Deep Learning

JUDUL

Deep Learning Approaches for Brain Tumor Detection: A Comparison between Custom CNN and YOLO Pretrained Models

Oleh:

1. Jason Evansaputra Sudargo – 2702243722
2. Moeh Adji Anggalaksana – 2702404323

BINUS University 2025/2026

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Otak manusia adalah organ yang sangat penting yang bertanggung jawab atas berbagai proses tubuh, seperti pernapasan, memori, emosi, penglihatan, keterampilan motorik, dan respons. Fungsi otak dapat terganggu ketika tumor mulai tumbuh di sana. Dua jenis tumor di otak adalah primer, yang tumbuh langsung di otak, dan *metastatik*, yang berasal dari organ lain tetapi menyebar ke otak.

Diagnosis tumor otak lebih menantang dibandingkan organ lain karena adanya *Blood-Brain Barrier* yang menghalangi deteksi dengan penanda radioaktif. Oleh karena itu, pencitraan MRI digunakan secara luas sebagai metode efektif untuk mendeteksi keberadaan tumor otak.

Secara epidemiologi, terdapat sekitar 7–11 kasus tumor otak per 100.000 orang setiap tahun, dengan 227.000 kematian, serta sekitar 7,7 juta penyintas yang hidup dengan disabilitas [1]. Diagnosis dini sangat penting karena dapat mengurangi angka kematian, menekan risiko disabilitas, dan memungkinkan tindakan medis yang lebih minimal invasif.

Namun, ahli radiologi memiliki beberapa keterbatasan saat mereka menginterpretasikan gambar MRI secara manual. Proses ini sangat bergantung pada pengalaman dan keahlian ahli, memakan waktu, dan rentan terhadap kelelahan dan kesalahan manusia, terutama karena jumlah besar data gambar yang dimiliki setiap pasien. Karena keterbatasan ini, sistem diagnosis berbasis komputer (CAD) diciptakan untuk meningkatkan efisiensi, akurasi, dan objektivitas dalam proses diagnosis.

Sebagai bagian dari kecerdasan buatan, *Deep Learning* telah menunjukkan peningkatan yang signifikan dalam tugas analisis citra medis dalam beberapa tahun terakhir. *Convolutional Neural Network* (CNN) telah menjadi standar emas secara khusus karena kemampuan untuk mempelajari fitur hierarkis seperti tekstur, bentuk, dan batas tumor secara otomatis dari data citra tanpa perlu melakukan rekayasa fitur manual.

Dalam penerapan CNN untuk deteksi tumor otak, terdapat dua pendekatan utama. Pendekatan pertama adalah membangun arsitektur dari awal (custom CNN). Arsitektur kustom ini menawarkan fleksibilitas karena dapat disesuaikan secara spesifik dengan dataset yang digunakan dan konseptualnya lebih sederhana. Namun, performanya mungkin terbatas dan seringkali membutuhkan data dalam jumlah sangat besar serta sumber daya komputasi yang intensif untuk mencapai hasil yang optimal.

Pendekatan kedua adalah memanfaatkan model yang telah dilatih sebelumnya (*pre-trained models*) pada dataset gambar yang masif, contohnya seperti arsitektur YOLO (You Only Look

Once). Model-model ini dikenal memiliki arsitektur canggih yang mampu mencapai akurasi dan kecepatan deteksi tinggi. Akan tetapi, pendekatan ini juga memiliki tantangan. Karena model ini umumnya dilatih pada dataset gambar umum (bukan gambar medis), fitur-fitur yang telah dipelajarinya mungkin tidak sepenuhnya optimal untuk mengenali karakteristik yang unik dan terkadang subtil dari citra MRI. Selain itu, arsitekturnya yang sudah baku menawarkan fleksibilitas yang lebih rendah untuk dimodifikasi jika dibandingkan dengan membangun model dari awal.

Mengingat adanya kelebihan dan kekurangan pada masing-masing pendekatan, muncul urgensi untuk membandingkan kedua metode tersebut secara langsung. Perbandingan ini penting untuk mengetahui secara jelas *trade-off* antara performa dan kompleksitas yang ditawarkan. Dengan demikian, dapat diidentifikasi pendekatan mana yang paling efektif dan efisien untuk digunakan dalam pengembangan sistem deteksi tumor otak otomatis.

