BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Kajian pustaka ini berfokus pada beberapa penelitian terdahulu yang relevan dalam bidang klasifikasi tumor otak berbasis citra MRI, dengan perhatian khusus pada studi yang menggunakan dataset *Brain Tumor Classification* oleh Sartaj Bhuvaji (2020). Dataset ini dipilih karena banyak digunakan sebagai standar pembanding pada berbagai penelitian terkini, sehingga memungkinkan evaluasi kinerja model secara setara dan konsisten. Berbagai penelitian menunjukkan bahwa arsitektur *deep learning* seperti VGG, ResNet, DenseNet, dan EfficientNet mampu mencapai akurasi hingga 96% ketika diaplikasikan pada dataset tersebut. Temuan ini sekaligus mengonfirmasi dominasi pendekatan *deep learning* dibandingkan metode *machine learning* konvensional, yang umumnya hanya mencapai akurasi 94%.

Meskipun sebagian besar penelitian menitikberatkan pada perbandingan arsitektur model, aspek optimasi *hyperparameter* masih belum banyak dieksplorasi secara sistematis. Sebagian besar studi hanya menggunakan konfigurasi *default* tanpa membandingkan metode optimasi yang berbeda, sehingga kesenjangan inilah yang menjadi dasar penelitian ini: melakukan evaluasi komprehensif terhadap empat metode *hyperparameter tuning* untuk menentukan konfigurasi parameter terbaik pada arsitektur EfficientNetB5 dengan dataset *Brain Tumor Classification*. Pendekatan ini tidak hanya menawarkan kontribusi empiris baru, tetapi juga memberikan analisis yang lebih lengkap mengenai dampak metode optimasi terhadap stabilitas, efisiensi waktu eksekusi, serta kualitas generalisasi model.

Untuk menggambarkan posisi penelitian ini dalam konteks literatur yang ada, uraian singkat dari hasil penelitian terdahulu disajikan pada Tabel 2. 1.

Tabel 2. 1 Tinjauan Pustaka Penelitian Terkait

No	Judul	Metode dan Penggunaan	Hasil Penelitian	Perbedaan dengan Penelitian Ini
1	*Combination of Political Optimizer, Particle Swarm Optimizer, and Convolutional Neural Network for Brain Tumor Detection* (Bashkandi dkk., 2023)	Xception berupa 20 *layer*, optimasi menggunakan *Political Optimization* (PO) dan *Particle Swarm Optimization* (PSO)	Menghasilkan akurasi, presisi, sensitivitas, spesifisitas, *f1-score*, dan *Jaccard Index* sebesar 97,09%, 97,25%, 98,04%, 96,07%, 92,97%, 91,99%	Kombinasi *Political Optimization* (PO) dan *Particle Swarm Optimization* (PSO) digunakan untuk mengoptimalkan arsitektur model secara menyeluruh.
2	*An Efficient Brain Tumor Detection and Classification Using Pre-Trained Convolutional Neural Network Models* (Rao dkk., 2024)	ResNet50 dan EfficientNet untuk deteksi dan klasifikasi tumor otak dari citra MRI	ResNet50 menghasilkan akurasi sebesar 96%, EfficientNet menghasilkan akurasi sebesar 98%, *precision* 98,5% dan *recall* 99%.	Menggunakan 2 arsitektur model CNN berupa ResNet50 dan EfficientNet untuk melakukan klasifikasi citra
3	*Deep Learning and Optimized Learning Machine for Brain Tumor Classification* (Sandhiya & Kanaga Suba Raja, 2024)	DenseNet201 dan InceptionV3	*Dataset* 1 dari Kaggle dengan akurasi sebesar 97,97%. *Dataset* 2 dari Hospital China dengan akurasi *training* sebesar 98,21%	Menggabungkan 2 arsitektur, proses klasifikasi menggunakan *Kernel Extreme Learning Machine* (KELM) yang dioptimasi menggunakan *Particle Swarm Optimization* (PSO)
4	*Development of Hybrid Models Based on Deep Learning and Optimized Machine Learning Algorithms for Brain Tumor Multi Classification* (Celik & Inik, 2024)	Kombinasi metode *Support Vector Machine* (*SVM*) dengan beberapa arsitektur model CNN	EfficientNetB0 menghasilkan akurasi 97,93%. Model *hybrid* (CNN-KNN) sebesar 97,15% dan waktu eksekusi 67 menit.	Menggabungkan metode *SVM* dengan arsitektur model CNN untuk klasifikasi multi-kategori pada penyakit tumor otak.
5	*Brain Tumor Classification using MRI Images and Deep CNNs* (Wong dkk., 2025)	ResNet50 dan VGG16	Akurasi 94.50% (ResNet50) dan 94.20% (VGG16).	Menggunakan dataset Sartaj Bhuvaji, fokus pada arsitektur ResNet & VGG.

Tabel 2. 1 Tinjauan Pustaka Penelitian Terkait (Lanjutan)

No	Judul	Metode dan Penggunaan	Hasil Penelitian	Perbedaan dengan Penelitian Ini
6	*A Transfer Learning-Based Model for Brain Tumor Detection in MRI Images* (Hencya dkk., 2023)	EfficientNetB3 dan VGG19	Akurasi 95.00%.	Menggunakan dataset yang sama (Sartaj Bhuvaji), berfokus pada perbandingan B3 dengan VGG19.
7	*MobDenseNet: A Hybrid Deep Learning Model for Brain Tumor Classification Using MRI* (Afroj dkk., 2025)	Model *hybrid* kustom (MobDenseNet).	Akurasi 96.10%.	Menggunakan dataset Sartaj Bhuvaji, serta mengusulkan arsitektur *hybrid* baru.
8	*Addressing The Role and Opportunities of Machine Learning Utilization in Brain Tumor Detection* (Lesmana dkk., 2024)	VGG16, InceptionV3, ResNet50, dan DenseNet121	VGG16 dengan akurasi sebesar 96.43%, InceptionV3 sebesar 92.40%, DenseNet121 sebesar 94.96%, dan ResNet50 sebesar 78.69%	Hanya membandingkan 4 arsitektur model CNN yang telah diusulkan dengan kemungkinan terjadinya *overfitting*
9	*Brain Tumor Classification using Machine Learning and Deep Learning Algorithms: A Comparison: Classifying brain MRI images on the basis of location of tumor and comparing the various Machine Learning and Deep Learning models used to predict best performance.* (Joshi dkk., 2022)	SVM dan CNN	Akurasi 93.60%	Menggunakan dataset Sartaj Bhuvaji dan membandingkan ML klasik (SVM) dengan DL.

Tabel 2. 1 Tinjauan Pustaka Penelitian Terkait (Lanjutan)

No	Judul	Metode dan Penggunaan	Hasil Penelitian	Perbedaan dengan Penelitian Ini
10	*Optimized Attention-Based Lightweight CNN Using Particle Swarm Optimization for Brain Tumor Classification* (Guder & Cetin-Kaya, 2025)	ChannelNet, SpatialNet, CbamNet. *hyperparameter tuning* dengan PSO	ChannelNet, SpatialNet, CbamNet sebesar 98,01%, 99,00%, dan 98,16%	Optimasi *hyperparameter* khusus menggunakan *Particle Swarm Optimization* (PSO), selain itu menggunakan arsitektur ringan.
11	Peningkatan Performa Model EfficientNetB5 Melalui Optimasi *Hyperparameter* Untuk Klasifikasi Tumor Otak	EfficientNetB5, *Bayesian Optimization*, PSO, *Grid Search*, dan *Random Search*	*Bayesian Optimization* dengan akurasi tertinggi sebesar 96,55%	Tidak hanya membandingkan metode *hyperparameter tuning* adaptif dengan konvensional, melainkan menganalisis stabilitas performa, efisiensi waktu, dampak augmentasi, dan pola kesalahan prediksi.

