh pada intensitas 0,025 sebesar 31,16 dB, yang mengindikasikan kualitas rekonstruksi yang baik, dengan nilai MSE paling rendah yaitu 0,0008604. Saat intensitas meningkat ke 0,1 dan 0,125 PSNR yang dihasilkan tetap tinggi 29,4030 dB dan 30,43 dB, sementara MSE hanya meningkat sedikit menjadi 0,0012411 dan 0,0010262. Ini menunjukkan bahwa model mampu mengidentifikasi dan menghapus *noise* Pepper dibandingkan dengan ketiga jenis *noise* lainnya.

Tabel 8. Hasil *Denoising Noise* Pepper

Varian *Noise*	Performa Rekonstruksi		Citra		
	MSE	PSNR *(dB)*	*Noisy*	Rekonstruksi	Asli
0,025	0,00086	31,165			
0,05	0,00095	30,734			
0,075	0,00095	30,752			
0,1	0,00124	29,403			

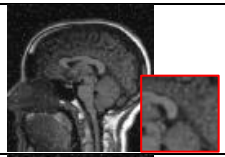
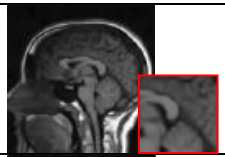
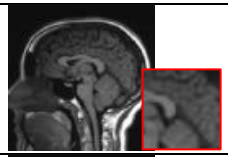
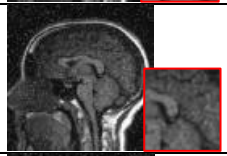
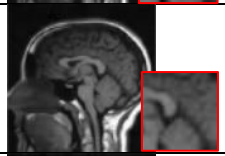
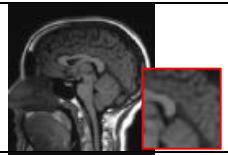
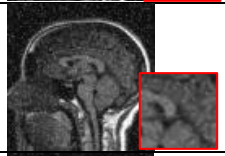
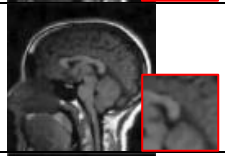
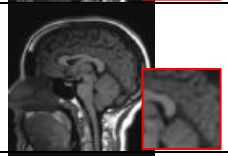
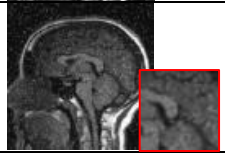
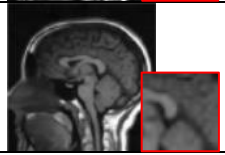
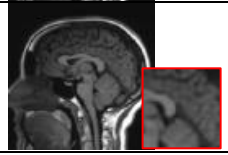
Varian *Noise*	Performa Rekonstruksi		Citra		
	MSE	PSNR *(dB)*	*Noisy*	Rekonstruksi	Asli
0,125	0,00103	30,434			
0,15	0,00140	28,905			
0,175	0,00169	28,147			
0,2	0,00179	27,931			

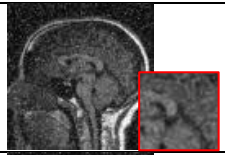
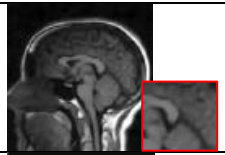
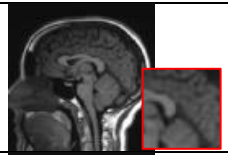
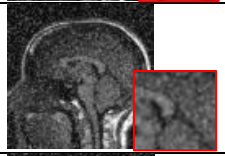
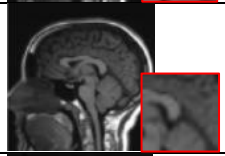
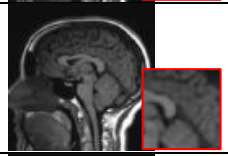
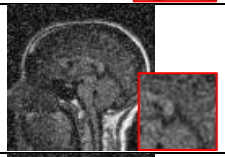
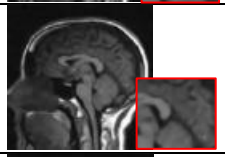
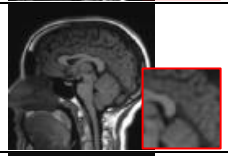
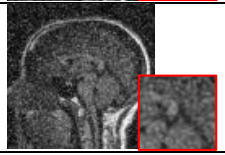
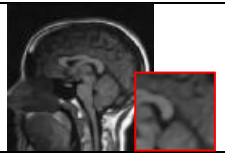
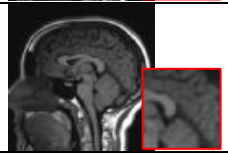
Berdasarkan Tabel 8 yang menampilkan hasil diperbesar dari *denoising Noise Pepper* pada citra MRI dengan lima tingkat varians berbeda, terlihat bahwa model CAE berhasil melakukan rekonstruksi citra dengan sangat baik pada seluruh tingkat *noise*. Pada varians 0,025 citra rekonstruksi menunjukkan kemiripan yang baik dengan citra asli secara visual. Seiring meningkatnya varians meskipun jumlah *noise* pada citra *noisy* terlihat semakin signifikan, model tetap mampu memulihkan detail citra tanpa menghasilkan artefak yang signifikan.

- **Hasil *Denoising Salt & Pepper***

Model CAE mencapai kinerja terbaik secara keseluruhan dalam menangani *noise* kombinasi *Salt* dan *Pepper*, seperti yang ditunjukkan pada Tabel 9. Terlihat dari nilai PSNR yang lebih rendah dibanding ketika menangani masing-masing *noise* secara terpisah. Pada intensitas 0,025 PSNR tercatat sebesar 29,95 dB dengan MSE sebesar 0,0011086 dan performa terendah terdapat pada intensitas 0,2 dengan PSNR sebesar 27,7834dB dengan MSE 0,0017607.

Tabel 9. Hasil *Denoising* Noise Salt & Pepper

Varian *Noise*	Performa Rekonstruksi		Citra		
	MSE	PSNR *(dB)*	*Noisy*	Rekonstruksi	Asli
0,025	0,00111	29,951			
0,05	0,00157	28,803			
0,075	0,00133	29,129			
0,1	0,00182	27,783			

Varian *Noise*	Performa Rekonstruksi		Citra		
	MSE	PSNR *(dB)*	*Noisy*	Rekonstruksi	Asli
0,125	0,00159	28,347			
0,15	0,00147	28,727			
0,175	0,00164	28,283			
0,2	0,00176	27,836			

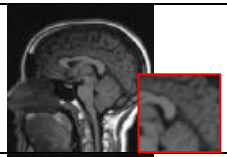
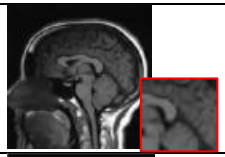
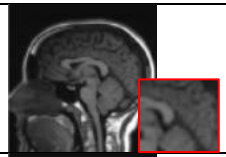
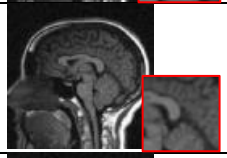
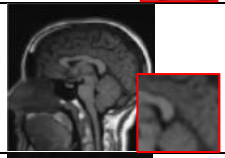
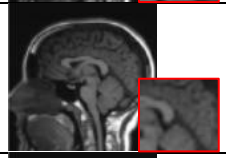
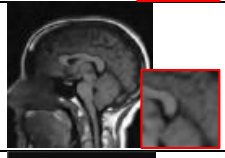
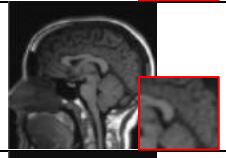
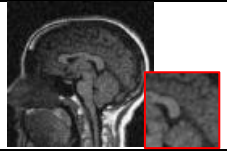
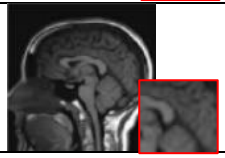
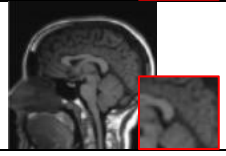
Hasil menunjukkan bahwa kombinasi dua jenis *noise* memperberat proses rekonstruksi citra oleh CAE. Kombinasi *noise* ini memberikan tantangan tambahan dalam proses pemulihan citra, yang terlihat dari nilai PSNR yang lebih rendah dan MSE yang lebih tinggi pada ketiga intensitas *noise* yang diuji. Berdasarkan Tabel 9 yang menyajikan hasil yang diperbesar dari *denoising* terhadap *noise* kombinasi *Salt* & *Pepper*. Pada varians 0,025 sampai 0,15 citra rekonstruksi menunjukkan kualitas visual yang baik dan menyerupai citra asli, dengan detail struktur otak yang tetap terjaga. Namun, mulai pada varian 0,175 performa model mengalami degradasi yang cukup signifikan, ditandai dengan hilangnya beberapa detail halus, dan ada *noise* yang tersisa dan masih terdapat sisa-sisa *noise* yang tidak berhasil dihilangkan sepenuhnya oleh model. Hal ini menandakan bahwa kapasitas model untuk mengeliminasi *noise* mulai mencapai batasnya pada tingkat varians ini. Sementara itu, pada varian 0,2 *noise*, model tampak mulai tidak mempertahankan detail halus utama citra dengan kualitas visual yang cukup, menjadikan varian dengan performa terburuk pada *noise* ini.

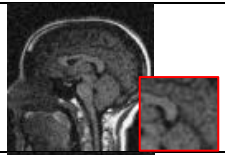
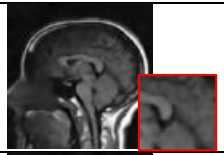
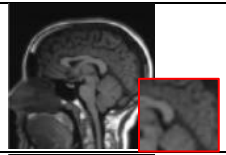
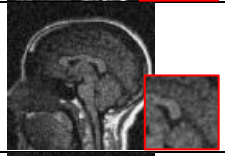
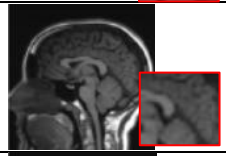
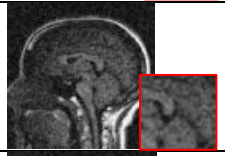
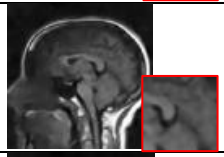
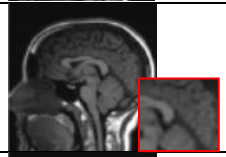
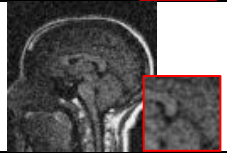
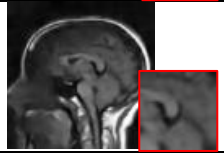
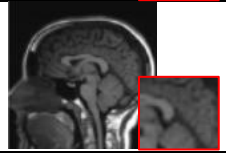
- **Hasil *Denoising Speckle***

Jenis *Noise Speckle* memberikan tingkat kesulitan terbesar bagi model terlihat pada Tabel 10. Tercatat penurunan PSNR yang cukup signifikan seiring peningkatan intensitas, mulai dari 30,5091 dB pada varian 0,025 menjadi 28,4495 dB pada varian 0,05, dan terus menurun hingga pada varian 0,175 menjadi 25,8317 dB, menjadi hasil paling buruk. Selain itu, terjadi peningkatan MSE dari 0,000986 ke 0,002770 yang merupakan nilai MSE tertinggi di antara semua kondisi. *Speckle Noise* yang bersifat multiplikatif dan menyebar secara acak di seluruh citra menyebabkan kesulitan dalam proses *denoising*, karena pola

gangguannya tidak sekadar impulsif tetapi memengaruhi intensitas piksel secara lokal. Karakteristik *Speckle noise* yang bersifat multiplikatif dan tersebar secara acak di seluruh citra, gangguan ini tidak hanya bersifat impulsif seperti pada *Noise Salt* dan *Pepper*, tetapi juga memengaruhi intensitas piksel secara lokal dan merata di seluruh area citra. Variasi ini bertujuan untuk mengevaluasi performa model *denoising* dalam menghadapi beragam kondisi degradasi citra MRI, mencakup tiga jenis gangguan utama yaitu Gaussian Noise, Salt & Pepper Noise, serta *Speckle* Noise yang secara umum sering ditemukan pada proses akuisisi citra medis.

Tabel 10. Hasil *Denoising* Noise *Speckle*

Varian *Noise*	Performa Rekonstruksi		Citra		
	MSE	PSNR *(dB)*	*Noisy*	Rekonstruksi	Asli
0,025	0,00099	30,509			
0,05	0,00160	28,449			
0,075	0,00178	27,734			
0,1	0,00184	27,689			

Varian *Noise*	Performa Rekonstruksi		Citra		
	MSE	PSNR *(dB)*	*Noisy*	Rekonstruksi	Asli
0,125	0,00178	27,745			
0,15	0,00246	26,461			
0,175	0,00277	25,831			
0,2	0,00279	25,893			

Berdasarkan Tabel 10 yang menyajikan hasil yang diperbesar dari *denoising Noise Speckle*, dapat diamati bahwa performa model CAE mengalami penurunan kualitas seiring meningkatnya intensitas *noise*. Pada varians 0,075 hasil rekonstruksi sudah mulai mengalami degradasi terhadap detail citra. Ketika intensitas *noise* meningkat ke 0,125 penurunan kualitas visual menjadi lebih jelas, dengan munculnya distorsi pada tepi dan hilangnya detail halus. Pada varians tertinggi yaitu 0,15 hasil rekonstruksi menunjukkan degradasi visual yang sangat signifikan, dengan citra yang tampak lebih buram dan kehilangan banyak detail dari MRI otak. Secara keseluruhan hasil evaluasi PSNR dan MSE untuk performa model CAE pada berbagai jenis *noise* dengan variasi tingkat *noise* 0,05; 0.1; dan 0,15 dimana nilai PSNR terbaik tercatat pada jenis *Noise Pepper* dengan varian 0,05 yaitu 30,7346 dB. Di sisi lain, nilai PSNR terendah tercatat pada jenis *Noise Speckle* dengan varian 0,15 yaitu 26,4614 dB.