1.2 Rumusan Masalah

Penelitian ini berfokus pada evaluasi dan perbandingan efektivitas dua pendekatan *deep learning* yang berbeda arsitektur *Convolutional Neural Network* (CNN) kustom dan model *pre-trained* YOLO untuk tugas deteksi tumor otak dari citra MRI. Adanya perbedaan fundamental dalam cara kerja dan karakteristik kedua model ini menimbulkan pertanyaan mengenai *trade-off* antara performa, kecepatan, dan fleksibilitas. Berdasarkan latar belakang tersebut, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana arsitektur *Convolutional Neural Network* (CNN) kustom dapat dirancang dan dibangun untuk melakukan klasifikasi dan deteksi tumor otak pada citra MRI secara efektif?
2. Bagaimana performa model *pre-trained* YOLO ketika diterapkan secara langsung untuk tugas deteksi tumor otak dari dataset citra MRI yang sama?
3. Bagaimana perbandingan kinerja antara model CNN kustom dengan model *pre-trained* YOLO jika diukur menggunakan metrik evaluasi standar seperti akurasi, presisi, *recall*, dan *F1-score*?
4. Apa saja kelebihan dan keterbatasan utama dari masing-masing pendekatan dalam konteks studi kasus deteksi tumor otak ini?

1.3 Tujuan Penelitian

Selaras dengan rumusan masalah yang telah dijabarkan, penelitian ini memiliki beberapa tujuan utama sebagai berikut:

1. Merancang, membangun, dan melatih model *Convolutional Neural Network* (CNN) kustom yang mampu mendeteksi keberadaan tumor otak dari citra MRI
2. Mengimplementasikan dan menerapkan model *pre-trained* YOLO untuk melakukan tugas deteksi tumor otak pada dataset citra MRI yang sama

3. Menganalisis dan membandingkan secara kuantitatif performa dari model CNN kustom dan model *pre-trained* YOLO berdasarkan metrik evaluasi meliputi akurasi, presisi, *recall*, dan *F1-score*
4. Memberikan analisis kualitatif mengenai *trade-off* antara kompleksitas arsitektur dengan performa yang dihasilkan oleh kedua pendekatan, serta mengidentifikasi kelebihan dan keterbatasan masing-masing model dalam studi kasus ini

1.4 Manfaat Penelitian

Hasil dari penelitian ini diharapkan dapat memberikan sejumlah manfaat yang signifikan, baik dari sisi akademis, praktis, maupun sosial.

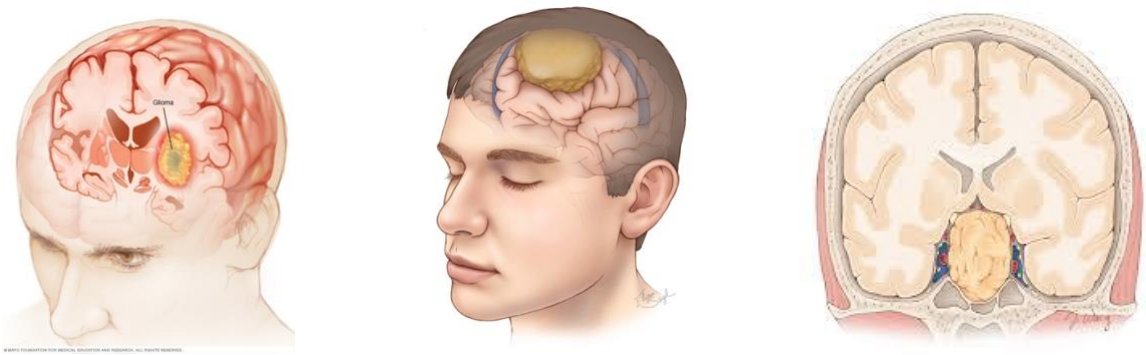
1. Manfaat Akademis
   Secara akademis, penelitian ini berkontribusi pada literatur ilmiah, khususnya dalam studi perbandingan metode *deep learning* untuk analisis citra medis. Hasil perbandingan antara arsitektur CNN kustom dan model *pre-trained* YOLO dapat menjadi referensi empiris mengenai kekuatan dan kelemahan masing-masing pendekatan dalam domain spesifik deteksi tumor otak.
2. Manfaat Praktis
   Secara praktis, temuan dari penelitian ini dapat menjadi panduan bagi para praktisi, pengembang, atau peneliti di bidang kecerdasan buatan dan kesehatan. Analisis mengenai *trade-off* antara performa dan kompleksitas model dapat membantu mereka dalam mengambil keputusan untuk memilih arsitektur yang paling sesuai dengan kebutuhan, sumber daya komputasi yang tersedia, dan tujuan spesifik dari sistem yang akan dikembangkan.
3. Manfaat Sosial
   Secara sosial, penelitian ini mendukung upaya pengembangan teknologi untuk deteksi dini tumor otak. Meskipun tidak langsung berdampak pada pasien, kemajuan dalam sistem deteksi otomatis berpotensi membantu para profesional medis di masa depan, sehingga dapat meningkatkan efisiensi dan objektivitas dalam proses diagnosis. Pada akhirnya, hal ini diharapkan dapat berkontribusi pada peningkatan kualitas layanan kesehatan dan peluang kesembuhan pasien.