Penelitian oleh Bashkandi dkk. (2023) menggabungkan dua metode optimasi, *Political Optimization* (PO) dan PSO, untuk mengoptimalkan arsitektur model Xception yang terdiri dari 20 *layer* dalam tugas deteksi tumor otak. Hasil yang diperoleh menunjukkan performa yang sangat baik, dengan tingkat akurasi mencapai 97,09%, presisi 97,25%, sensitivitas 98,04%, dan *f1-score* 92,97%. Pendekatan ini mengungguli algoritma *baseline* lainnya, yang menunjukkan bahwa kombinasi *metaheuristic optimization* seperti PO dan PSO dapat secara signifikan meningkatkan performa model dibandingkan dengan teknik optimasi konvensional. Penelitian ini memberikan kontribusi penting dalam menggunakan metode optimasi gabungan untuk meningkatkan akurasi model *deep learning* dalam klasifikasi citra medis.

Penelitian oleh Rao dkk. (2024) membandingkan dua arsitektur model CNN yang populer, yaitu ResNet50 dan EfficientNet, untuk deteksi dan klasifikasi tumor

otak dari citra MRI. Kedua model ini diuji dengan menggunakan augmentasi data untuk meningkatkan kinerja model. Hasil yang diperoleh menunjukkan bahwa EfficientNet menghasilkan akurasi 98%, dengan presisi 98,5% dan *recall* 99%, yang menunjukkan performa superior dibandingkan dengan ResNet50 yang memperoleh akurasi 96%. Penelitian ini menunjukkan keunggulan EfficientNet dalam mengklasifikasikan tumor otak secara akurat, serta memberikan *insight* penting terkait pemilihan arsitektur model yang tepat dalam aplikasi medis berbasis *deep learning*.

Penelitian oleh Sandhiya & Kanaga Suba Raja (2024) menggabungkan dua arsitektur DenseNet201 dan InceptionV3 dengan teknik *Discrete Cosine Transform* (DCT) untuk ekstraksi fitur yang lebih beragam pada citra MRI tumor otak. Model klasifikasi yang digunakan adalah *Kernel Extreme Learning Machine* (KELM) yang dioptimasi menggunakan PSO. Pengujian dilakukan pada dua dataset, yaitu dataset dari Kaggle dan Hospital China. Hasil yang diperoleh menunjukkan bahwa akurasi pada *training* dan *testing* pada dataset Kaggle adalah 96,17% dan 97,97%, sementara pada dataset Hospital China, akurasi *training* dan *testing* mencapai 97,92% dan 98,21%. Penelitian ini menunjukkan bahwa gabungan arsitektur *deep learning* dengan teknik PSO dapat meningkatkan akurasi klasifikasi tumor otak, dengan keunggulan pada ekstraksi fitur yang lebih kaya dan proses klasifikasi yang lebih akurat.

Penelitian oleh Celik & Inik (2024) mengembangkan model *hybrid* dengan menggabungkan *Support Vector Machine* (SVM) dan berbagai arsitektur model CNN, termasuk DarkNet19-SVM, DarkNet53-SVM, DenseNet201-SVM, EfficientNetB0-SVM, InceptionV3-SVM, NasNetMobile-SVM, ResNet50-SVM, ResNet101-SVM, dan Xception-SVM. Penelitian ini bertujuan untuk melakukan klasifikasi *multi-class* tumor otak. Hasil yang diperoleh menunjukkan akurasi yang bervariasi tergantung pada kombinasi metode yang digunakan, dengan waktu pelatihan rata-rata sekitar 67 menit. Namun, arsitektur InceptionV3 membutuhkan waktu pelatihan yang jauh lebih lama, yaitu 370 menit. Pendekatan ini menunjukkan bahwa kombinasi SVM dengan berbagai model CNN dapat

memperbaiki kinerja klasifikasi tumor otak, meskipun ada variasi dalam waktu yang dibutuhkan untuk pelatihan model.

Penelitian oleh Wong dkk. (2025) menggunakan dua arsitektur CNN yang populer, yaitu ResNet50 dan VGG16, untuk mengklasifikasikan tumor otak berdasarkan citra MRI dari dataset Sartaj Bhuvaji. Hasil yang diperoleh menunjukkan akurasi 94,50% untuk ResNet50 dan 94,20% untuk VGG16. Penelitian ini fokus pada penerapan arsitektur ResNet dan VGG dalam konteks klasifikasi tumor otak, yang merupakan pendekatan dasar namun kuat dalam bidang deteksi medis berbasis *deep learning*. Meski akurasinya sangat baik, hasil ini menunjukkan adanya ruang untuk peningkatan lebih lanjut, terutama dalam hal penerapan arsitektur yang lebih canggih seperti EfficientNet.

Penelitian oleh Hencya dkk. (2023) menggunakan *transfer learning* dengan model EfficientNetB3 dan VGG19 untuk mendeteksi tumor otak dari citra MRI menggunakan dataset Sartaj Bhuvaji. Hasil penelitian menunjukkan akurasi 95,00%, dengan perbandingan antara kedua model yang fokus pada kekuatan EfficientNetB3 dalam menangani tugas klasifikasi medis. Penelitian ini memberikan wawasan penting mengenai keunggulan EfficientNet dalam hal efisiensi dan akurasi dibandingkan dengan model VGG19, yang lebih umum digunakan dalam berbagai aplikasi klasifikasi citra medis. Meskipun akurasi yang dicapai cukup baik, penelitian ini juga menyoroti pentingnya pemilihan model yang tepat untuk mencapai optimalisasi lebih lanjut pada dataset tertentu.

Penelitian oleh Afroj dkk. (2025) mengusulkan model *hybrid* kustom, yang disebut MobDenseNet, untuk mengklasifikasikan tumor otak berdasarkan citra MRI dari dataset Sartaj Bhuvaji. MobDenseNet menggabungkan keunggulan arsitektur DenseNet dengan beberapa teknik optimasi untuk meningkatkan akurasi dan efisiensi model. Hasil penelitian menunjukkan akurasi 96,10%, yang menunjukkan bahwa model *hybrid* ini mampu mengungguli beberapa model standar dalam tugas klasifikasi tumor otak. Penelitian ini juga berfokus pada penerapan arsitektur baru yang dapat diadaptasi untuk meningkatkan kinerja model pada dataset medis yang terbatas, memberikan kontribusi baru dalam penggunaan model *hybrid* untuk deteksi medis.

Penelitian oleh Lesmana dkk. (2024) membandingkan empat arsitektur model CNN untuk deteksi tumor otak, yaitu VGG16, InceptionV3, ResNet50, dan DenseNet121. Hasil penelitian menunjukkan bahwa VGG16 mencapai akurasi 96,43%, sementara InceptionV3 memperoleh 92,40%, DenseNet121 mencapai 94,96%, dan ResNet50 hanya memperoleh 78,69%. Penelitian ini menunjukkan bahwa meskipun VGG16 memiliki performa terbaik di antara keempat arsitektur, ada kemungkinan *overfitting* pada beberapa model, terutama pada ResNet50 yang memiliki akurasi yang lebih rendah. Pendekatan ini memberikan gambaran yang jelas tentang perbandingan arsitektur CNN dalam aplikasi deteksi tumor otak, dengan menyoroti tantangan *overfitting* yang perlu diperhatikan dalam pengembangan model *deep learning* medis.