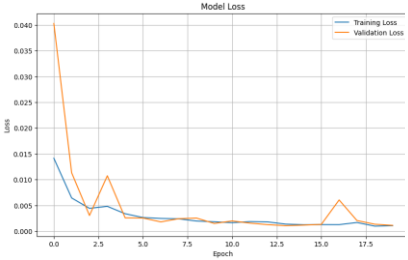
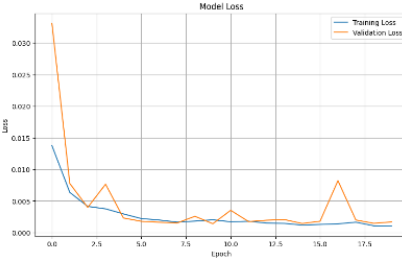
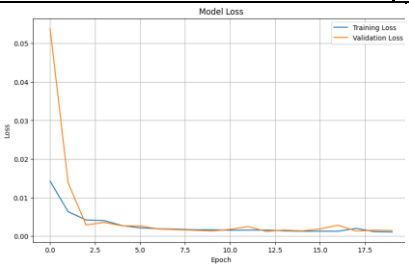
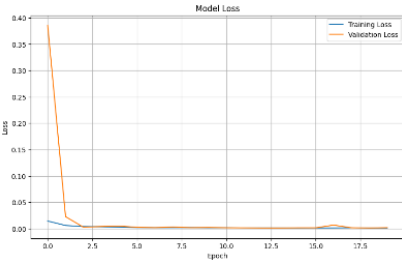
- **Evaluasi Kurva *Loss***

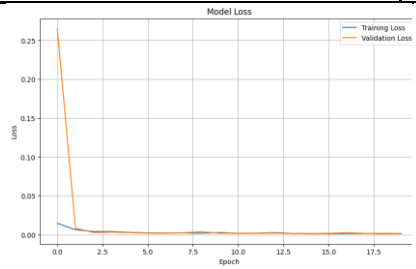
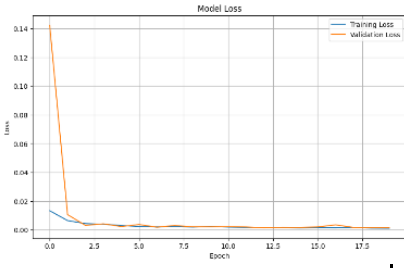
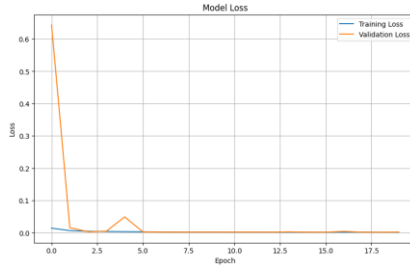
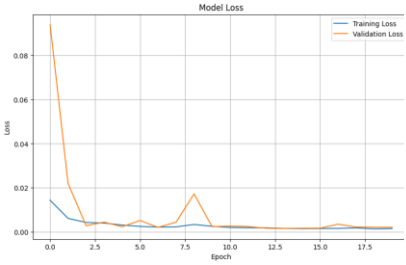
Kurva *Loss* MSE memberikan gambaran mengenai efektivitas proses pelatihan, kestabilan model, serta performa terhadap berbagai jenis *noise* yang diuji pada pembahasan ini, sehingga membantu dalam mengoptimalkan dan memvalidasi model CAE.

Kurva *Loss* MSE *Salt*

Kurva *Loss* MSE pada Gambar 44 menampilkan performa model *Autoencoder* dalam melakukan *denoising* terhadap citra MRI yang terkena *noise* jenis *Salt* dengan berbagai tingkat varian *noise*. Secara umum, kurva menunjukkan bahwa model mampu belajar dengan baik pada *noise* varian rendah, ditandai dengan penurunan tajam pada nilai *Training Loss* dan *Validation*

Loss, serta gap yang kecil di antara keduanya. Hal ini terlihat jelas pada Gambar (a) 0,025 dan Gambar (b) 0,05, di mana model mencapai nilai MSE yang rendah baik pada data pelatihan maupun validasi walaupun terdapat indikasi model sedikit *overfitting* dikarenakan kurva sempat sedikit meningkat di akhir *Epoch*. Ketika varian *noise* meningkat, pada Gambar (c) 0,075 hingga Gambar (d) 0,1 mulai nilai pada *Validation Loss* dan *Training Loss* mulai meningkat. Ini menandakan bahwa model mulai menghadapi tantangan dalam mengenali pola bersih dari citra yang terganggu oleh *noise*.

Kurva *Loss* MSE Noise Salt	
	
Train_MSE *Epoch* ke-1 : 0.0141 ... *Epoch* ke-20 : 0,0011 Val_MSE *Epoch* ke-1 : 0.0403 ... *Epoch* ke-20 : 0,0011 0,025	Train_MSE *Epoch* ke-1 : 0,0252 ... *Epoch* ke-20 : 0,0010 Val_MSE *Epoch* ke-1 : 0,0331 ... *Epoch* ke-20 : 0,0017 0,05
	
Train_MSE *Epoch* ke-1 : 0,0143 ... *Epoch* ke-20 : 0,0011 Val_MSE *Epoch* ke-1 : 0,0538 ... *Epoch* ke-20 : 0,0014	Train_MSE *Epoch* ke-1 : 0,0274 ... *Epoch* ke-20 : 0,0011 Val_MSE *Epoch* ke-1 : 0,3854 ... *Epoch* ke-20 : 0,0020

Kurva *Loss* MSE Noise Salt	
0,075	0,1
 Train_MSE *Epoch* ke-1 : 0,0147 ... *Epoch* ke-20 : 0,0013 Val_MSE *Epoch* ke-1 : 0,0515 ... *Epoch* ke-20 : 0,0015	 Train_MSE *Epoch* ke-1 : 0,0236 ... *Epoch* ke-20 : 0,0012 Val_MSE *Epoch* ke-1 : 0,1422 ... *Epoch* ke-20 : 0,0016
0,125	0,15
 Train_MSE *Epoch* ke-1 : 0,0142 ... *Epoch* ke-20 : 0,0013 Val_MSE *Epoch* ke-1 : 0,2649 ... *Epoch* ke-20 : 0,0015	 Train_MSE *Epoch* ke-1 : 0,0143 ... *Epoch* ke-20 : 0,0015 Val_MSE *Epoch* ke-1 : 0,0941 ... *Epoch* ke-20 : 0,0020
(g) 0,175	0,2

Gambar 44. Kurva *Loss* MSE Noise Salt

Pada Gambar (e) 0,125 dan Gambar (f) 0,15, terdapat sedikit pelebaran jarak antara *Training Loss* dan *Validation Loss* pada awal *Epoch*, yang mengindikasikan berkurangnya kemampuan generalisasi model. Fenomena penurunan performa lebih jelas terlihat pada *noise* varian Gambar (g) 0,175 dan Gambar (h) 0,2. Untuk Gambar (g) 0,175, meskipun *Training Loss* turun dari 0,0142 ke 0,0013 *Validation Loss* tetap

fluktuatif dan tinggi, dari 0,2649 ke 0,0015. Pada Gambar (h) 0,2, *Training Loss* menurun dari 0,0143 ke 0,0015, tetapi *Validation Loss* berawal dari nilai tinggi 0,0941 dan meskipun berakhir di 0,0020, menunjukkan adanya ketidakstabilan selama pelatihan. Dapat disimpulkan bahwa semakin tinggi tingkat varian *Noise Salt* yang diberikan ke citra, maka semakin besar tantangan yang dihadapi model dalam melakukan *denoising*. Model bekerja optimal pada *noise* rendah namun efektivitasnya menurun seiring meningkatnya intensitas *noise*.

Kurva *Loss* MSE *Pepper*

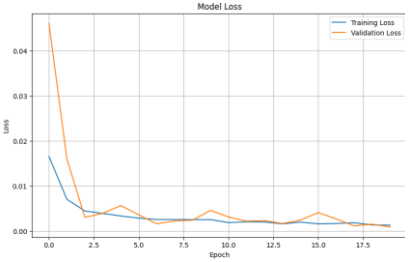
Kurva *Loss* MSE pada Gambar 45 menggambarkan performa model dalam melakukan *denoising* terhadap citra MRI yang terkontaminasi *Noise Pepper* dengan berbagai tingkat varian. Secara umum, kurva menunjukkan penurunan nilai *Training Loss* dan *Validation Loss* seiring berjalannya *Epoch*, namun peningkatan varian *noise* berdampak pada kestabilan dan efektivitas proses *denoising*. Pada Gambar (a) 0,025, *Training* MSE menurun dari 0,0299 pada *Epoch* ke-1 menjadi 0,0014 di *Epoch* ke-20. *Validation* MSE juga menurun dari 0,0460 menjadi 0,0008, menunjukkan kinerja yang sangat baik. Pada Gambar (b) 0,05 dengan *Training* MSE dari 0,0275 ke 0,0015 dan *Validation* MSE dari 0,0436 ke 0,0009, menandakan kemampuan *denoising* model yang masih kuat.

Untuk Gambar (c) 0,075, model menunjukkan penurunan *Training* MSE dari 0,0148 ke 0,0013 dan *Validation* MSE dari 0,0509 ke 0,0009, memperlihatkan performa yang cukup stabil. Namun, pada Gambar (d) 0,1 meskipun *Training* MSE turun dari 0,0241 ke 0,0014, *Validation* MSE yang awalnya 0,0463 hanya menurun ke 0,0012, menunjukkan model mulai mengalami kesulitan dalam *denoising noise*.

Ketika varian meningkat menjadi 0,125, *Training* MSE turun dari 0,0145 ke 0,0014, dan *Validation* MSE dari 0,0155 ke 0,0010. Meskipun hasil akhir tergolong baik, nilai *Validation* MSE awal yang tinggi menandakan bahwa *noise* mulai menghambat proses awal *denoising*. Pada Gambar (f) 0,15, *Training* MSE turun dari 0,0246 ke 0,0016 dan *Validation* MSE dari 0,0372 ke 0,0014, memperlihatkan penurunan efektivitas yang lebih signifikan. Varian yang lebih tinggi yaitu 0,175 dan 0,2 memberikan tantangan yang lebih besar.

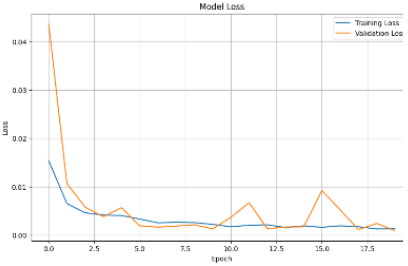
Pada Gambar (g) 0,175 *Training* MSE menurun dari 0,0142 ke 0,0019, tetapi *Validation* MSE dimulai dari nilai tinggi 0,0584 sebelum turun ke 0,0017. Sementara itu, pada Gambar (h) 0,2, *Training* MSE turun dari 0,0149 ke 0,0018 dan *Validation* MSE dari 0,0554 ke 0,0013, menunjukkan bahwa meskipun model mampu menurunkan *Loss*, *noise* yang tinggi tetap memicu fluktuasi besar di awal pelatihan. Secara keseluruhan, performa model dalam menangani *Noise Pepper* mengikuti pola yang mirip dengan *Noise Salt*, yakni performa sangat baik pada varian rendah hingga 0,075, namun mulai mengalami ketidakstabilan dan kesulitan generalisasi pada varian tinggi seperti pada varian 0,1 keatas. Terdapat beberapa indikasi model sedikit mengalami *overfitting* ditandai mulai meningkatnya kurva pada ecpoh terakhir di beberapa varian. Fenomena ini menunjukkan bahwa kompleksitas *noise* mempengaruhi kemampuan model dalam *denoising* citra.

Kurva *Loss* MSE Noise Pepper



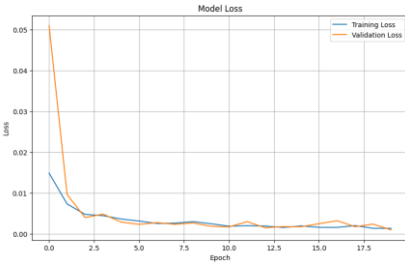
Train_MSE *Epoch* ke-1 : 0.0299 ... *Epoch* ke-20 : 0,0014
Val_MSE *Epoch* ke-1 : 0.0460 ... *Epoch* ke-20 : 0,0008

0,025



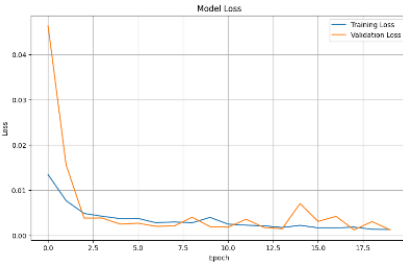
Train_MSE *Epoch* ke-1 : 0,0275 ... *Epoch* ke-20 : 0,001
Val_MSE *Epoch* ke-1 : 0,0436 ... *Epoch* ke-20 : 0,0009

0,05



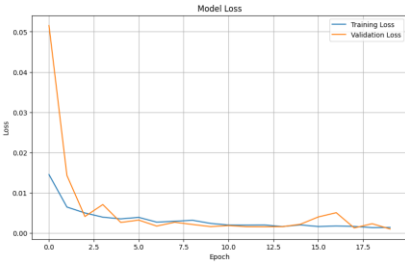
Train_MSE *Epoch* ke-1 : 0,0148 ... *Epoch* ke-20 : 0,0013
Val_MSE *Epoch* ke-1 : 0,0509 ... *Epoch* ke-20 : 0,0009

0,075

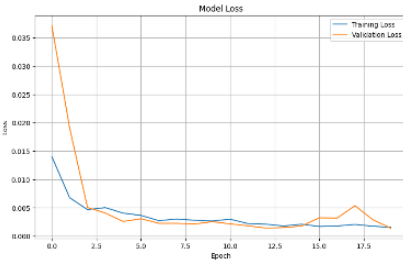


Train_MSE *Epoch* ke-1 : 0,0241 ... *Epoch* ke-20 : 0,0014
Val_MSE *Epoch* ke-1 : 0,0463 ... *Epoch* ke-20 : 0,0012

0,1



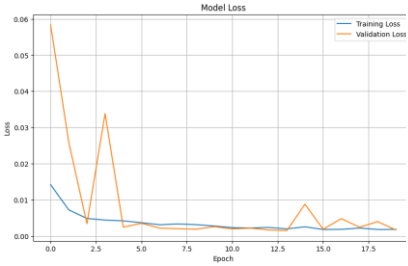
Train_MSE *Epoch* ke-1 : 0,0145 ... *Epoch* ke-20 : 0,0014
Val_MSE *Epoch* ke-1 : 0,0515 ... *Epoch* ke-20 : 0,0010



Train_MSE *Epoch* ke-1 : 0,0246 ... *Epoch* ke-20 : 0,0016
Val_MSE *Epoch* ke-1 : 0,0372 ... *Epoch* ke-20 : 0,0014

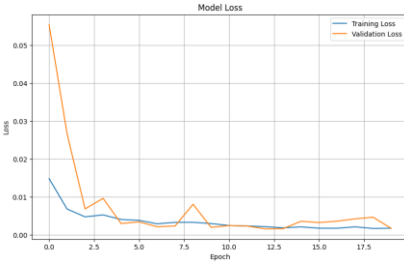
Kurva *Loss* MSE Noise Pepper

0,125



Train_MSE *Epoch* ke-1 : 0,0142 ... *Epoch* ke-20 : 0,0019
Val_MSE *Epoch* ke-1 : 0,0584 ... *Epoch* ke-20 : 0,0017

0,15



Train_MSE *Epoch* ke-1 : 0,0149 ... *Epoch* ke-20 : 0,0018
Val_MSE *Epoch* ke-1 : 0,0554 ... *Epoch* ke-20 : 0,0018

0,175

0,2

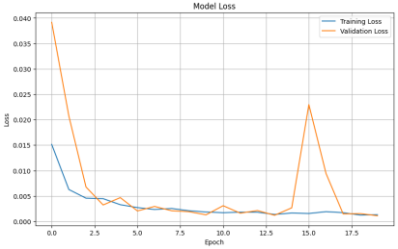
Gambar 45. Kurva *Loss* MSE Noise Pepper

Kurva *Loss* MSE Salt and Pepper

Kurva *Loss* MSE pada Gambar 46. menunjukkan performa model dalam menangani *noise* kombinasi *Salt and Pepper* pada berbagai tingkat varian. Secara umum, model mampu menurunkan nilai *Loss* yang konsisten pada data pelatihan maupun validasi di semua varian, meskipun kompleksitas *noise* gabungan ini membuat proses *denoising* menjadi lebih sulit dibandingkan dengan *noise* tunggal seperti *Salt* atau *Pepper* saja. Pada Gambar (a) 0,025, nilai *Training* MSE menurun dari 0,0151 di *Epoch* ke-1 menjadi 0,0013 di *Epoch* ke-20, dan *Validation* MSE dari 0,0392 menjadi 0,0011. Ada lonjakan besar di *validation Loss* sementara *training Loss* tetap rendah pada *Epoch* ke-15 mengindikasikan *overfitting* sesaat atau gangguan dalam data validasi saat pelatihan. Pada Gambar (b) 0,05 dengan *Training* MSE dari 0,0256 ke 0,0012 dan *Validation* MSE dari 0,0435 ke 0,0016, menunjukkan bahwa model masih mampu melakukan generalisasi dengan baik meski *noise* meningkat. Ketika varian

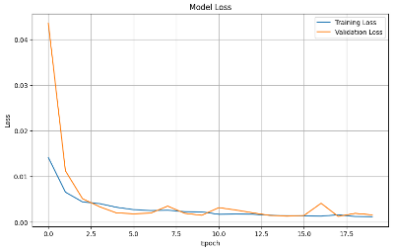
bertambah menjadi Gambar (c) 0,075 dan Gambar (d) 0,1, model tetap menunjukkan penurunan yang signifikan. Pada varian 0,075, *Training* MSE menurun dari 0,0132 ke 0,0012 dan *Validation* MSE dari 0,0436 ke 0,0013. Pada varian 0,1, *Training* MSE turun dari 0,0268 ke 0,0015 dan *Validation* MSE dari 0,0434 ke 0,0018. Meskipun penurunan *Loss* masih terjadi, variasi grafik mulai menunjukkan sedikit fluktuasi, terutama pada *Validation Loss*.

Kurva *Loss* MSE Noise Salt and Pepper



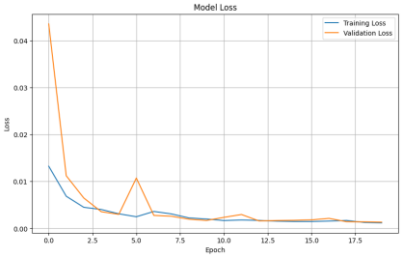
Train_MSE *Epoch* ke-1 : 0,0151 ... *Epoch* ke-20 : 0,0013
Val_MSE *Epoch* ke-1 : 0,0392 ... *Epoch* ke-20 : 0,0011

0,025



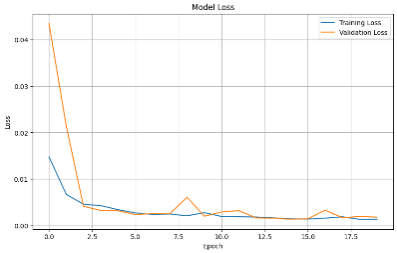
Train_MSE *Epoch* ke-1 : 0,0256 ... *Epoch* ke-20 : 0,0012
Val_MSE *Epoch* ke-1 : 0,0435 ... *Epoch* ke-20 : 0,0016

0,05



Train_MSE *Epoch* ke-1 : 0,0132 ... *Epoch* ke-20 : 0,0012
Val_MSE *Epoch* ke-1 : 0,0436 ... *Epoch* ke-20 : 0,0013

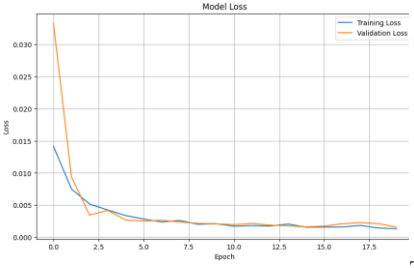
0,075



Train_MSE *Epoch* ke-1 : 0,0268 ... *Epoch* ke-20 : 0,0015
Val_MSE *Epoch* ke-1 : 0,0434 ... *Epoch* ke-20 : 0,0018

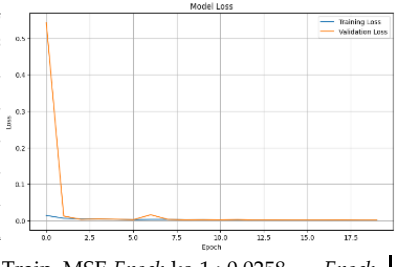
0,1

Kurva *Loss* MSE Noise Salt and Pepper



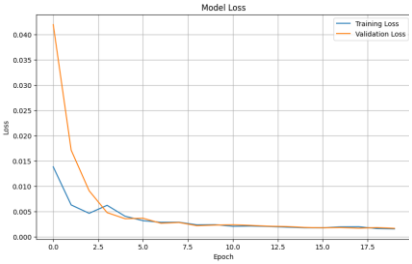
Train_MSE *Epoch* ke-1 : 0,0141 ... *Epoch* ke-20 : 0,0013
Val_MSE *Epoch* ke-1 : 0,0332 ... *Epoch* ke-20 : 0,0016

0,125



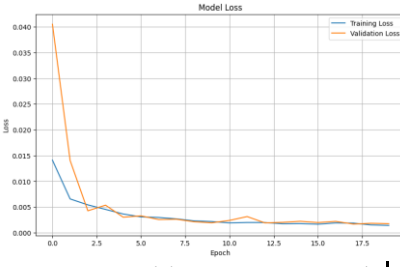
Train_MSE *Epoch* ke-1 : 0,0258 ... *Epoch* ke-20 : 0,0014
Val_MSE *Epoch* ke-1 : 0,5425 ... *Epoch* ke-20 : 0,0015

0,15



Train_MSE *Epoch* ke-1 : 0,0138 ... *Epoch* ke-20 : 0,0015
Val_MSE *Epoch* ke-1 : 0,0419 ... *Epoch* ke-20 : 0,0016

0,175



Train_MSE *Epoch* ke-1 : 0,0141 ... *Epoch* ke-20 : 0,0014
Val_MSE *Epoch* ke-1 : 0,0405 ... *Epoch* ke-20 : 0,0017

0,2

Gambar 46. Kurva *Loss* MSE Noise Salt and Pepper

Pada Gambar (e) 0,125, nilai *Training* MSE turun dari 0,0141 ke 0,0013 dan *Validation* MSE dari 0,0332 ke 0,0016, yang menunjukkan kestabilan model. Pada Gambar (f) 0,15, terlihat sedikit peningkatan *Validation* MSE awal dengan nilai 0,0455 meskipun akhirnya turun menjadi 0,0015 di *Epoch* ke-20. *Training* MSE turun dari 0,0258 ke 0,0014, mengindikasikan proses pelatihan tetap berjalan optimal. Untuk varian tinggi

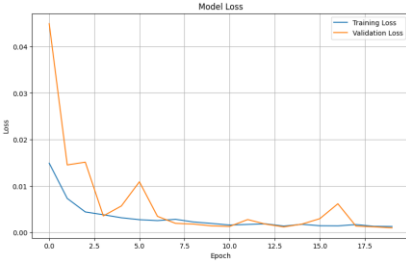
yaitu pada Gambar (g) 0,175 dan Gambar (h) 0,2, model tetap mampu menjaga *Loss* di kisaran rendah. Pada varian 0,175, *Training* MSE menurun dari 0,0138 ke 0,0015 dan *Validation* MSE dari 0,0419 ke 0,0016. Sementara pada varian 0,2, *Training* MSE turun dari 0,0141 ke 0,0014 dan *Validation* MSE dari 0,0405 ke 0,0017. Meskipun nilai awal *Validation* MSE cukup tinggi, penurunan hingga mendekati nilai *training* menunjukkan model tetap mampu beradaptasi.

Secara keseluruhan, model menunjukkan kemampuan yang baik dalam mengatasi *noise* gabungan *Salt and Pepper*, terutama hingga varian 0,1. Namun pada varian yang lebih tinggi, efek *noise* mulai tampak pada *Validation Loss* yang sedikit lebih tinggi dan kurang stabil di awal pelatihan. Fenomena ini mencerminkan tantangan yang lebih besar dalam *denoising* gambar dengan *noise* kompleks, namun arsitektur model mampu mempertahankan performa yang cukup konsisten hingga akhir *Epoch*.

Kurva *Loss* MSE *Speckle*

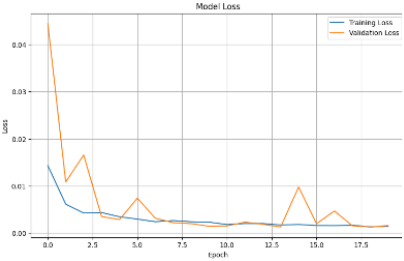
Kurva *Loss* MSE pada Gambar 47 menunjukkan performa model dalam menangani *noise Speckle* pada berbagai tingkat varian. Dari kurva-kurva tersebut, terlihat bahwa *Speckle Noise* merupakan jenis *noise* yang paling sulit untuk diatasi oleh model, dibandingkan dengan ketiga jenis *noise* lainnya (*Salt*, *Pepper*, dan *Salt* & *Pepper*). Hal ini tercermin dari tingginya nilai Validation MSE pada tahap awal pelatihan, yang mengindikasikan bahwa model masih kesulitan dalam melakukan generalisasi data secara optimal. Laju penurunan kurva yang relatif lambat, dikombinasikan dengan fluktuasi yang cukup signifikan sepanjang proses pelatihan, menunjukkan adanya ketidakstabilan optimasi yang perlu diperhatikan dalam evaluasi performa model.