Penelitian oleh Guder & Cetin-Kaya (2025) mengusulkan penggunaan arsitektur *lightweight* CNN yang dioptimalkan dengan PSO untuk klasifikasi tumor otak. Tiga arsitektur yang diuji adalah ChannelNet, SpatialNet, dan CbamNet, yang dioptimasi untuk meningkatkan akurasi tanpa mengorbankan performa di perangkat dengan sumber daya terbatas. Hasil penelitian menunjukkan akurasi sebesar 98,01% untuk ChannelNet, 99,00% untuk SpatialNet, dan 98,16% untuk CbamNet saat diuji tanpa *noise*. Pendekatan ini menunjukkan bahwa PSO sangat efektif untuk *hyperparameter tuning*, serta arsitektur *lightweight* CNN memberikan hasil yang sangat baik dengan penggunaan sumber daya yang lebih efisien. Penelitian ini berkontribusi dalam pengembangan model yang diimplementasikan pada perangkat *edge computing* untuk deteksi tumor otak.

Penelitian oleh Joshi dkk. (2022) membandingkan metode *Machine Learning* (ML) klasik, yaitu *Support Vector Machine* (SVM), dengan *Deep Learning* (DL) menggunakan CNN untuk klasifikasi tumor otak berdasarkan lokasi tumor pada citra MRI. Penelitian ini menggunakan dataset Sartaj Bhuvaji dan memperoleh akurasi 93,60%. Hasil ini menunjukkan bahwa meskipun SVM memiliki akurasi yang baik, CNN cenderung lebih unggul dalam menangani kompleksitas citra medis dan dapat mengenali fitur lebih dalam yang membedakan berbagai jenis tumor. Penelitian ini memberikan wawasan tentang perbandingan

antara ML dan DL, serta menunjukkan bahwa *deep learning* memberikan hasil yang lebih baik dalam aplikasi klasifikasi tumor otak.

Berdasarkan tinjauan pustaka yang telah dipaparkan, berbagai penelitian terdahulu menunjukkan beragam pendekatan dalam klasifikasi tumor otak menggunakan teknik *deep learning* dan *machine learning*, dengan berbagai arsitektur model dan metode optimasi yang digunakan. Meskipun penelitian-penelitian ini memberikan kontribusi signifikan dalam klasifikasi tumor otak melalui citra MRI, setiap studi memiliki kelebihan dan kekurangan yang saling melengkapi. Oleh karena itu, dalam penelitian ini, berbagai temuan dari studi-studi terdahulu akan dijadikan landasan teori dan metodologi untuk memperkuat pendekatan yang digunakan, serta membantu mengidentifikasi inovasi dan perbaikan yang dilakukan, terutama dalam hal optimasi *hyperparameter* dan evaluasi performa.

2.2 Dasar Teori

2.2.1 Tumor Otak

Tumor otak adalah pertumbuhan sel di otak yang tidak normal dan tidak terkendali yang terjadi di dalam atau di sekitar otak. Sel-sel abnormal tersebut memiliki karakteristik yang berbeda dari sel normal. Tumor otak jinak terdiri dari sel yang tidak aktif, sedangkan tumor otak ganas terdiri dari sel kanker yang aktif membelah dan memiliki struktur heterogen. Tumor otak dapat dibedakan menjadi dua jenis: tumor otak primer dan metastasis. Tumor primer terbatas pada otak, sementara tumor metastasis melibatkan penyebaran sel kanker ke bagian tubuh lain di luar otak (Bouguerra dkk., 2024).

Beberapa lokasi yang bisa terkena tumor otak adalah jaringan saraf dalam otak (*Glioma*), kelenjar pituitari (*Pituitary*), dan selaput yang melindungi otak (*Meningioma*). *Glioma* adalah jenis tumor yang paling sering dialami oleh orang dewasa. Jenis ini berasal dari sel pendukung otak yang disebut *glia*. *Pituitary* adalah jenis tumor yang muncul di dalam kelenjar *pituitary*. Jenis ini tumbuh perlahan, dan biasanya tidak menyebar ke bagian tubuh lain. *Meningioma* adalah tumor jinak yang berada di selaput *meninges*, yaitu membran yang melindungi otak dan sumsum tulang belakang (Nada Nafisa dkk., 2023).

2.2.2 *Deep Learning* (DL)

DL merupakan bagian dari *neural computation* yang menggunakan jaringan saraf tiruan dengan beberapa lapis untuk mempelajari representasi fitur secara otomatis dari data berstruktur maupun tidak berstruktur. Berbeda dari pendekatan *shallow learning* yang masih bergantung pada rekayasa fitur manual, DL mampu mengekstraksi pola kompleks secara hierarkis, mulai dari pola sederhana seperti tepi dan tekstur hingga pola semantik tingkat tinggi yang berkaitan dengan bentuk objek dan struktur anatomis (Taqiyuddin, 2023). Pendekatan ini menjadikan DL sangat efektif untuk domain yang memiliki karakteristik visual kompleks, termasuk citra medis.

Pada ranah pengolahan citra medis, CNN menjadi arsitektur yang paling dominan digunakan. CNN mampu mengekstrak fitur visual melalui operasi konvolusi yang mempertahankan informasi spasial, sehingga memungkinkan model mengenali pola patologis secara lebih presisi. Studi-studi terkini menunjukkan bahwa arsitektur DL modern seperti ResNet, DenseNet, dan EfficientNet terus meningkatkan performa pada klasifikasi citra MRI, sekaligus mengurangi kesalahan diagnostik yang sering ditemui pada metode konvensional (Mienye dkk., 2025).

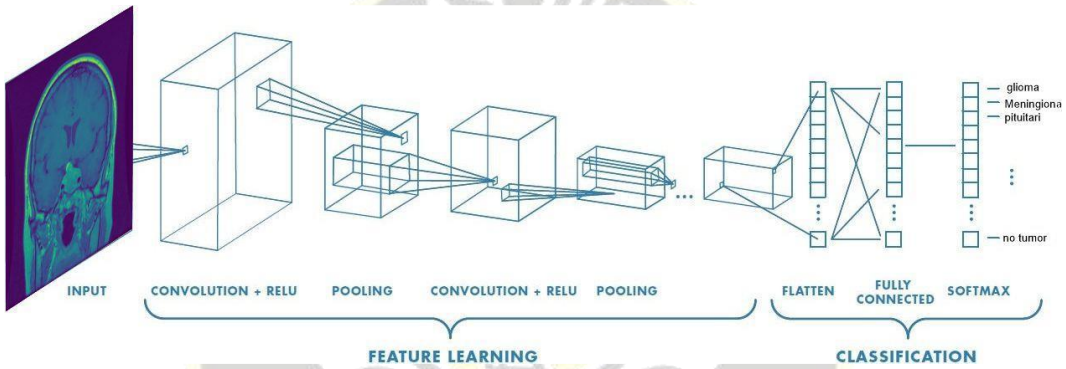
Selain keunggulan akurasi, DL modern juga menyoroti sejumlah tantangan teknis. Diantaranya adalah kebutuhan data berlabel dalam jumlah besar, variasi kualitas citra antar perangkat MRI, risiko *overfitting*, serta keterbatasan interpretabilitas model yang sering kali dianggap "*black box*". Untuk mengatasi masalah tersebut, berbagai pendekatan kontemporer dikembangkan, seperti augmentasi data, optimasi *hyperparameter*, serta teknik pembelajaran hemat label seperti *semi-supervised learning* dan *self-supervised learning* (Jin dkk., 2025). Teknik interpretabilitas seperti Grad-CAM juga semakin umum digunakan untuk menyediakan visualisasi berbasis perhatian sehingga keputusan model dapat divalidasi secara klinis (Patrício dkk., 2023).

2.2.3 *Convolutional Neural Network* (CNN)

CNN adalah salah satu algoritma dalam DL yang menggunakan arsitektur jaringan saraf tiruan (Rahmayuna, 2022). CNN merupakan hasil dari

pengembangan *multi-layer perceptron* (MLP). Perbedaannya terletak pada representasi *neuron*. CNN menampilkan *neuron* dalam bentuk 2 dimensi, sedangkan MLP merepresentasikan *neuron* hanya dalam 1 dimensi (Kusumaningrum dkk., 2020). Karena direpresentasikan dalam bentuk 2 dimensi, CNN cocok digunakan untuk pemrosesan dan pengenalan pola berbasis citra.