Kurva *Loss* MSE Noise *Speckle*



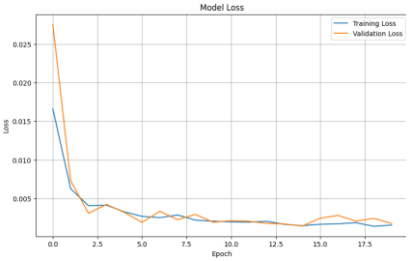
Train_MSE *Epoch* ke-1 : 0,0149 ... *Epoch* ke-20 : 0,0013
Val_MSE *Epoch* ke-1 : 0,0449 ... *Epoch* ke-20 : 0,0009

0,025



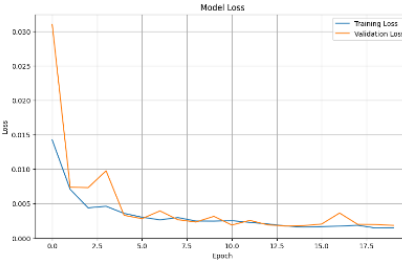
Train_MSE *Epoch* ke-1 : 0,0260 ... *Epoch* ke-20 : 0,0016
Val_MSE *Epoch* ke-1 : 0,0444 ... *Epoch* ke-20 : 0,0016

0,05



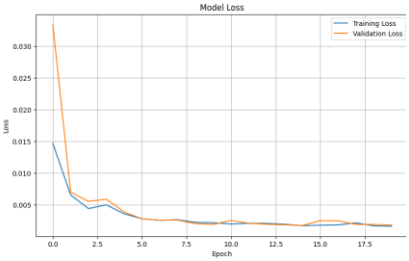
Train_MSE *Epoch* ke-1 : 0,0166 ... *Epoch* ke-20 : 0.0016
Val_MSE *Epoch* ke-1 : 0,0275 ... *Epoch* ke-20 : 0,0018

0,075

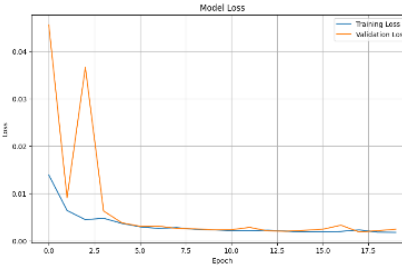


Train_MSE *Epoch* ke-1 : 0.0255 ... *Epoch* ke-20 : 0.0015
Val_MSE *Epoch* ke-1 : 0.0310 ... *Epoch* ke-20 : 0.0018

0,1



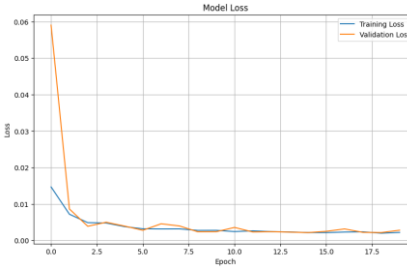
Train_MSE *Epoch* ke-1 : 0,0147 ... *Epoch* ke-20 : 0,0016
Val_MSE *Epoch* ke-1 : 0,0334 ... *Epoch* ke-20 : 0,0018



Train_MSE *Epoch* ke-1 : 0,0245 ... *Epoch* ke-20 : 0,0019
Val_MSE *Epoch* ke-1 : 0,0456 ... *Epoch* ke-20 : 0,0025

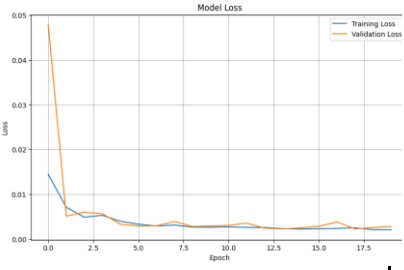
Kurva *Loss* MSE Noise *Speckle*

0,125



Train_MSE *Epoch* ke-1 : 0,0147 ... *Epoch* ke-20 : 0,0022
Val_MSE *Epoch* ke-1 : 0,0591 ... *Epoch* ke-20 : 0,0028

0,15



Train_MSE *Epoch* ke-1 : 0,0145 ... *Epoch* ke-20 : 0,0021
Val_MSE *Epoch* ke-1 : 0,0479 ... *Epoch* ke-20 : 0,0028

0,175

0,2

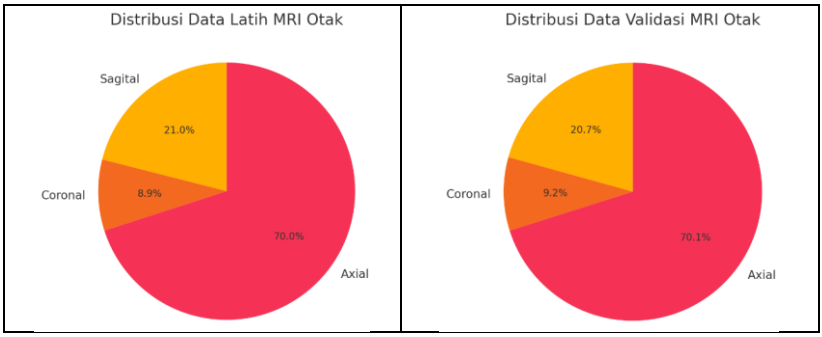
Gambar 47. Performa Model Menangani *Noise Speckle*.

Pada Gambar (a) 0,025, model memulai dengan Train MSE sebesar 0,0149 dan *Validation* MSE sebesar 0,0449, kemudian menurun masing-masing menjadi 0,0013 dan 0,0009 di *Epoch* ke-20. Meski terjadi penurunan, nilai awal yang tinggi menunjukkan bahwa *Speckle Noise* mengganggu struktur citra dengan cukup parah bahkan pada varian rendah. Pola ini juga terjadi pada Gambar (b) 0,05, dengan Train MSE turun dari 0,0260 ke 0,0016 dan *Validation* MSE dari 0,0444 ke 0,0016. Pada Gambar (e) 0,125, Train MSE turun dari 0,0147 ke 0,0016 dan *Validation* MSE dari 0,0334 ke 0,0018, sedangkan pada varian Gambar (f) 0,15, Train MSE dari 0,0245 ke 0,0019 dan *Validation* MSE dari 0,0420 ke 0,0025

➢ Analisis per Kelas Bidang MRI (Sagital, Koronal, Aksial)

Analisis keseimbangan data merupakan langkah yang penting dalam pelatihan model CAE untuk memastikan apakah model belajar secara adil, menyeluruh, dan akurat dari seluruh

representasi data citra MRI otak. Berdasarkan distribusi data yang ditunjukkan pada Gambar 48 terlihat bahwa persebaran data pada masing-masing bidang citra MRI otak, yaitu sagital, koronal, dan aksial , tidak seimbang. Data latih didominasi oleh citra aksial dengan jumlah 1.458, diikuti oleh sagital sebanyak 438, dan koronal sebanyak 186. Demikian pula pada data validasi, citra aksial berjumlah 366, sementara sagital dan koronal masing-masing hanya 108 dan 48 citra.



Gambar 48. Keseimbangan Data

Ketimpangan ini mengindikasikan bahwa model CAE akan lebih banyak belajar dari citra aksial dibandingkan dua bidang lainnya. Oleh karena itu, dilakukan analisis lebih lanjut terhadap hasil *denoising* yang dihasilkan oleh model untuk setiap bidang secara terpisah. Tujuan dari analisis ini adalah untuk mengevaluasi sejauh mana model dapat meng-generalisasi proses penghilangan *noise* pada masing-masing bidang, serta untuk mengetahui apakah ketidakseimbangan data tersebut berdampak pada kualitas hasil rekonstruksi citra. Analisis keseimbangan data digambarkan dengan *bar chart* berdasarkan nilai MSE dan PSNR tiap kelas bidang MRI otak. Jenis dan varian *noise* dipilih berdasarkan hasil terbaik di setiap jenis dan Metrik evaluasi diambil melalui rata-rata MSE dan

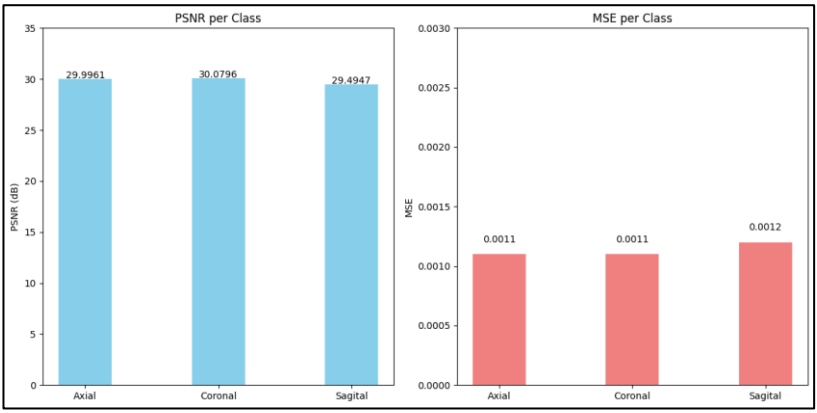
PSRN setiap kelas untuk membentuk analisis yang lebih representatif dan objektif.

Kode pada merupakan implementasi evaluasi performa model *Autoencoder* terhadap citra medis yang telah terkontaminasi *noise*. Proses dimulai dengan mendefinisikan fungsi *load_and_prepare_image*, yang bertugas untuk memuat citra dalam skala abu-abu (*grayscale*), mengubah ukurannya menjadi 256x256 piksel, melakukan normalisasi dengan skala 0 hingga 1, serta mengembalikan *array* citra dalam bentuk tiga dimensi (tinggi, lebar, dan saluran warna). Selanjutnya, fungsi utama *evaluate_model_by_class* digunakan untuk mengevaluasi performa model pada masing-masing kelas citra, yaitu Aksial, Koronal, dan Sagital. Fungsi ini membaca seluruh citra dalam folder yang telah ditentukan untuk setiap kelas, lalu menambahkan *noise* buatan ke setiap citra menggunakan fungsi *add_Noise*. Model *Autoencoder* yang telah dimuat sebelumnya kemudian digunakan untuk merekonstruksi citra yang telah diberi *noise*. Untuk mengukur kualitas hasil rekonstruksi, dua metrik digunakan yaitu MSE dan PSNR. Kedua metrik ini dihitung antara citra asli dan citra hasil rekonstruksi. Nilai MSE mengindikasikan besarnya kesalahan rekonstruksi, sedangkan PSNR mengukur kualitas visual rekonstruksi dalam satuan desibel (dB). Hasil evaluasi berupa rata-rata MSE dan PSNR untuk masing-masing kelas kemudian disimpan dalam *dictionary class_metrics*. Model yang digunakan diload dari file eksternal bertipe "*.h5*" dengan opsi *compile=False* untuk memuat model tanpa melakukan kompilasi ulang. Direktori yang berisi citra uji telah ditentukan untuk masing-masing kelas. Setelah proses evaluasi selesai, hasil evaluasi ditampilkan untuk setiap kelas, yang mencakup nilai rata-rata MSE dan PSNR. Dengan pendekatan ini, peneliti dapat mengevaluasi kemampuan model dalam merekonstruksi citra yang telah

terdistorsi oleh *noise* serta mengamati performanya berdasarkan sudut pandang citra medis yang berbeda.

Hasil *Denoising* Setiap Kelas pada *Noise Salt* Varian 0,025

Terlihat pada Gambar 49 bidang aksial , bidang yang memiliki jumlah data paling banyak dalam set pelatihan dan validasi, menghasilkan nilai PSNR tertinggi dan MSE yang relatif rendah. Hal ini menunjukkan bahwa model CAE mampu melakukan proses *denoising* dengan lebih baik pada citra aksial, karena model lebih sering belajar terhadap variasi data dari bidang ini selama pelatihan. Sebaliknya, pada bidang sagital yang jumlah datanya jauh lebih sedikit dibandingkan aksial, performa model sedikit menurun dengan MSE yang lebih tinggi dan PSNR yang lebih rendah. Meskipun koronal merupakan bidang dengan jumlah data paling sedikit dalam dataset 186 untuk data latih dan 48 untuk data validasi, performa model pada bidang ini justru lebih baik dibandingkan bidang sagital, yang memiliki jumlah data lebih banyak. Hal ini ditunjukkan oleh nilai rata-rata MSE yang sama dengan aksial 0,0011 dan PSNR yang sedikit lebih tinggi dari aksial, yaitu 30,0796 dB, dan lebih tinggi daripada sagital dengan PSNR sebesar 29,4947 dB.

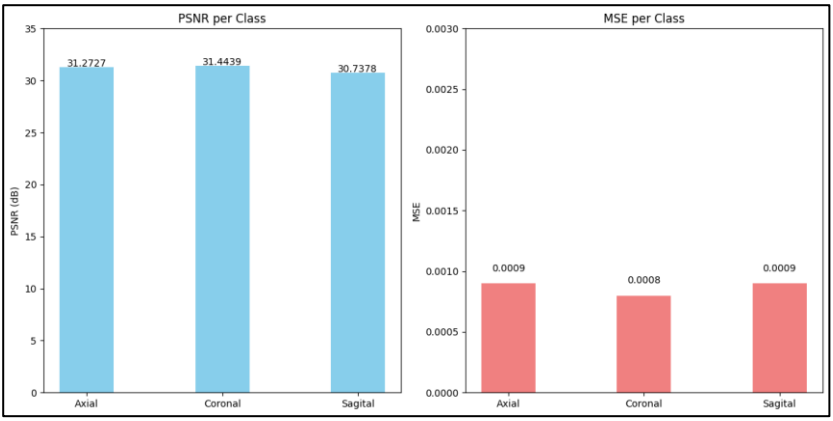


Gambar 49. Diagram Bar setiap kelas hasil *Salt*

Hasil *Denoising* Setiap Kelas pada *Noise Pepper* Varian 0,025

Berdasarkan Gambar 50 bidang Sagital memiliki performa terendah dengan MSE tertinggi 0,0009 dan PSNR terendah 30,7378 dB. Meskipun bidang Aksial jumlah datanya paling banyak, nilai PSNR-nya bukan yang tertinggi, yaitu hanya bernilai 31,2727 dB. Ini menunjukkan bahwa meskipun memiliki data lebih banyak memang dapat membantu performa model, kualitas citra aksial atau kompleksitas struktur pada bidang ini mungkin sedikit lebih tinggi, sehingga hasil *denoising* tidak mencapai puncak performa.