Dalam CNN, data diolah melalui proses *convolution* sebelum memasuki tahap klasifikasi menggunakan lapisan *fully-connected*. Proses ini terdiri dari dua tahap utama, yaitu *feature learning* dan *classification* seperti yang ditunjukkan pada Gambar 2. 1



Gambar 2. 1 Proses Pengolahan Citra dengan CNN (Husen, 2024)

Proses dimulai dengan *input* citra yang kemudian melalui beberapa lapisan *convolution* dan *pooling* untuk ekstraksi fitur. Lapisan pertama adalah lapisan *convolution* yang diikuti oleh fungsi aktivasi ReLU (*Rectified Linear Unit*), kemudian lapisan *pooling* untuk mengurangi dimensi data. Proses ini diulang beberapa kali untuk mendapatkan fitur yang lebih kompleks. Setelah itu, data di-*flatten* dan diteruskan ke lapisan *fully connected* untuk melakukan klasifikasi. Terakhir, fungsi *softmax* digunakan untuk menentukan probabilitas dari berbagai kelas.

Arsitektur pada CNN memiliki beberapa komponen (Rahmayuna, 2022), diantaranya:

1. Lapisan *Convolution*

Lapisan *convolution* berfungsi untuk mengurangi ukuran gambar dengan menggunakan *filter*. *Filter* ini membagi *input* citra menjadi bagian-bagian kecil

(misalnya 3x3 atau 5x5), lalu menggabungkannya menjadi *output* berupa *feature map*.

2. Fungsi Aktivasi

Fungsi aktivasi membantu jaringan saraf dalam membuat keputusan dan mempelajari fitur yang kompleks dari *input* citra. Fungsi aktivasi yang sering digunakan adalah *Rectified Linear Unit* (ReLU) yang berperan penting untuk menghindari kesalahan dalam membaca gambar karena intensitas gambar berkisar antara 0 hingga 255.

3. *Pooling*

Pooling adalah lapisan yang bertujuan untuk mengurangi sensitivitas terhadap *output feature map*. Dalam proses ini, setiap nilai di dalam kotak digunakan untuk membuat gambar dengan ukuran lebih kecil. Fungsinya adalah memperoleh nilai yang paling penting dari *input* citra. Dua metode yang biasa digunakan adalah *max pooling* dan *average pooling*. *Max pooling* bekerja dengan mengambil nilai tertinggi dalam kotak tertentu, sedangkan *average pooling* menghitung rata-rata dari semua nilai dalam kotak tersebut.

4. *Flattening*

Flattening merupakan proses mengubah data dari *array* 2 dimensi menjadi vektor 1 dimensi. *Flattening* bertujuan agar fitur-fitur yang telah dikumpulkan dapat masuk ke tahap *fully connected*.

5. Lapisan *Fully Connected*

Lapisan ini merupakan struktur dasar dari jaringan saraf yang menghubungkan semua *neuron* menjadi satu dimensi. Lapisan ini berfungsi dengan mengubah matriks menjadi bentuk *flatten* atau satu dimensi. Hasil yang diperoleh berupa data satu dimensi ini kemudian digunakan sebagai *input* awal yang akan dilatih dengan metode klasifikasi untuk menemukan *output* klasifikasi (Rahmayuna, 2022).

6. Fungsi *Softmax*

Fungsi ini dilakukan pada lapisan *output* selama tahap *fully connected* untuk menghasilkan nilai antara 0 dan 1 Fungsi ini digunakan dalam kasus klasifikasi *multi-class* yang bertujuan untuk mengklasifikasikan citra yang telah

diproses sebelumnya dengan menentukan probabilitas untuk setiap kelas (Zilziana Muflihati Noor, 2023).

2.2.4 Augmentasi Citra

Augmentasi citra adalah teknik yang banyak digunakan untuk menghasilkan data *training* tambahan dan meningkatkan generalisasi dalam analisis citra medis. Augmentasi gambar digunakan untuk meningkatkan akurasi dan ketahanan model deteksi dan klasifikasi tumor otak yang dilatih pada citra MRI (Raghuwanshi dkk., 2024). Augmentasi gambar terdiri dari 2 jenis, diantaranya:

1. Augmentasi Citra Statis

Augmentasi citra statis adalah teknik yang diterapkan pada dataset sebelum dilakukan proses *training* dimulai sebanyak satu kali sehingga data hasil augmentasi dapat digunakan. Karakterisitk teknik ini adalah:

a. Teknik ini dilakukan hanya sekali saja sebelum proses *training* dimulai sehingga tidak memerlukan komputasi tambahan.
b. Setiap kali proses *training* berjalan, model akan melihat data augmentasi yang sama sehingga bermanfaat untuk kebutuhan eksperimen.
c. Ruang penyimpanan yang diperlukan lebih besar karena semua variasi data dihasilkan sekaligus.

Kelebihan teknik ini adalah proses komputasi yang cepat dan tidak memengaruhi waktu *training* karena augmentasi telah dilakukan sebelum proses tersebut berjalan. Kekurangan teknik ini adalah membutuhkan ruang penyimpanan yang lebih besar serta variasi data terbatas pada augmentasi gambar sebelumnya.

2. Augmentasi Citra Dinamis

Augmentasi citra dinamis merupakan teknik yang dilakukan secara *real-time* selama proses *training*. Teknik ini menciptakan variasi data baru secara acak serta langsung diterapkan pada setiap *batch* data sebelum disajikan ke model. Ini berarti setiap *epoch* atau iterasi pelatihan dapat memiliki variasi data yang berbeda sehingga memperbanyak dataset secara dinamis. Karakteristik teknik ini adalah:

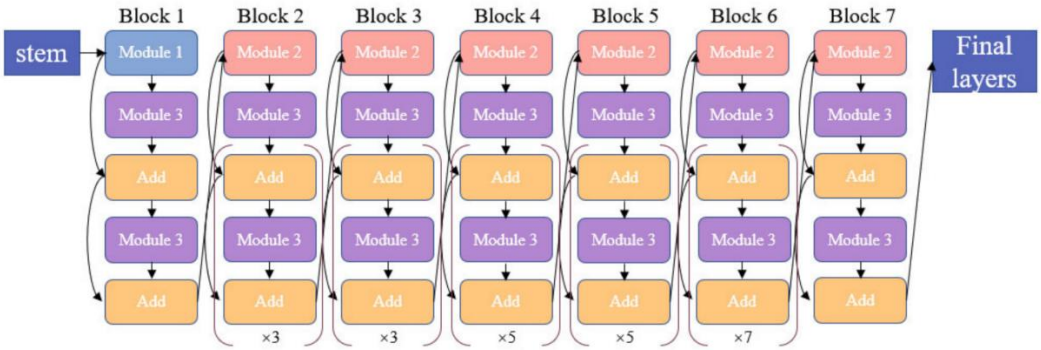
a. Dilakukan secara acak untuk setiap *batch data* sehingga memungkinkan variasi data yang tak terbatas selama proses *training*.

b. Tidak memerlukan ruang penyimpanan tambahan karena data hasil augmentasi tidak disimpan secara permanen.
c. Memerlukan komputasi tambahan selama proses *training* untuk melakukan augmentasi data pada tiap *batch*.

Kelebihan teknik ini adalah menghasilkan variasi data yang lebih luas dan lebih beragam selama proses *training*. Selain itu, teknik ini juga membantu model untuk lebih *robust* dan mampu menggeneralisasi data baru dengan lebih baik. Kekurangan teknik ini adalah memerlukan komputasi tambahan karena proses augmentasi dilakukan secara *real-time* sehingga terjadi kemungkinan memperlambat proses *training*.