Walaupum merupakan bidang dengan data yang paling sedikit bidang Koronal justru menunjukkan performa *denoising* terbaik diantara ketiga bidang dengan PSNR tertinggi 31,4439 dB. Fenomena ini dapat disebabkan oleh karakteristik citra koronal yang memiliki struktur yang lebih konsisten atau struktur citra yang lebih sederhana daripada bidang sagital, sehingga model mampu melakukan *denoising* dengan lebih efektif meskipun data terbatas. Struktur yang lebih sederhana ini memungkinkan model untuk lebih mudah belajar menghilangkan *noise* dan mempertahankan detail pada citra.



Gambar 50. Diagram Bar setiap kelas hasil *Pepper*

Hasil *Denoising* Setiap Kelas pada *Noise Salt and Pepper* Varian 0,025

Berdasarkan Gambar 51, bidang aksial menunjukkan nilai MSE sebesar 0,0011 dan PSNR sebesar 30,0501 dB yang mencerminkan performa *denoising* baik pada *noise* ini. Hal ini selaras dengan jumlah data aksial yang paling banyak digunakan dalam pelatihan, sehingga model memperoleh lebih banyak variasi fitur dan mampu melakukan generalisasi yang lebih baik terhadap *noise*. Meskipun jumlah data koronal merupakaan data dengan jumlah paling sedikit hasil *denoising* pada bidang ini justru menjadi yang terbaik, dengan PSNR 30,1113 dB dan MSE 0,0011 dibandingkan bidang sagital yang menunjukkan MSE tertinggi 0,0012 dan PSNR terendah 29,5010 dB dibandingkan kedua bidang lainnya. Pada citra sagital, terlihat model mengalami kesulitan dalam *denoising*. Citra sagital, meskipun memiliki lebih banyak data daripada koronal, mengalami kesulitan dalam *denoising*. Citra sagital memiliki pola yang lebih sulit dikenali dibandingkan dengan aksial dan koronal.

Gambar 51. Diagram Bar setiap kelas hasil *Salt and Pepper*

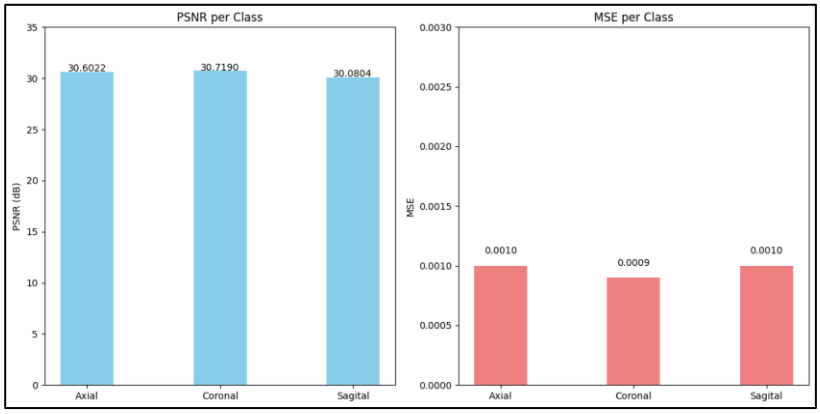
Hasil *Denoising* Setiap Kelas pada *Noise Speckle* Varian 0,05

Terlihat pada Gambar 52 model menghasilkan MSE sebesar 0,0011 dan PSNR sebesar 30,6022 dB, yang menunjukkan kinerja yang cukup baik dalam mengurangi *Noise Speckle* pada citra aksial. Pada Koronal hasil *denoising* sedikit lebih tinggi dibandingkan dengan citra aksial , dengan MSE sebesar 0,0011 dan PSNR sebesar 30,7190 dB. Sementara itu, pada Sagital, model menunjukkan MSE tertinggi sebesar 0,0012 dan PSNR terendah sebesar 30,0804 dB di antara ketiga kelas. Sama seperti di ketiga *noise* lainya.

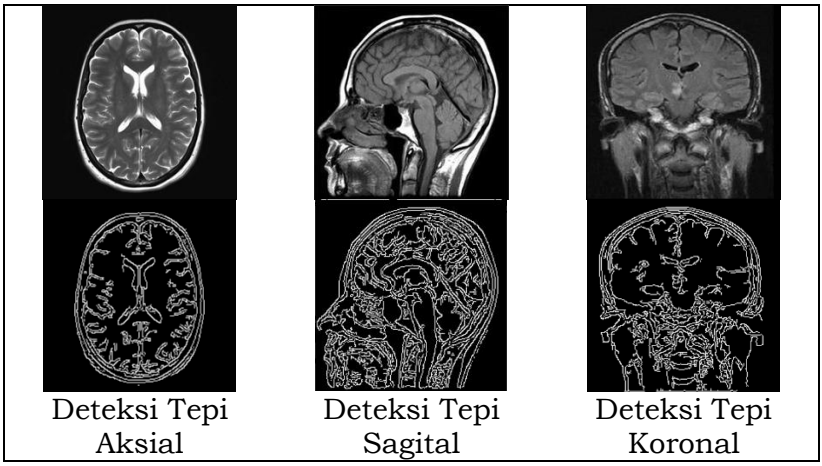
Dalam konteks *denoising* untuk empat jenis *Noise Salt*, *Pepper*, *Salt and Pepper*, dan *Speckle* pada varian 0,025, hasil yang diperoleh menunjukkan adanya perbedaan performa model pada setiap bidang citra MRI otak, meskipun distribusi data tidak seimbang antar bidang. Penurunan kinerja pada bidang Sagital yang memiliki lebih banyak data dibandingkan dengan Koronal yang memiliki lebih sedikit data namun menghasilkan hasil *denoising* yang lebih baik, bahkan menjadi kelas dengan PSNR dan MSE tertinggi pada keempat jenis *noise*. Hasil ini

menunjukan pentingnya struktur citra dan karakteristik *noise* dalam mempengaruhi kinerja model bukan hanya jumlah data.

Hal ini menunjukan bahwa struktur citra pada bidang koronal dan aksial, yang memiliki lebih sedikit detail, lebih mudah untuk dipertahankan, hal ini terlihat dari deteksi tepi yang lebih sederhana pada kedua kelas citra tersebut seperti yang dapat dilihat pada Gambar 53. Sebaliknya, pada bidang sagital yang memiliki jumlah dataset lebih banyak dibandingkan dengan bidang koronal, model CAE mengalami kesulitan dalam mempertahankan detail citra setelah diinduksi dengan empat jenis *noise* yang berbeda. Kesulitan ini terjadi karena citra pada bidang sagital mengandung lebih banyak informasi yang harus dipertahankan.



Gambar 52. Diagram Bar setiap kelas hasil *Speckle*



Gambar 53. Garis Tepi MRI Otak

Dapat disimpulkan hasil *denoising* tidak hanya dipengaruhi oleh banyaknya jumlah dataset pada setiap kelas, melainkan struktur citra MRI juga memiliki peran yang sangat penting dalam mempengaruhi kinerja model CAE untuk mengurangi *noise* sambil mempertahankan detail citra. Meskipun bidang koronal memiliki jumlah dataset yang paling sedikit, kelas koronal tidak menunjukkan hasil terendah dalam proses *denoising*.

➢ Perbandingan CAE vs *U-Net*

Secara keseluruhan, hasil evaluasi yang terlihat pada Tabel 11 dan Tabel 12 menunjukkan bahwa arsitektur *U-Net* unggul dibandingkan dengan CAE dalam hal kemampuan untuk mengurangi *noise* dan menghasilkan gambar yang lebih berkualitas. Keunggulan *U-Net* dalam berdasarkan metrik PSNR dan MSE ini menjadikannya pilihan yang lebih baik untuk aplikasi yang memerlukan pemulihan gambar dengan kualitas tinggi. Kelebihan ini dikarenakan struktur arsitektur

U-Net yang dirancang dengan struktur *skip-connection*. Hal ini membuatnya sedikit kurang efektif dalam mempertahankan detail-detail penting pada gambar yang terkontaminasi *noise*, yang tercermin dari nilai PSNR yang lebih rendah dan MSE yang lebih tinggi dibandingkan dengan *U-Net*. Meskipun CAE juga merupakan model yang kuat dalam *denoising*, arsitekturnya tidak memiliki *skip-connection*. membuatnya sedikit kurang efektif dalam mempertahankan detail-detail penting pada gambar.

Tabel 11. Perbandingan PSNR model *U-Net* dan CAE

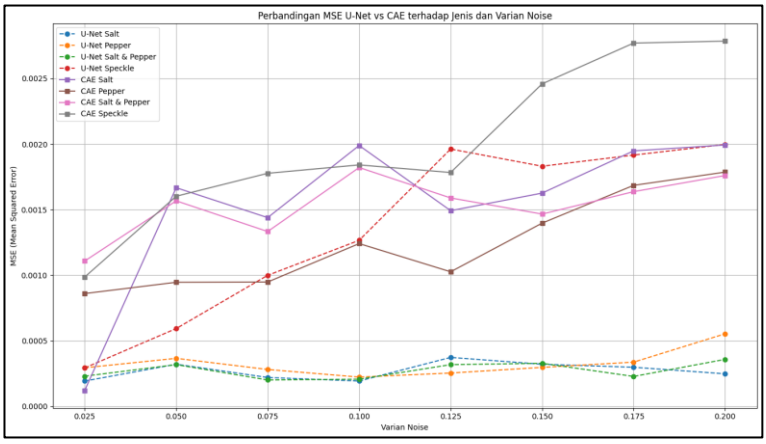
Metrik Evaluasi	Model	Varian *Noise*	Jenis *Noise*			
			Salt	*Pepper*	*Salt & Pepper*	*Speckle*
PSNR (dB)	*U-Net*	0,025	37,4460	35,3204	36.6556	35,4622
		0,05	35,4287	34,6990	35,2931	32,4261
		0,075	36,7699	35,8220	37,3201	30,1526
		0,1	37,4250	36,7617	37,2047	29,1379
		0,125	35,1250	36,4131	35,2936	27,2971
		0,15	35,4003	35,5391	35,2127	27,5417
		0,175	35,5539	35,2404	36,8610	27,3604
		0,2	36,3261	33,2813	34,9644	27,2220
	CAE	0,025	29,9262	31,1651	29,9518	30,5091
		0,05	28,2765	30,7346	28,8030	28,4495
		0,075	28,8862	30,7521	29,1296	27,7341
		0,1	27,5998	29,4030	27,7834	27,6893
		0,125	28,6794	30,4348	28,3477	27,7458
		0,15	28,1191	28,9051	28,7272	26,4614
		0,175	27,5775	28,1476	28,2832	25,8317
		0,2	27,3503	27,9315	27,8364	25,8930

Tabel 12. Perbandingan MSE model *U-Net* dan CAE

Metrik Evaluasi	Model	Varian *Noise*	Jenis *Noise*			
			Salt	*Pepper*	*Salt & Pepper*	*Speckle*
MSE	*U-Net*	0,025	0,000193	0,000294	0,000230	0,000293
		0,05	0,000321	0,000366	0,000318	0,000593
		0,075	0,000220	0,000281	0,000201	0,000999
		0,1	0,000193	0,000223	0,000207	0,001266
		0,125	0,000373	0,000254	0,000318	0,001962
		0,15	0,000321	0,000297	0,000328	0,001831
		0,175	0,000298	0,000337	0,000228	0,001916
		0,2	0,000248	0,000553	0,000358	0,001997
	CAE	0,025	0,001121	0,000860	0,001108	0,000986
		0,05	0,001667	0,000946	0,001567	0,001602
		0,075	0,001439	0,000948	0,001333	0,001777
		0,1	0,001989	0,001241	0,001821	0,001841
		0,125	0,001492	0,001026	0,001589	0,001783
		0,15	0,001626	0,001397	0,001466	0,002459
		0,175	0,001948	0,001685	0,001637	0,002770
		0,2	0,001993	0,001786	0,001760	0,002785