2.2.5 Arsitektur Model EfficientNetB5

EfficientNetB5 merupakan salah satu varian dari arsitektur EfficientNet, yang bertujuan untuk meningkatkan efisiensi dan performa dalam tugas klasifikasi gambar. Model ini lebih efisien dan akurat daripada model sebelumnya dengan kemampuan mengenali gambar berukuran lebih besar, serta menggunakan pendekatan *compound scaling* untuk meningkatkan resolusi, kedalaman, dan lebar jaringan secara proporsional sehingga menghasilkan arsitektur yang lebih baik untuk melakukan klasifikasi gambar (Tan & Le, 2019). Struktur dari arsitektur ini ditunjukkan pada Gambar 2. 2.



Gambar 2. 2 Struktur Arsitektur Model EfficientNetB5 (Ge dkk., 2022)

Struktur pada model ini terdiri dari beberapa blok utama (blok 1 hingga blok 7) yang masing-masing berisi modul-modul tertentu, diikuti oleh lapisan awal (*stem*) dan lapisan akhir (*final layers*).

1. Lapisan pertama, yaitu lapisan *stem* menerima input berupa gambar dengan resolusi 456×456×3. Pada lapisan ini dilakukan proses konvolusi awal untuk mengekstraksi fitur dasar dari gambar.
2. Selanjutnya, lapisan blok-blok utama terdiri dari modul-modul utama yang dirancang menggunakan blok *Mobile Inverted Bottleneck Convolution* (MBConv) dengan kombinasi fungsi *depthwise convolution*, *pointwise convolution*, dan modul *squeeze-and-excitation*. Modul 1 dan 2 berisi lapisan konvolusi dasar dengan transformasi khusus untuk mengekstraksi fitur, dan Modul 3 berisi fungsi residual dengan mekanisme *skip connection*, yang memungkinkan informasi dari lapisan sebelumnya diteruskan secara langsung untuk menjaga stabilitas gradien selama pelatihan. Fungsi-fungsi ini dapat ditulis dalam Persamaan 2. 1, Persamaan 2. 2, Persamaan 2. 3, dan Persamaan 2. 4.

$$y = ReLU\big(BN(Conv_d(x)\big) \tag{2.1}$$

y adalah *depthwise convolution,* x adalah *input*, $Conv_d$ adalah *depthwise convolution*, dan BN adalah *batch normalization*.

$$z = \sigma\Big(BN(Conv_p(y)\Big) \tag{2.2}$$

z adalah *pointwise convolution,* $Conv_p$ adalah *pointwise convolution*, dan σ adalah fungsi aktivasi.

$$s = \sigma\big(W_2 \,.\, ReLU(W_1 \,.\, z_{global})\big) \tag{2.3}$$

s adalah *squeeze-and-excitation,* W_1 dan W_2 adalah matriks bobot dari lapisan *fully connected*, dan z_{global} adalah hasil agregasi global dari z.

$$y_{output} = x + s \,.\, z \tag{2.4}$$

y_{output} adalah *output* dari lapisan *residual connection*, x adalah *pointwise convolution*, s adalah *squeeze-and-excitation* atau *scaling factor,* dan z adalah *pointwise convolution* atau *output* transformasi fitur.

3. Tiap blok memiliki jumlah replikasi yang berbeda. Pada blok 2 dan 3 memiliki replikasi modul sebanyak 3 kali, blok 4 memiliki replikasi modul sebanyak 5 kali, serta blok 5 dan 6 memiliki replikasi modul sebanyak 7 kali. Jumlah

modul dalam setiap blok mengikuti prinsip *compound scaling* dengan kedalaman jaringan (jumlah modul) ditingkatkan secara sistematis untuk varian model yang lebih besar. *Compound scaling* dapat ditulis dengan Persamaan 2. 5

$$d = \alpha^{\phi}, w = \beta^{\phi}, r = \gamma^{\phi} \tag{2.5}$$

dengan syarat:

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2, \quad \alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

d adalah kedalaman *scaling factor* (*depth*), w adalah lebar *scaling factor* (*width*), r adalah resolusi *scaling factor* (*resolution*), ϕ adalah koefisien *compound*, dan α, β, γ adalah konstanta yang ditentukan untuk menyesuaikan skala.

4. Diantara modul, terdapat lapisan *add* yang berfungsi sebagai *residual connection* yang membantu meningkatkan efisiensi pelatihan dengan memastikan gradien tetap stabil.
5. Setelah melalui blok-blok utama, fitur yang telah diproses dikompresi menggunakan lapisan *global average pooling*, kemudian diteruskan ke lapisan terakhir (*fully connected layer*) yang bertugas melakukan klasifikasi berdasarkan *output* dari model.

Masing-masing blok terdiri dari beberapa modul, diantaranya adalah:

1. Modul 1

   Modul ini hanya digunakan pada blok 1 saja dan memiliki komposisi sebagai berikut:

   a. *Expansion Layer*

      Layer ini menggunakan operasi 1×1 *convolution*, yaitu kernel berukuran 1×1 piksel persegi untuk memperluas jumlah *channel*. Jika *input* memiliki sebanyak C *channel*, maka *output* yang dihasilkan dapat ditulis dengan Persamaan 2. 6:

$$output\ channel = C \times expansion\ ratio \tag{2.6}$$

Output channel adalah jumlah *channel* yang dihasilkan oleh *expansion layer*, *C* adalah jumlah *channel* dari *input*, dan *expansion ratio* adalah faktor pengali untuk memperluas jumlah *channel* sebelum proses *convolution*.

b. *Depthwise Convolution*

Layer ini menggunakan kernel berukuran 3×3 piksel persegi yang memproses setiap *channel* secara terpisah tanpa mencampurkan informasi antar *channel*. Pada konvolusi standar, jumlah parameter bergantung pada jumlah *channel input* dikalikan dengan jumlah *output* dan ukuran kernel, sedangkan pada *layer* ini hanya bergantung pada jumlah *channel input* dan ukuran kernel. Namun, karena informasi antar *channel* belum digabungkan, *layer* ini biasanya diikuti oleh *pointwise convolution* (konvolusi 1×1) yang bertugas menggabungkan informasi antar *channel* sehingga menghasilkan representasi fitur yang lebih kaya.

c. *Squeeze-and-Excitation* (SE) *Block*

Layer ini berfungsi untuk memberi bobot pada *channel* yang berperan penting dan yang tidak berperan penting. *Layer* ini memiliki beberapa komponen berupa:

1) *Squeeze*

Komponen ini bertujuan untuk merangkum informasi tiap *channel* yang direpresentasikan dalam bentuk vektor satu angka yang merupakan rata-rata seluruh piksel dari sebuah citra.

2) *Excitation*

Komponen ini bertujuan untuk mempelajari berapa pentingnya tiap *channel* yang akan digunakan. Pada komponen ini, vektor tersebut melewati 2 *layer fully connected* secara berurutan, dengan *layer* pertama akan mengurangi dimensi dengan rasio pengurangan tertentu, seperti 1/4 atau 1/16 dari jumlah *channel*. Sedangkan *layer* kedua akan mengembalikan dimensi tersebut ke ukuran semula. Terakhir, fungsi aktivasi berupa *sigmoid* digunakan untuk menghasilkan nilai bobot

antara 0 dan 1 yang mencerminkan tingkat kepentingan tiap *channel* terhadap proses klasifikasi yang sedang dipelajari.

3) *Reweight*

Komponen ini akan mengalikan bobot yang telah didapat tadi dengan *input feature map* asli secara elemen-per-elemen (*element wise multiplication*) sehingga *channel* yang dianggap penting akan diperkuat, sedangkan *channel* yang kurang relevan akan ditekan kontribusinya.

d. *Projection Layer*

Layer ini menggunakan operasi 1×1 *convolution* yang berfungsi untuk mengembalikan jumlah *channel* ke ukuran semula, sehingga dimensi *output* kembali sesuai dengan *input* awal sebelum proses ekspansi (*expansion layer*). *Layer* ini bekerja dengan mengurangi dimensi dari *tensor* berukuran $C \times expansion\ ratio$ *channel* menjadi C *channel*.

e. *Skip Connection* dan *Drop Connect*

Teknik *skip connection* akan menghubungkan *input* langsung ke *output* dari sebuah blok dalam jaringan tanpa melalui semua lapisan didalamnya. Cara kerjanya adalah dengan menambahkan hasil *output* dari blok (setelah melalui beberapa *layer*) dengan *input* awal secara langsung untuk membantu model agar informasi penting dari *input* tidak hilang saat diproses oleh banyak *layer*, selain itu juga bertujuan agar proses *training* lebih stabil dan cepat. Dalam implementasinya di arsitektur ini, *layer* ini hanya digunakan jika ukuran (jumlah *channel* dan dimensi spasial) *input* dan *output* sama, sehingga hasil penjumlahannya tetap valid.

Selain *skip connection*, terdapat juga teknik *drop connect* yang merupakan variasi dari *dropout*. Jika *Dropout* bekerja dengan menghapus beberapa neuron secara acak saat *training*, maka *drop connect* justru menghapus koneksi antar neuron (bobot) secara acak. Dalam implementasinya di arsitektur ini, *drop connect* diterapkan pada *skip connection*, yang mana tidak semua informasi dari *input* akan langsung

dijumlahkan ke *output*. Beberapa koneksi akan "dimatikan" secara acak selama *training* untuk mencegah model terlalu bergantung pada satu jalur informasi saja sehingga model menjadi lebih fleksibel dan tidak mudah mengalami *overfitting*. Ketika model digunakan untuk pengujian, semua koneksi diaktifkan kembali seperti biasa.

2. Modul 2

Merupakan variasi dari MBConv yang lebih kompleks dari Modul 1, digunakan pada blok 2 hingga blok 7 dengan jumlah kernel dan kedalaman (*depth*) yang bervariasi. Komposisi pada modul ini hampir sama dengan Modul 1 dengan beberapa perbedaan seperti pada Tabel 2. 1, diantaranya:

Tabel 2. 1 Perbedaan Modul 1 dengan Modul 2

Fitur	Modul 1	Modul 2
Expansion Layer	1 atau tidak ada ekspansi	Lebih dari 1, biasanya sebesar 6
Depthwise Convolution	Hanya menggunakan ukuran kernel 3x3	Menggunakan ukuran kernel 3x3 atau 5x5
SE (*Squeeze-and-Excitation*)	Tidak ada	Ada
Skip Connection dan *Drop Connect*	Biasanya tidak ada	Ada, jika dimensi tiap *channel* cocok
Jenis Konvolusi	Konvolusi biasa atau MBConv tanpa ekspansi	MBConv dengan ekspansi
Fungsi Utama	Ekstraksi fitur awal	Ekstraksi fitur lanjutan

Dengan kombinasi struktur MBConv yang kompleks, kernel yang bervariasi, ekspansi *channel* yang besar, serta integrasi SE *module*, blok 2 hingga blok 7 dirancang untuk menangani berbagai tingkat kompleksitas fitur visual dalam citra.

3. Modul 3

Modul ini merupakan modul tambahan yang sering digunakan setelah blok MBConv (blok 1 dan blok 2) dan sebelum penjumlahan residual (Modul *Add*). Biasanya terdiri dari satu atau lebih lapisan 1×1 *convolution* yang bertujuan untuk melakukan transformasi non-linier tambahan pada *output* MBConv tanpa menambah beban komputasi secara signifikan. Penggunaan konvolusi 1×1 memungkinkan model untuk menggabungkan informasi antar

channel, menyesuaikan dimensi *channel*, dan menyaring informasi yang paling relevan sebelum hasil akhir ditambahkan kembali ke *input* melalui *skip connection*.

4. Modul *Add*

Modul ini merujuk pada teknik *skip connection*, yaitu teknik yang menghubungkan *input* awal suatu blok langsung ke *output* akhirnya melalui proses penjumlahan. Proses ini dilakukan jika dimensi (baik ukuran spasial maupun jumlah *channel*) dari *input* dan *output* sama. Jika syarat terpenuhi, maka *input* asli akan langsung dijumlahkan (*element-wise addition*) dengan *output* dari blok konvolusional (MBConv) sebelum diteruskan ke blok berikutnya. Jika dimensi tidak sama, biasanya penyesuaian dilakukan terlebih dahulu, misalnya dengan 1×1 *convolution* pada *input* agar ukurannya sesuai dengan *output*. Penjumlahan ini membuat jaringan mempelajari "residu" atau selisih antara *input* dan *output*, bukan memetakan ulang semua fitur dari awal yang terbukti lebih mudah dan efisien dalam proses *training* jaringan dalam.

2.2.6 *Hyperparameter Tuning*

Hyperparameter tuning adalah proses sistematis untuk menentukan kombinasi nilai *hyperparameter* yang menghasilkan performa terbaik pada model pembelajaran mesin, khususnya pada arsitektur *deep learning* seperti CNN. *Hyperparameter* adalah parameter yang ditetapkan sebelum proses pelatihan dimulai dan tidak dipelajari oleh model, berbeda dengan bobot dan bias yang dioptimalkan selama *training*. Nilai *hyperparameter* yang tepat sangat berpengaruh pada stabilitas *training*, kecepatan konvergensi, serta kemampuan generalisasi model (Raiaan dkk., 2024).

Dalam konteks CNN dan model berbasis *transfer learning*, beberapa *hyperparameter* utama yang menentukan kualitas pembelajaran antara lain:

1. Learning Rate

Learning rate menentukan seberapa besar perubahan bobot model yang dilakukan pada setiap iterasi. Jika nilai *learning rate* terlalu tinggi, model dapat melewatkan titik minimum yang optimal, sedangkan jika terlalu rendah, model bisa mengalami pelatihan yang sangat lambat.

2. *Dropout Rate*

Dropout mencegah model terlalu sempurna dengan menggeser beberapa neuron secara acak selama pelatihan. *Dropout rate* yang terlalu rendah dapat menyebabkan model terlalu sesuai dengan data *training* dan mengalami *overfitting*, sementara jika *dropout rate* yang terlalu tinggi, model dapat kehilangan informasi penting untuk pembelajaran.

3. *Batch Size*

Batch size mengacu pada jumlah sampel yang digunakan dalam setiap iterasi pembaruan bobot. Ukuran *batch* yang terlalu besar dapat memperlambat pelatihan, sedangkan ukuran yang terlalu kecil dapat menyebabkan fluktuasi besar dalam pembaruan bobot dan memperpanjang waktu pelatihan.

Ada beberapa pendekatan untuk melakukan *hyperparameter tuning*, yang dibahas di bawah ini:

1. *Grid Search*

Grid Search merupakan metode *hyperparameter tuning* yang bekerja dengan menguji seluruh kombinasi parameter dalam ruang pencarian yang telah ditentukan. Pendekatan ini memastikan bahwa setiap kombinasi diuji secara sistematis sehingga peluang menemukan konfigurasi terbaik sangat tinggi. Metode ini banyak digunakan pada ruang parameter yang relatif kecil dan model yang tidak terlalu kompleks karena cakupan pencariannya bersifat menyeluruh. Namun, kelemahan utama *Grid Search* adalah ketika ruang parameter membesar, jumlah kombinasi meningkat secara eksponensial, sehingga waktu komputasi dan sumber daya yang dibutuhkan menjadi sangat tinggi. Selain itu, *Grid Search* cenderung menghabiskan banyak waktu pada titik-titik yang kurang relevan dalam ruang parameter dibandingkan metode yang lebih adaptif (Bergstra & Bengio, 2012).