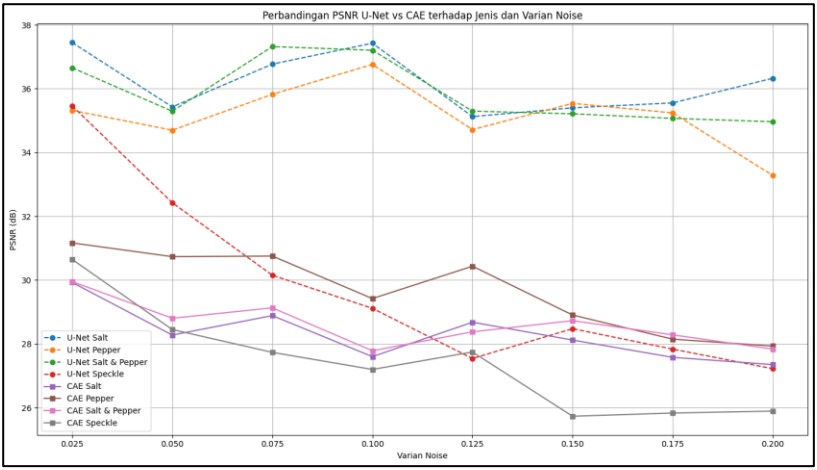
Secara garis besar terlihat pada Tabel 11 dan Tabel 12, performa model *U-Net* dan CAE dalam melakukan *denoising* terhadap berbagai jenis *noise* menunjukkan perbedaan signifikan yang tergambar melalui metrik evaluasi PSNR dan MSE. Model *U-Net* secara konsisten unggul dibandingkan CAE pada seluruh kombinasi jenis dan varian *noise*. Hasil terbaik *U-Net* terlihat pada *Noise Salt* varian 0,025 dan 0,1 dengan PSNR tertinggi sebesar 37,4460 dB dan MSE terendah sebesar 0,000193, mencerminkan kemampuan rekonstruksi yang sangat baik dan akurat mendekati citra asli. Performa terburuk *U-Net* tercatat pada *Noise Speckle* varian 0,2, dengan PSNR

sebesar 27,2220 dB dan MSE sebesar 0,001997, menandakan *U-Net* juga mengalami kesulitan dalam menanggulangi *Noise Speckle*. Sementara itu, model CAE menunjukkan performa terbaik pada *Noise Pepper* varian 0,025, dengan PSNR sebesar 31,1651 dB dan MSE sebesar 0,000860. Namun secara umum, nilai PSNR yang dicapai CAE lebih rendah dibandingkan *U-Net*, menunjukkan hasil rekonstruksi yang kurang detail. Terdapat satu hasil CAE yang berhasil mengalahkan *U-Net* di intensitas dan jenis *noise* yang sama yaitu terjadi pada *Speckle* varian 0,125. CAE menghasilkan nilai PSNR sebesar 27,7458 dB dan MSE sebesar 0,001783, sementara *U-Net* hanya menghasilkan nilai PSNR sebesar 27,2971 dB dan MSE 0,001962 sebesar menegaskan bahwa walaupun *Noise Speckle* menjadi tantangan utama bagi kedua model, terdapat satu skenario dimana CAE yang lebih sederhana menghasilkan hasil lebih baik dibandingkan *U-Net* terlihat pada *cell* bewarna kuning.



Gambar 54. Perbandingan MSE model *U-Net* dan CAE

Berdasarkan Gambar 54 secara umum, model *U-Net* menunjukkan performa yang lebih baik (nilai MSE lebih rendah) dibandingkan dengan CAE untuk semua jenis *noise* di hampir seluruh rentang varian. Hal ini mencerminkan kemampuan *U-Net* dalam merekonstruksi citra yang lebih mendekati kondisi aslinya, terutama dalam menangani *noise* jenis salt dan pepper, di mana kurva MSE *U-Net* lebih stabil dan rendah. Sebaliknya, model CAE menunjukkan nilai MSE yang cenderung lebih tinggi, khususnya pada *noise* jenis salt & pepper dan *Speckle*, yang mengindikasikan bahwa model ini kurang efektif dalam menangani *noise* dengan karakteristik kompleks dan acak.



Gambar 55. Perbandingan PSNR model *U-Net* dan CAE

Berdasarkan Gambar 55 secara keseluruhan, model *U-Net* menunjukkan performa *denoising* yang lebih unggul dibandingkan CAE, terlihat dari nilai PSNR yang secara konsisten lebih tinggi di semua jenis *noise* dan pada hampir seluruh tingkat varian. Pada *noise* jenis salt dan salt & pepper, model *U-Net* mampu mempertahankan nilai PSNR di atas 35

dB ketika varian *noise* meningkat, sedangkan CAE menghasilkan nilai PSNR yang relatif rendah, terutama pada *noise* jenis *Speckle*. Penurunan nilai PSNR yang cukup tajam juga terlihat pada model CAE ketika varian *noise* bertambah, khususnya untuk *Speckle noise*, yang mengindikasikan keterbatasan model CAE dalam menangani *noise* berstruktur acak dan kompleks.

Dari analisis ini, dapat disimpulkan bahwa *U-Net* lebih kuat terhadap variasi *noise*, terutama pada kondisi *noise* ringan hingga sedang. Sementara CAE, meskipun masih mampu memberikan hasil rekonstruksi yang cukup baik pada beberapa varian *noise* rendah, cenderung menunjukkan degradasi performa yang lebih cepat seiring meningkatnya intensitas *noise*, khususnya untuk *Speckle Noise*. Hal ini disebabkan kelebihan *U-Net* yang memanfaatkan *Skip-connection* dimana informasi yang telah diekstraksi pada lapisan awal (*low-level features*) dapat dilewatkan atau "di-*skip*" langsung ke bagian *decoder* untuk mempertahankan detail penting dalam gambar meskipun terpapar *noise*. Sebaliknya, meskipun CAE juga merupakan model yang kuat dalam pemulihan gambar, arsitekturnya tidak dirancang dengan pendekatan *skip-connection*. Hal ini membuatnya sedikit kurang efektif dalam mempertahankan detail-detail penting pada gambar yang terkontaminasi *noise*, yang tercermin dari nilai PSNR yang lebih rendah dan MSE yang lebih tinggi dibandingkan dengan *U-Net*.

BAB VIII
IMPLIKASI KLINIS DAN REKOMENDASI PRAKTIS

➢ Potensi Penggunaan CAE dalam Sistem Pendukung Diagnosis

Citra MRI yang jernih dan minim *noise* merupakan prasyarat fundamental bagi ketepatan interpretasi diagnostik. Dalam kerangka inilah CAE menemukan posisinya yang paling strategis, yaitu sebagai modul prapemrosesan dalam sebuah sistem pendukung keputusan klinis. CAE mampu memulihkan kualitas citra dengan nilai *Peak Signal-to-Noise Ratio* (PSNR) tertinggi mencapai 31,17 dB pada citra yang terkorupsi *Noise* Pepper sebesar 2,5 persen. Capaian ini melampaui ambang 30 dB yang, sebagaimana dirumuskan pada Bab 2, dikategorikan sebagai kualitas yang sangat baik dan sesuai untuk keperluan diagnosis medis. Dengan demikian, secara kuantitatif CAE telah memenuhi standar mutu yang dibutuhkan untuk mendukung pembacaan citra pada beberapa kondisi gangguan.

Posisi CAE dalam alur diagnosis dapat dipahami sebagai tahap antara, yaitu citra hasil akuisisi yang masih terdegradasi terlebih dahulu dilewatkan melalui jaringan untuk dipulihkan menjadi citra yang lebih bersih, sebelum kemudian dianalisis lebih lanjut, baik oleh tenaga medis secara langsung maupun oleh modul analisis otomatis pada tahap berikutnya. Sebagaimana telah diuraikan pada Bab 2, penggabungan teknik *denoising* berbasis *Autoencoder* dengan model klasifikasi seperti *Convolutional Neural Network* (CNN) memungkinkan kualitas

masukan yang dianalisis menjadi jauh lebih baik, sehingga hasil diagnosis pun berpotensi lebih akurat. CAE dengan demikian tidak berdiri sebagai sistem diagnosis yang otonom, melainkan sebagai komponen penguat yang meningkatkan keandalan keseluruhan rangkaian sistem.

Keunggulan lain yang menjadikan CAE relevan untuk integrasi sistem terletak pada sifat pelatihannya yang tidak memerlukan pelabelan eksplisit dalam konteks reduksi *noise*. Karakter ini memberikan keluwesan dan skalabilitas, karena model dapat dilatih memanfaatkan ketersediaan citra dalam jumlah besar tanpa beban anotasi yang mahal. Setelah dilatih, proses *denoising* dapat berlangsung secara otomatis, konsisten, dan relatif cepat, sehingga membuka peluang penerapan pada alur kerja klinis yang menuntut keluaran yang seragam dan dapat diandalkan dari waktu ke waktu.

➢ Manfaat untuk Radiolog dan Praktisi Medis

Manfaat paling langsung dari penerapan CAE dirasakan pada peningkatan kejelasan visual citra. Reduksi *noise* yang efektif memungkinkan struktur halus jaringan otak serta batas antara jaringan normal dan patologis tampak lebih tegas. Pembahasan ini menunjukkan bahwa model mampu memulihkan detail citra MRI otak tanpa menghasilkan artefak yang signifikan pada sebagian besar kondisi gangguan yang diuji. Bagi radiolog, kondisi ini secara langsung mengurangi ketidakpastian interpretasi yang kerap timbul akibat gangguan acak, terutama pada diagnosis penyakit neurologis yang menuntut visualisasi rinci seperti tumor otak, stroke, *multiple sclerosis*, maupun penyakit *Alzheimer* sebagaimana disinggung pada Bab 1.

Pada tataran alur kerja, kemampuan memulihkan citra yang terdegradasi membuka kemungkinan efisiensi yang berarti. Apabila *noise* yang muncul selama akuisisi dapat direduksi secara komputasional pada tahap pascapemindaian, maka kebutuhan untuk mengulang pemindaian akibat kualitas citra yang kurang memadai dapat ditekan. Hal ini berimplikasi pada penghematan waktu, peningkatan kenyamanan pasien, serta berpotensi memperpendek durasi pemindaian. Selain itu, penerapan model yang konsisten dapat berkontribusi pada standardisasi kualitas citra, sehingga variasi interpretasi yang bersumber dari perbedaan kualitas masukan antarpemeriksaan dapat diperkecil.

Walaupun demikian, manfaat tersebut perlu di-tempatkan dalam kerangka yang tepat. CAE berperan sebagai alat bantu yang memperkuat kapasitas tenaga medis, bukan sebagai pengganti penilaian klinis. Keputusan diagnostik akhir tetap berada pada radiolog dan praktisi medis, yang memadukan informasi citra dengan konteks klinis pasien secara menyeluruh. Dengan demikian, kontribusi terbesar CAE bukanlah menggantikan keahlian manusia, melainkan menyediakan citra dengan kualitas yang lebih baik sebagai landasan bagi pengambilan keputusan yang lebih percaya diri dan terinformasi.

➢ Batasan Implementasi dan Akurasi

Berdasarkan pembahasan yang telah dilakukan mengenai penggunaan *Convolutional Autoencoder* (CAE) untuk mereduksi *Noise* pada citra MRI otak, dapat disimpulkan bahwa penggunaan CAE sebagai metode reduksi *noise* terbukti efektif. Model CAE berhasil mengurangi berbagai jenis *Noise* yang ada pada citra MRI, seperti *Salt*, *Pepper*, dan *Salt & Pepper*, dengan metrik evaluasi MSE dan PSNR yang menunjukkan

peningkatan kualitas citra setelah dilakukan *denoising*. Model CAE berhasil meningkatkan kualitas citra, dengan nilai PSNR tertinggi mencapai 31,17 dB dan MSE terendah 0,00086 pada citra MRI yang terkorupsi *Noise* Pepper sebesar 2,5 persen, serta tetap mampu memulihkan detail citra tanpa menghasilkan artefak yang signifikan. Meskipun demikian, sejumlah batasan perlu disadari secara jujur agar penerapan model di lingkungan klinis tidak melampaui jangkauan validitas yang telah dibuktikan.

Batasan pertama berkaitan dengan performa relatif model. Hasil perbandingan memperlihatkan bahwa CAE secara konsisten unggul lebih rendah dibandingkan arsitektur *U-Net* pada hampir seluruh kombinasi jenis dan varian *noise*. Ketiadaan jalur *skip-connection* pada CAE menyebabkan sebagian informasi spasial beresolusi tinggi hilang selama proses encoding, sehingga model relatif kurang mampu mempertahankan detail halus dibandingkan *U-Net*. Implikasinya, untuk aplikasi yang menuntut presisi rekonstruksi tertinggi, CAE konvensional belum tentu menjadi pilihan paling optimal.

Batasan kedua menyangkut keterbatasan terhadap jenis *noise* tertentu. Model menunjukkan degradasi performa yang paling tajam ketika berhadapan dengan *Speckle Noise*, dengan nilai PSNR menurun hingga sekitar 25,83 dB pada varian tinggi. Sifat *Speckle Noise* yang multiplikatif dan menyatu dengan informasi asli citra terbukti jauh lebih sulit dieliminasi dibandingkan *noise* impulsif. Hal ini menegaskan bahwa efektivitas model tidak seragam untuk semua karakteristik gangguan, dan kewaspadaan diperlukan terutama pada citra yang terpapar *noise* berstruktur acak dan kompleks.

Batasan ketiga bersumber dari karakteristik data dan kondisi pengujian. Kualitas rekonstruksi terbukti dipengaruhi oleh ketidakseimbangan distribusi data antarbidang irisan,

sehingga konsistensi performa antarbidang tidak sepenuhnya terjamin. Lebih jauh, model dilatih dan diuji menggunakan *noise* sintetis yang diterapkan secara seragam, sementara *noise* pada citra klinis nyata bersifat lebih heterogen. Dataset yang digunakan juga hanya memuat citra otak normal tanpa kasus patologis, sehingga kemampuan model dalam mempertahankan fitur abnormal yang bernilai diagnostik tinggi, seperti tumor atau lesi, belum tervalidasi. Keterbatasan ini merupakan persoalan krusial, sebab justru fitur-fitur tersebut yang menjadi sasaran utama pemeriksaan.