2. *Random Search*

Random Search menawarkan alternatif yang lebih efisien dibandingkan *Grid Search* dengan mengevaluasi kombinasi *hyperparameter* secara acak dari ruang pencarian yang sama. Dengan menghindari eksplorasi seluruh kombinasi, *Random Search* dapat menemukan konfigurasi yang baik dalam waktu yang jauh lebih singkat, terutama ketika hanya sebagian kecil dimensi parameter yang

memiliki pengaruh signifikan pada performa model. *Random Search* juga mampu menghasilkan hasil yang kompetitif bahkan dengan jumlah percobaan yang jauh lebih sedikit (Bergstra & Bengio, 2012). Namun, metode ini tidak memberikan jaminan bahwa kombinasi paling optimal akan ditemukan, karena pemilihan titik dilakukan secara acak sehingga kualitas solusi bergantung pada jumlah iterasi.

3. *Bayesian Optimization*

Bayesian Optimization merupakan pendekatan yang lebih canggih karena memanfaatkan model probabilistik, misalnya *Gaussian Process* untuk memperkirakan fungsi objektif dan memilih kombinasi *hyperparameter* yang memiliki peluang terbesar menghasilkan performa terbaik. Dengan memanfaatkan *surrogate model* dan *acquisition function*, metode ini mampu menyeimbangkan eksplorasi dan eksploitasi ruang parameter secara adaptif. Keunggulannya terletak pada efisiensi pencarian, terutama pada kasus *deep learning* yang memiliki biaya komputasi tinggi per iterasi, sehingga metode ini mampu menemukan konfigurasi optimal dengan jumlah evaluasi yang jauh lebih sedikit. *Bayesian Optimization* sangat efektif dalam optimasi fungsi yang mahal dan multidimensi. Meskipun demikian, implementasinya lebih kompleks dalam hal perancangan model probabilistik, dan memerlukan pemahaman statistik yang lebih mendalam dibandingkan *Grid Search* atau *Random Search* (Turner dkk., 2021).

4. *Particle Swarm Optimization* (PSO)

PSO adalah algoritma yang menggunakan prinsip populasi, yang diambil dari cara burung atau ikan berkumpul dan bergerak bersama. Dalam mekanismenya, setiap solusi direpresentasikan sebagai partikel yang bergerak dalam ruang parameter berdasarkan pengalaman terbaiknya (*personal best*) dan pengalaman terbaik dari seluruh kelompok (*global best*). Gerakan adaptif ini membuat PSO mampu menangkap pola pencarian yang lebih kaya dan sering kali lebih cepat menemukan solusi yang kompetitif pada masalah nonlinear dan multidimensi. PSO juga terkenal mampu menjaga keseimbangan eksplorasi–eksploitasi secara dinamis. PSO efektif dalam berbagai tugas optimasi, termasuk *tuning* dalam model *deep learning*. Meski demikian, PSO memiliki sensitivitas tinggi terhadap parameter seperti ukuran populasi dan kecepatan partikel. Bila tidak diatur dengan

tepat, partikel dapat mengalami konvergensi prematur dan terjebak pada solusi lokal (Qolomany dkk., 2020).

2.2.7 *Confusion Matrix* untuk Klasifikasi *Multi-Class*

Klasifikasi *multi-class* adalah metode dalam DL yang bertujuan untuk mengklasifikasikan data ke dalam satu dari beberapa kelas yang berbeda, yaitu tiga atau lebih. Berbeda dengan klasifikasi biner yang hanya membedakan antara dua kelas, klasifikasi *multi-class* menangani masalah yang lebih kompleks dengan berbagai aplikasi, seperti pengenalan citra, analisis sentimen, dan pengenalan aktivitas manusia. Terdapat 2 metode yang digunakan untuk klasifikasi ini, diantaranya (Chakraborty & Dey, 2024):

1. *One-vs-Rest* (OvR)

Metode ini melatih model untuk membedakan satu kelas dengan semua kelas lainnya secara terpisah sehingga perlu membangun beberapa model klasifikasi biner untuk kumpulan data dari beberapa kelas. Jumlah label kelas dalam kumpulan data dan jumlah pengklasifikasi biner yang dihasilkan harus sama sehingga memerlukan penguraian himpunan data *multi-class* menjadi masalah klasifikasi biner diskret. Kelebihan metode ini adalah kemudahan implementasi dan efisiensi saat jumlah kelas tidak terlalu besar. Namun, metode ini juga memiliki kekurangan, yaitu potensi ketidakseimbangan data antara kelas utama dan kelas lainnya yang dapat mempengaruhi performa model.

2. *One-vs-One* (OvO)

Metode ini melatih model untuk membedakan setiap kombinasi dua kelas. Metode ini memiliki keunggulan dalam hal kemampuan untuk mengurangi kompleksitas dalam model, karena setiap model hanya perlu membedakan dua kelas pada satu waktu. Namun, metode ini juga memiliki kekurangan, yaitu jumlah model yang dibutuhkan bertambah sangat cepat seiring bertambahnya jumlah kelas.

Confusion matrix digunakan untuk menghitung kinerja dari model klasifikasi. Nilai yang ditampilkan di setiap baris menunjukkan jumlah data yang telah dikelompokkan ke dalam beberapa kelas. *Confusion matrix* juga menilai apakah model telah memahami pola-pola penting dari data pelatihan serta menentukan sejauh mana model menggeneralisasi data baru yang belum pernah

dilihat (Lesmana dkk., 2024). Secara umum, validasi klasifikasi menggunakan beberapa parameter seperti pada Tabel 2. 2, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN).

Tabel 2. 2 *Confusion Matrix Multi-Class*

		Prediksi			
		Kelas 1	Kelas 2	Kelas 3	Kelas 4
Aktual	Kelas 1	TP1	FN1	FN2	FN3
	Kelas 2	FP1	TP2	FN4	FN5
	Kelas 3	FP2	FP3	TP3	FN6
	Kelas 4	FP4	FP5	FP6	TP4

Pada klasifikasi *multi-class*, perhitungan TP, FP, FN, dan TN dilakukan dengan pendekatan *one-vs-rest* untuk setiap kelas ke-i, yaitu dengan memperlakukan satu kelas sebagai kelas positif dan seluruh kelas lainnya sebagai kelas negatif. TP menunjukkan data sebenarnya yang bernilai positif dan berhasil diklasifikasikan ke dalam kelas positif. FP menunjukkan data sebenarnya yang bernilai negatif, tetapi diklasifikasikan ke dalam kelas positif. FN menunjukkan data sebenarnya yang bernilai negatif, tetapi diklasifikasikan ke dalam kelas negatif. TN menunjukkan data sebenarnya yang bernilai negatif dan berhasil diklasifikasikan ke dalam kelas negatif. Untuk klasifikasi *multi-class*, TN tidak selalu ditonjolkan karena mencakup jumlah data dari semua kelas lain di luar kelas yang sedang dianalisis. Untuk setiap kelas ke-i, nilai TN dapat dihitung menggunakan Persamaan 2. 7.

$$TN_i = N - (TP_i + FP_i + FN_i) \tag{2.7}$$

N adalah total jumlah data, TP_i adalah *True Positive* pada kelas ke-i, FP_i adalah *False Positive* pada kelas ke-i, dan FN_i adalah *False Negative* pada kelas ke-i.

Setelah mendapatkan nilai tersebut, perhitungan kinerja model dilakukan dengan menghitung nilai akurasi, presisi, *recall*, dan *f1-score*. Pada penelitian ini, metrik presisi, *recall*, dan *f1-score* dihitung menggunakan pendekatan *macro-average*, sehingga setiap kelas memiliki bobot yang sama dalam evaluasi. Misalkan terdapat k kelas, maka metrik evaluasi dirumuskan sebagai berikut.