Batasan keempat bersifat metodologis dan etis. Metrik MSE dan PSNR yang digunakan sebagai tolok ukur belum sepenuhnya mencerminkan kualitas perseptual citra sebagaimana dipersepsi oleh mata manusia, sehingga nilai numerik yang baik tidak selalu setara dengan kebermanfaatan diagnostik yang setara. Selain itu, model pembelajaran mendalam kerap bersifat sukar ditafsirkan atau bersifat *black box*, padahal dalam praktik medis keputusan harus dapat dipertanggungjawabkan. Pertanyaan mengenai akuntabilitas apabila terjadi kesalahan, perlindungan dan keamanan data pasien, serta keterjelasan cara kerja model, sebagaimana telah disinggung dalam pembahasan *Explainable* AI pada bab terdahulu, merupakan isu yang harus dijawab sebelum adopsi berskala luas. Atas dasar seluruh pertimbangan tersebut, direkomendasikan agar penerapan CAE di lingkungan klinis didahului dengan validasi klinis yang ketat, diposisikan sebagai alat bantu yang berada di bawah pengawasan tenaga medis, dan terus disempurnakan untuk menutup celah-celah keterbatasan yang telah diuraikan.

BAB IX
TANTANGAN DAN ARAH PEMBAHASAN MASA DEPAN

➢ Penanganan *Noise* Kompleks

Salah satu batasan mendasar dari pembahasan ini terletak pada karakter *noise* yang digunakan dalam proses pelatihan dan pengujian. Empat jenis gangguan yang diuji, yaitu *Salt*, *Pepper*, *Salt* & *Pepper*, serta *Speckle*, diinjeksikan secara sintetis dan terdistribusi merata ke seluruh permukaan citra dengan tingkat varian yang terkontrol. Pendekatan ini memang memberikan keunggulan metodologis berupa kemampuan untuk mengisolasi variabel dan mengukur performa model secara objektif terhadap intensitas gangguan tertentu. Akan tetapi, kondisi tersebut belum sepenuhnya merepresentasikan realitas *noise* yang muncul pada citra MRI klinis, yang umumnya bersifat heterogen, tidak seragam secara spasial, dan kerap merupakan hasil superposisi dari beberapa sumber gangguan sekaligus.

Arah pengembangan yang pertama adalah pemodelan *noise* dengan distribusi spasial yang lebih bervariasi. Sebagai contoh, *noise* dapat diterapkan hanya pada sebagian wilayah citra, misalnya pada setengah bidang gambar atau pada kuadran tertentu seperti bagian kiri, kanan, atas, maupun bawah citra. Implementasi semacam ini dapat dilakukan melalui pembagian citra ke dalam struktur *grid*, misalnya partisi 2x2 atau 3x3, lalu menerapkan gangguan secara acak hanya pada sebagian sel *grid* yang terpilih. Skema ini memaksa model untuk membedakan wilayah yang terdegradasi dari

wilayah yang masih bersih, sehingga model dituntut mempelajari ekstraksi fitur dari konteks yang tidak terganggu untuk merekonstruksi bagian yang rusak. Strategi ini berpotensi meningkatkan efisiensi *denoising* sekaligus menyiapkan model agar lebih adaptif terhadap pola degradasi yang terlokalisasi, yang lebih sering dijumpai pada praktik pencitraan nyata.

Arah kedua menyangkut jenis *noise* yang lebih representatif terhadap fenomena fisis MRI. Pada Bab 1 telah disebutkan bahwa *Rician Noise* merupakan gangguan yang khas pada citra *magnitude* MRI dan secara dominan memengaruhi area berintensitas rendah, namun jenis *noise* ini belum diuji secara langsung dalam pembahasan ini. Selain itu, citra klinis kerap terpapar artefak gerakan akibat pergerakan pasien selama pemindaian, ketidakhomogenan medan magnet, serta gangguan sinyal yang bergantung pada intensitas. Temuan pembahasan ini bahwa *Speckle Noise* yang bersifat multiplikatif memberikan tantangan terberat bagi CAE menjadi indikasi penting bahwa *noise* yang berstruktur kompleks dan menyatu dengan informasi asli citra memerlukan pendekatan yang lebih khusus. Oleh karena itu, pembahasan lanjutan perlu menguji ketahanan model terhadap *noise* majemuk, yaitu kombinasi beberapa jenis gangguan yang hadir secara bersamaan dengan proporsi yang berbeda-beda.

Arah ketiga berkaitan dengan metodologi pelatihan yang lebih dekat dengan kondisi nyata. Salah satu kendala fundamental dalam pemrosesan citra medis adalah ketiadaan citra acuan yang benar-benar bersih, karena dalam praktik klinis hampir mustahil memperoleh pasangan citra terdegradasi dan citra ideal dari subjek yang sama. Untuk mengatasi keterbatasan ini, pengembangan ke depan dapat mengeksplorasi paradigma pembelajaran mandiri seperti

Noise2Noise dan *Noise2Void*, yang memungkinkan model dilatih tanpa memerlukan citra acuan bersih. Selain itu, teknik adaptasi domain dapat dimanfaatkan untuk menjembatani perbedaan distribusi antara *noise* sintetis yang digunakan saat pelatihan dan *noise* aktual yang dihadapi pada data klinis, sehingga generalisasi model terhadap kondisi *real-world* dapat ditingkatkan secara signifikan.

➢ Penerapan pada Jenis Citra Medis Lainnya (CT, *X-Ray*)

Arsitektur CAE yang dikembangkan dalam pembahasan ini, dengan struktur *encoder* yang memampatkan citra menjadi representasi laten dan *decoder* yang merekonstruksinya kembali, pada dasarnya bersifat generik dan tidak terikat secara eksklusif pada modalitas MRI. Sifat ini membuka peluang konseptual untuk menerapkan pendekatan serupa pada modalitas citra medis lain, khususnya *Computed Tomography* (CT) dan *X-Ray*, yang juga menghadapi persoalan degradasi kualitas citra. Meskipun demikian, transferabilitas tersebut tidak dapat diasumsikan berjalan otomatis, sebab setiap modalitas memiliki karakteristik fisis dan profil noise yang berbeda secara mendasar.

Pada citra CT, gangguan dominan umumnya bersumber dari noise kuantum yang mengikuti distribusi *Poisson*, terutama pada pencitraan CT dosis rendah, serta artefak garis (*streak artifact*) yang muncul akibat keterbatasan rekonstruksi. Sementara itu, citra *X-Ray* kerap dipengaruhi oleh quantum mottle dan hamburan radiasi (*scatter*) yang menurunkan kontras. Karakteristik gangguan ini berbeda dari noise impulsif seperti Salt dan Pepper maupun noise multiplikatif seperti *Speckle* yang menjadi fokus pembahasan ini. Perbedaan ini mengisyaratkan bahwa model yang telah dilatih pada citra MRI tidak dapat langsung digunakan untuk modalitas lain tanpa

penyesuaian, melainkan memerlukan pelatihan ulang atau penalaan halus (*fine-tuning*) menggunakan data yang representatif terhadap modalitas yang bersangkutan.

Secara metodologis, pendekatan pembelajaran transfer (*transfer learning*) menawarkan jalur yang menjanjikan, di mana representasi fitur yang telah dipelajari model pada satu modalitas dapat dijadikan landasan awal untuk modalitas lain, sehingga mempercepat konvergensi dan mengurangi kebutuhan data berlabel dalam jumlah besar. Selain itu, perbedaan resolusi spasial, rentang intensitas, dan kontras antarmodalitas menuntut adaptasi arsitektural, misalnya penyesuaian jumlah lapisan konvolusi, ukuran kernel, atau dimensi ruang laten agar sesuai dengan kompleksitas struktur anatomi yang direpresentasikan. Pertanyaan ilmiah yang relevan untuk diteliti lebih lanjut adalah sejauh mana sebuah arsitektur *denoising* tunggal mampu menggeneralisasi lintas modalitas, dan sejauh mana diperlukan model spesifik untuk masing-masing modalitas.

Implikasi klinis dari perluasan ini cukup substansial. Apabila pendekatan CAE terbukti dapat digeneralisasi, maka *denoising* berbasis pembelajaran mendalam dapat berperan sebagai tahap prapemrosesan universal yang meningkatkan kualitas citra pada berbagai modalitas diagnostik. Manfaat ini akan terasa paling nyata pada pencitraan CT dosis rendah, di mana reduksi noise yang efektif memungkinkan penurunan dosis radiasi tanpa mengorbankan kualitas diagnostik, sehingga turut berkontribusi pada keselamatan pasien.

➢ Kombinasi dengan Teknik Lain Seperti VAE atau GAN

Keterbatasan CAE konvensional yang teridentifikasi dalam pembahasan ini, khususnya kecenderungan kehilangan detail halus pada tingkat *noise* yang tinggi serta selisih performa

terhadap *U-Net*, sebagian besar berakar pada karakter arsitekturnya. CAE standar memetakan citra ke ruang laten secara deterministik dan dioptimasi terutama dengan fungsi *Loss* berbasis *Mean Squared Error* (MSE), yang secara inheren cenderung menghasilkan rekonstruksi yang halus namun kurang tajam pada batas-batas struktur. Oleh karena itu, integrasi CAE dengan arsitektur generatif yang lebih mutakhir menjadi arah pengembangan yang sangat relevan.

Variational Autoencoder (VAE) menawarkan perbaikan melalui pemodelan ruang laten secara probabilistik, bukan deterministik. Dengan memaksakan struktur distribusi tertentu pada ruang laten, VAE memperoleh kemampuan regularisasi yang lebih baik dan berpotensi menghasilkan generalisasi yang lebih kokoh, terutama pada kasus dengan tingkat *noise* yang tinggi sebagaimana ditunjukkan oleh kesulitan model terhadap *Speckle Noise*. Eksplorasi VAE diharapkan mampu menghasilkan representasi laten yang lebih bermakna dan tahan terhadap variasi gangguan, sehingga rekonstruksi yang dihasilkan lebih konsisten lintas jenis *noise*.

Generative Adversarial Network (GAN) menyediakan mekanisme yang berbeda melalui pelatihan adversarial antara jaringan generator dan diskriminator. Melalui fungsi *Loss* adversarial, GAN mendorong generator untuk menghasilkan citra yang secara perseptual lebih tajam dan realistis, yang berpotensi menutup celah dalam mempertahankan detail anatomi, yaitu aspek yang menjadi keunggulan *U-Net* berkat *skip-connection*. Pendekatan terkondisi seperti *conditional GAN* dapat secara langsung dilatih untuk memetakan citra ber-*noise* menjadi citra bersih. Lebih lanjut, hibridisasi dapat ditempuh dengan menyisipkan mekanisme atensi, pembelajaran residual, ataupun *skip-connection* ke dalam kerangka CAE, sehingga arsitektur secara bertahap mendekati keunggulan struktural *U-*

Net. Pada dimensi evaluasi, perlu dipertimbangkan pula penggunaan metrik dan fungsi *Loss* perseptual seperti *Structural Similarity Index* (SSIM) atau *feature Loss*, mengingat MSE dan PSNR yang digunakan dalam pembahasan ini belum sepenuhnya mencerminkan kualitas visual sebagaimana dipersepsi oleh pengamat manusia.

Sekalipun menjanjikan, pemanfaatan model generatif dalam konteks medis menuntut kehati-hatian yang tinggi. Model seperti GAN memiliki risiko menghasilkan halusinasi, yaitu pembentukan struktur yang tampak meyakinkan namun tidak benar-benar terdapat pada citra asli. Dalam pencitraan medis, risiko ini berimplikasi serius karena dapat memunculkan atau menghilangkan fitur patologis yang menentukan keputusan diagnosis. Selain itu, pelatihan GAN dikenal rentan terhadap ketidakstabilan konvergensi. Konsekuensinya, setiap pengembangan ke arah ini wajib disertai mekanisme pengendalian fidelitas terhadap citra asli, validasi klinis yang ketat, serta penguatan aspek interpretabilitas melalui pendekatan *Explainable AI* (XAI) agar keluaran model dapat dipertanggungjawabkan secara medis dan etis.

- **Pengembangan Dataset Berskala Besar dan Anotasi Medis**

Kualitas dan cakupan dataset merupakan determinan utama dalam keberhasilan model pembelajaran mendalam. Dataset yang digunakan dalam pembahasan ini terdiri atas 434 citra MRI otak normal yang bersumber dari satu repositori publik, dengan distribusi antarbidang yang tidak seimbang, yaitu didominasi bidang aksial, diikuti bidang sagital, dan paling sedikit bidang koronal. Ketidakseimbangan ini terbukti memengaruhi konsistensi kualitas rekonstruksi antarbidang,

sebagaimana ditunjukkan oleh analisis per kelas pada bab sebelumnya. Selain itu, dataset hanya memuat citra otak normal tanpa kasus patologis, sehingga kemampuan model dalam mempertahankan fitur abnormal belum dapat diuji.

Keterbatasan tersebut membawa implikasi penting terhadap validitas klinis. Sebuah model *denoising* yang ideal tidak cukup hanya mampu memulihkan struktur anatomi normal, tetapi juga harus mempertahankan secara akurat fitur patologis seperti tumor, lesi, maupun anomali jaringan, justru karena fitur-fitur inilah yang menjadi sasaran utama diagnosis. Pelatihan yang semata-mata didasarkan pada citra normal berisiko menghasilkan model yang menggeneralisasi pola jaringan sehat namun lalai mempertahankan, atau bahkan menghaluskan, struktur abnormal yang bernilai diagnostik tinggi.

Atas dasar itu, arah pengembangan dataset di masa depan perlu diarahkan pada pembangunan kumpulan data berskala besar yang bersumber dari berbagai institusi, mencakup distribusi bidang irisan yang seimbang, serta memuat baik kasus normal maupun beragam kondisi patologis. Idealnya, dataset semacam ini disertai pasangan citra ber-*noise* dan citra bersih yang nyata, bukan sekadar noise sintetis, agar model belajar dari karakteristik degradasi yang autentik. Anotasi oleh tenaga medis ahli menjadi komponen yang tidak tergantikan, baik untuk menandai struktur anatomi dan patologi maupun untuk keperluan validasi klinis. Ketersediaan tolok ukur (*benchmark*) yang terstandardisasi juga akan memungkinkan perbandingan performa antarmetode dilakukan secara adil dan dapat direproduksi.

Pengumpulan data medis berskala besar tidak dapat dilepaskan dari pertimbangan tata kelola, privasi, dan keamanan data pasien. Sentralisasi data dari banyak institusi

menimbulkan tantangan etis dan regulatif yang signifikan. Sebagai jalan keluar, pendekatan pembelajaran terfederasi (*federated learning*) dapat dipertimbangkan, karena memungkinkan pelatihan model secara kolaboratif lintas institusi tanpa harus memindahkan data pasien dari lokasi asalnya, sehingga kerahasiaan data tetap terjaga sekaligus keragaman data tetap diperoleh. Pendekatan ini menyelaraskan kebutuhan akan dataset yang luas dengan tuntutan perlindungan data yang ketat dalam domain kesehatan.

Secara keseluruhan, keempat dimensi yang diuraikan dalam bab ini membentuk satu lintasan pengembangan yang koheren, yaitu pergeseran dari pembuktian konsep dalam kondisi eksperimental yang terkendali menuju sistem *denoising* yang kokoh, dapat digeneralisasi lintas modalitas, dan layak dipercaya untuk penerapan klinis. Penanganan noise yang lebih kompleks memperkuat relevansi model terhadap kondisi nyata, perluasan ke modalitas lain memperluas dampaknya, integrasi dengan arsitektur generatif meningkatkan kualitas rekonstruksinya, dan pengembangan dataset yang kredibel menjamin validitas serta keterpercayaannya. Realisasi arah-arah pembahasan ini menuntut kolaborasi yang erat antara ilmuwan komputer, insinyur, dan tenaga medis, sebagaimana ditekankan sejak awal buku ini, agar inovasi pemrosesan citra medis benar-benar dapat diterjemahkan menjadi peningkatan kualitas pelayanan kesehatan yang nyata, akurat, dan aman bagi pasien.

DAFTAR PUSTAKA

[1] I. W. A. Wijaya Kusuma dan A. Kusumadewi, "Penerapan Metode Contrast Stretching, Histogram Equalization Dan Adaptive Histogram Equalization Untuk Meningkatkan Kualitas Citra Medis Mri," Simetris J. Tek. Mesin, Elektro dan Ilmu Komput., vol. 11, no. 1, hal. 1–10, 2020, doi: 10.24176/simet.v11i1.3153.

[2] J. Shedbalkar, K. Prabhushetty, dan A. Inchalc, "A Comparative Analysis of Filters for Noise Reduction and Smoothening of Brain MRI Images," 2021 6th Int. Conf. Converg. Technol. I2CT 2021, hal. 1–6, 2021, doi: 10.1109/I2CT51068.2021.9417979.

[3] M. B. Darici dan Z. Erdem, "A Comparative Study on Denoising from Facial Images Using Convolutional Autoencoder," Gazi Univ. J. Sci., vol. 36, no. 3, hal. 1122–1138, 2023, doi: 10.35378/gujs.1051655.

[4] S. Li, J. Zhou, D. Liang, dan Q. Liu, "MRI denoising using progressively distribution-based neural network," Magn. Reson. Imaging, vol. 71, no. July 2019, hal. 55–68, 2020, doi: 10.1016/j.mri.2020.04.006.

[5] W. El-Shafai dkk., "Efficient deep-learning-based autoencoder denoising approach for medical image diagnosis," Comput. Mater. Contin., vol. 70, no. 3, hal. 6107–6125, 2022, doi: 10.32604/cmc.2022.020698.

[6] C. Janiesch, P. Zschech, dan K. Heinrich, "Machine learning and deep learning," Electron. Mark., vol. 31, no. 3, hal. 685–695, Sep 2021, doi: 10.1007/s12525-021-00475-2.

[7] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, dan J. Peter Campbell, "Introduction to machine learning, neural networks, and deep learning," Transl. Vis. Sci. Technol., vol. 9, no. 2, hal. 1–12, 2020, doi: 10.1167/tvst.9.2.14.

[8] A. Khan, A. Sohail, U. Zahoora, dan A. S. Qureshi, A survey of the recent architectures of deep convolutional neural networks, vol. 53, no. 8. Springer Netherlands, 2020. doi: 10.1007/s10462-020-09825-6.

[9] G. Lin, "2D image edge detection enhancement using convolutional neural network," Appl. Comput. Eng., vol. 2, no. 1, hal. 779–786, Mar 2023, doi: 10.54254/2755-2721/2/20220519.

[10] L. Zhang dkk., "Structure-Feature based Graph Self-adaptive Pooling," in Proceedings of The Web Conference 2020, New York, NY, USA: ACM, Apr 2020, hal. 3098–3104. doi: 10.1145/3366423.3380083.

[11] Z. J. Wang dkk., "CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization," IEEE Trans. Vis. Comput. Graph., vol. 27, no. 2, hal. 1396–1406, 2021, doi: 10.1109/TVCG.2020.3030418.

[12] A. Ghafar dan U. Sattar, "Convolutional Autoencoder for Image Denoising," UMT Artif. Intell. Rev., vol. 1, no. 2, hal. 1–11, 2021, doi: 10.32350/air.0102.01.

[13] A. Higaki dkk., "Content-based image retrieval for the diagnosis of myocardial perfusion imaging using a deep convolutional autoencoder," J. Nucl. Cardiol., vol. 30, no. 2, hal. 540–549, Apr 2023, doi: 10.1007/s12350-022-03030-4.

[14] Y. FAROOQ dan S. SAVAŞ, "Noise Removal from the Image Using Convolutional Neural Networks-Based Denoising Auto Encoder," J. Emerg. Comput. Technol., vol. 3, no. 1, hal. 21–28, 2024, doi: 10.57020/ject.1390428.

[15] Patrisius Batarius, Alfry Aristo Sinlae, dan Elisabeth F. Fahik, "Analysis of the Quality of Natural Dyes in Weaving Exposed to Sunlight Using MSE and PSNR Parameters," J. RESTI (Rekayasa Sist. dan Teknol. Informasi), vol. 6, no. 5, hal. 797–802, Okt 2022, doi: 10.29207/resti.v6i5.4339.

[16] Y. Y. Al-Aboosi, R. S. Issa, dan A. Khalid Jassim, "Image denosing in underwater acoustic noise using discrete wavelet transform with different noise level estimation," Telkomnika (Telecommunication Comput. Electron. Control., vol. 18, no. 3, hal. 1439–1446, 2020, doi: 10.12928/TELKOMNIKA.V18I3.14381.

[17] N. Weiskopf, L. J. Edwards, G. Helms, S. Mohammadi, dan E. Kirilina, "Quantitative magnetic resonance imaging of brain anatomy and in vivo histology," Nat. Rev. Phys., vol. 3, no. 8, hal. 570–588, 2021, doi: 10.1038/s42254-021-00326-1.

[18] P. Sriramakrishnan, T. Kalaiselvi, M. Thirumalaiselvi, S. T. Padmapriya, dan S. Ramkumar, "A Role of Medical Imaging Techniques in Human Brain Tumor Treatment," Int. J. Recent Technol. Eng., vol. 8, no. 4S2, hal. 565–568, 2019, doi: 10.35940/ijrte.d1105.1284s219.

[19] P. Bedi, S. B. Goyal, D. K. Yadav, S. Kumar, dan M. Sharma, "Hybrid Learning Model for Metal Artifact Reduction," J. Phys. Conf. Ser., vol. 1714, no. 1, hal. 012021, Jan 2021, doi: 10.1088/1742-6596/1714/1/012021.

[20] P. Mona, "Implementasi Metode Bilateral Filter Untuk Mengurangi Derau Pada Citra Magnetic Resonance Imaging (MRI)," J. Inf. dan Teknol. Ilm., vol. 7, no. 3, hal. 259–263, 2020.

[21] D. N. H. Thanh, N. H. Hai, V. B. S. Prasath, L. M. Hieu, dan J. M. R. S. Tavares, "A two-stage filter for high density salt and pepper denoising," Multimed. Tools Appl., vol. 79, no. 29–30, hal. 21013–21035, Agu 2020, doi: 10.1007/s11042-020-08887-6.

[22] Z. Jeelani dan F. Qadir, "Cellular automata-based approach for salt-and-pepper noise filtration," J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 2, hal. 365–374, Feb 2022, doi: 10.1016/j.jksuci.2018.12.006.

[23] A. Halder, R. Choudhuri, P. Bhattacharya, dan A. Sarkar, "A Novel Statistical High Density Salt-and-Pepper Noise Removal Algorithm for Brain Magnetic Resonance Images," in Proceedings of the Thirteenth Indian Conference on Computer Vision, Graphics and Image Processing, New York, NY, USA: ACM, Des 2022, hal. 1–9. doi: 10.1145/3571600.3571631.

[24] B. Hemalatha, B. Karthik, dan C. V Krishna Reddy, "Speckle Noise Removal from Biomedical MRI Images and Classification by Multi-Support Vector Machine," EAI Endorsed Trans. Pervasive Heal. Technol., vol. 10, Feb 2024, doi: 10.4108/eetpht.10.5076.

[25] M. R. Ziemann, R. R. Joshi, dan C. A. Metzler, "Time-varying implicit neural representations for unsupervised speckle denoising in dynamic scenes," in Unconventional Imaging, Sensing, and Adaptive Optics 2024, S. R. Bose-Pillai, J. J. Dolne, dan M. Kalensky, Ed., SPIE, Okt 2024, hal. 12. doi: 10.1117/12.3028176.

[26] M. Ghaffar, R. Fatima, dan W. Bashir, "Deep Learning based Enhanced Adaptive Despeckling for Ultrasound Images," Pakistan J. Sci. Res., vol. 3, no. 1, hal. 14–19, Okt 2023, doi: 10.57041/pjosr.v3i1.954.

[27] and G. N. Kujawa, S., "Kujawa, S., and Niedbała, G. Artificial Neural Networks in Agriculture. Agriculture 2021, 11, 497.pdf," Agriculture, hal. 1–6, 2021.

[28] S. Volkova, "An Overview on Data Augmentation for Machine Learning," 2024, hal. 143–154. doi: 10.1007/978-3-031-55349-3_12.

[29] X. Wang, Y. Hua, E. Kodirov, D. A. Clifton, dan N. M. Robertson, "IMAE for Noise-Robust Learning: Mean Absolute Error Does Not Treat Examples Equally and Gradient Magnitude's Variance Matters," hal. 1–19, 2019, [Daring]. Tersedia pada: http://arxiv.org/abs/1903.12141

[30] V. Karpukhin, O. Levy, J. Eisenstein, dan M. Ghazvininejad, "Training on synthetic noise improves robustness to natural noise in machine translation," W-NUT@EMNLP 2019 - 5th Work. Noisy User-Generated Text, Proc., hal. 42–47, 2019, doi: 10.18653/v1/d19-5506.

[31] V. R. Joseph, "Optimal ratio for data splitting," Stat. Anal. Data Min., vol. 15, no. 4, hal. 531–538, 2022, doi: 10.1002/sam.11583.

[32] G. P. Kumar, G. S. Priya, M. Dileep, B. E. Raju, A. R. Shaik, dan K. V. S. H. G. Sarman, "Image Deconvolution using Deep Learning-based Adam *Optimizer*," in 2022 6th International Conference on Electronics, Communication and Aerospace Technology, IEEE, Des 2022, hal. 901–904. doi: 10.1109/ICECA55336.2022.10009073.

[33] D. R. Newlin dan C. Seldev Christopher, "Medical image denoising using different techniques," Int. J. Sci. Technol. Res., vol. 9, no. 3, hal. 1061–1066, 2020.

[34] U. Kulkarni dkk., "Image Denoising using Autoencoders : Denoising noisy imgaes by removing noisy pixels/grains from natural images using Deep learning and autoencoders techniques," in 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), IEEE, Apr 2023, hal. 1–6. doi: 10.1109/I2CT57861.2023.10126382.

BIODATA PENULIS

Dr. Ir. I Gede Susrama Mas Diyasa, ST., MT., IPU

Penulis merupakan dosen di Universitas Pembangunan Nasional "Veteran" Jawa Timur adengan pakar di bidang *Image Processing*, Biomedik, dan *Computer Vision*. Selain menjabat sebagai Wakil Dekan 1, beliau juga merupakan dosen pengajar di Program Studi Magister Teknologi Informasi. Beliau menyelesaikan pendidikan pada jenjang S1 di Institut Teknologi Adhi Tama Surabaya serta S2 dan S3 di Institut Teknologi Sepuluh Nopember.