Akurasi menunjukkan perbandingan antara jumlah kelas yang diklasifikasikan secara benar dan jumlah total prediksi sehingga tingginya nilai akurasi mencerminkan seberapa akurat model dalam melakukan klasifikasi (Celik & Inik, 2024). Untuk menghitung akurasi, digunakan Persamaan 2. 8.

$$akurasi = \frac{\sum_{i=1}^{k} TP_i}{N} \tag{2.8}$$

Presisi menunjukkan perbandingan antara jumlah data yang diprediksi sebagai kelas positif dan benar, terhadap seluruh data yang diprediksi sebagai kelas positif. Hasil dari presisi menunjukkan seberapa akurat model dalam memprediksi kelas secara tepat (Celik & Inik, 2024). Untuk menghitung presisi dari tiap kelas ke-i, digunakan rumus Persamaan 2. 9.

$$presisi_i = \frac{TP_i}{TP_i + FP_i} \tag{2.9}$$

Untuk memperoleh nilai presisi secara keseluruhan pada klasifikasi *multi-class*, digunakan pendekatan *macro-average* yang menghitung nilai presisi sebagai rata-rata dari presisi sebanyak k kelas, sehingga seluruh kelas diperlakukan secara setara tanpa mempertimbangkan perbedaan jumlah sampel pada masing-masing kelas. Untuk menghitungnya, dapat menggunakan Persamaan 2. 10.

$$presisi_{macro} = \frac{\sum_{i=1}^{k} presisi_i}{k} \tag{2.10}$$

Recall atau *sensitivity* adalah perbandingan antara jumlah data yang diprediksi benar sebagai positif dengan semua data yang sebenarnya positif. Hasil dari *recall* menggambarkan seberapa sering model secara benar memprediksi kelas positif. Untuk menghitung *recall* tiap kelas ke-i, digunakan rumus seperti pada Persamaan 2. 11.

$$recall_i = \frac{TP_i}{TP_i + FN_i} \tag{2.11}$$

Nilai *recall* secara keseluruhan pada klasifikasi *multi-class* didapat dengan menggunakan pendekatan *macro-average* yang menghitung nilai *recall* sebagai rata-rata dari *recall* sebanyak k kelas. Nilai ini digunakan untuk mengevaluasi kemampuan model dalam mendeteksi seluruh kelas secara merata sehingga sensitivitas model terhadap kelas minoritas dan mayoritas memiliki kontribusi yang sama dalam evaluasi akhir. Nilai tersebut dapat dihitung menggunakan Persamaan 2. 12.

$$recall_{macro} = \frac{\sum_{i=1}^{k} recall_i}{k} \tag{2.12}$$

F1-score atau *f-measure* digunakan untuk mengukur akurasi pengujian. Nilai ini mencerminkan seberapa seimbang antara nilai presisi dengan nilai *recall*. Tujuannya agar dapat melihat seberapa baik model dalam mengklasifikasikan hasil prediksi positif dan negatif secara tepat (Husen, 2024). Untuk menghitung *f1-score* setiap kelas ke-i, digunakan Persamaan 2. 13.

$$f1_i = 2 \times \frac{presisi_i \; x \; recall_i}{presisi_i + recall_i} \tag{2.13}$$

Nilai *f1-score* secara keseluruhan didapat melalui *macro-average* yang digunakan untuk menilai keseimbangan antara presisi dan *recall* pada klasifikasi *multi-class*. Nilai ini dihitung dengan merata-ratakan *f1-score* dari setiap kelas sebanyak k, sehingga mampu merepresentasikan performa model secara komprehensif tanpa bias terhadap kelas dengan jumlah data yang lebih besar. Untuk menghitung nilai tersebut, digunakan Persamaan 2. 14.

$$f1_{macro} = \frac{\sum_{i=1}^{k} f1_i}{k} \tag{2.14}$$

Untuk memperjelas proses perhitungan metrik evaluasi pada klasifikasi *multi-class*, berikut disajikan contoh perhitungan berdasarkan *confusion matrix*. Contoh ini bertujuan untuk menunjukkan cara menentukan nilai TP, FP, FN, dan TN serta menurunkan metrik evaluasi yang digunakan dalam penelitian ini.

Tabel 2. 3 Contoh *Confusion Matrix Multi-Class*

		Predicted			
		glioma tumor	*meningioma tumor*	*pituitary tumor*	*no tumor*
Actual	*glioma tumor*	30	4	1	0
	meningioma tumor	3	30	0	0
	pituitary tumor	0	1	29	1
	no tumor	0	0	1	30

Tabel 2. 3 menunjukkan contoh *confusion matrix* hasil pengujian model klasifikasi tumor otak dengan empat kelas, yaitu *glioma tumor*, *meningioma tumor*, *pituitary tumor*, dan *no tumor*. Baris merepresentasikan kelas aktual, sedangkan kolom menunjukkan kelas hasil prediksi model. Nilai diagonal utama menunjukkan jumlah prediksi yang benar pada masing-masing kelas, sementara nilai di luar diagonal menunjukkan kesalahan klasifikasi (*misclassification*). Pada bagian ini ditampilkan contoh perhitungan untuk satu kelas, yaitu kelas *glioma tumor*. Pemilihan kelas ini didasarkan pada pertimbangan bahwa prosedur perhitungan nilai TP, FP, FN, dan TN pada klasifikasi *multi-class* bersifat identik untuk setiap kelas, sehingga contoh pada satu kelas telah cukup merepresentasikan proses perhitungan secara keseluruhan. Dengan demikian, perhitungan untuk kelas lainnya dapat diperoleh dengan menerapkan prinsip yang sama berdasarkan nilai pada *confusion matrix*.

Berdasarkan Tabel 2. 3, terlihat bahwa:

1. *N* atau jumlah data yang digunakan adalah sebanyak 130 data.
2. TP (*True Positive*): Data kelas *glioma tumor* yang diprediksi sebagai kelas *glioma tumor*, sebanyak 30 data.
3. FP (*False Positive*): Data selain kelas *glioma tumor* yang diprediksi sebagai kelas *glioma tumor*, yaitu sebanyak $3 + 0 + 0 = 3$ data yang berasal dari kelas *meningioma tumor*.

4. FN (*False Negative*): Data kelas *glioma tumor* yang diprediksi sebagai kelas lain, yaitu sebanyak $4 + 1 + 0 = 5$ data yang masing-masing diprediksi sebagai kelas *meningioma tumor* dan *pituitary tumor*.
5. TN (*True Negative*): Seluruh data selain kelas *glioma tumor* yang diprediksi bukan kelas *glioma tumor*. Dengan menggunakan Persamaan 2. 7, nilai TN yang diperoleh adalah $130 - (30 + 3 + 5) = 92$ data.

Setelah nilai TP, FP, FN, dan TN didapat, perhitungan metrik untuk kelas *glioma tumor* dapat dilakukan sebagai berikut:

$$presisi_{glioma} = \frac{30}{30 + 3} = 0{,}9090 \rightarrow 90{,}90\%$$

$$recall_{glioma} = \frac{30}{30 + 5} = 0{,}8571 \rightarrow 85{,}71\%$$

$$f1_{glioma} = \frac{2 \times 0{,}9090 \times 0{,}8571}{0{,}9090 + 0{,}8571} = 0{,}8823 \rightarrow 88{,}23\%$$

Perhitungan nilai TP, FP, FN, dan TN dilakukan untuk setiap kelas secara independen dengan prinsip yang sama seperti contoh pada kelas *glioma tumor*. Perlu dicatat bahwa akurasi pada klasifikasi *multi-class* merupakan metrik evaluasi global yang dihitung berdasarkan keseluruhan prediksi model terhadap seluruh kelas, sehingga tidak diturunkan pada contoh perhitungan tiap kelas dan dibahas secara khusus pada Bab 4 berdasarkan hasil evaluasi model.