BAB 2
TINJAUAN PUSTAKA

2.1 Tumor Otak

Tumor otak merupakan salah satu penyakit dimana terdapat pertumbuhan sel abnormal dan tidak terkendali di sekitar otak. Pertumbuhan tumor otak biasanya ditandai oleh pertumbuhan massa padat tidak terkendali yang dibentuk oleh sel-sel [2]. Penyebabnya bisa berasal dari sel otak itu sendiri (primer) atau dari sel organ tubuh lain yang menjalar ke otak (metastatis). Beberapa jenis tumor otak yang umum ditemukan yaitu glioma, meningioma, dan pituitary.

Glioma merupakan jenis tumor otak yang tumbuh dari sel glial, yaitu sel penunjang yang berfungsi menjaga dan melindungi neuron di otak. Meningioma merupakan tumor otak yang berasal dari *meninges* dan sumsum tulang belakang. Sementara pituitary adalah jenis tumor otak yang tumbuh di kelenjar pituitary dimana kelenjar ini bertugas untuk mengontrol fungsi tubuh melalui produksi dan pelepasan hormon ke aliran darah [3]. Gambar 1 merupakan ilustrasi dari ketiga jenis tumor otak.



Gambar 1a: Glioma *Gambar 1b: Meningioma* *Gambar 1c: Pituitary*

Gambar 1: Ilustrasi Tumor Otak

2.2 Citra MRI (Magnetic Resource Imaging)

Magnetic Resonance Imaging adalah teknik pencitraan medis yang menggunakan medan magnet kuat untuk menghasilkan gambar detail dari struktur internal tubuh. Dalam proses pemeriksaan otak, MRI terbukti sangat efektif untuk mendeteksi berbagai kelainan seperti tumor, pendarahan, infeksi, dan lainnya. Beberapa alat lainnya dapat digunakan untuk mendeteksi kelainan di otak, akan tetapi, MRI adalah metode yang paling populer karena terbukti sangat aman [3].

Citra yang dihasilkan oleh MRI memiliki resolusi yang tinggi dan dalam bentuk format *grayscale* dengan beberapa jenis pengambilan gambar seperti T1-*weighted* atau T2-*weighted*.

Resolusi yang tinggi memungkinkan perbedaan kontras antar jaringan otak dapat terlihat dengan jelas [3]. Namun, diperlukan ketelitian yang tinggi dan pengalaman lapangan yang cukup oleh dokter spesialis radiologi agar didapat hasil interpretasi citra MRI yang tepat. Hal ini dikarenakan otak memiliki struktur yang kompleks menjadikan kesulitan tersendiri dalam identifikasi tumor otak [2].

Penelitian Mbuyamba tahun 2017 menunjukkan bahwa meningkatnya jumlah pasien mempengaruhi beban kerja dokter sehingga menyebabkan diagnosa manual tidak direkomendasikan [2]. Apabila interpretasi dilakukan secara manual, faktor subjektif dapat mempengaruhi hasil diagnosa. Oleh karena itu, diperlukan alat atau mesin yang mampu menginterpretasikan penyakit tumor otak ini secara presisi.

2.3 Deep Learning untuk Analisis Citra

Pada awalnya, tugas segmentasi dan klasifikasi tumor otak dilakukan menggunakan algoritma *Machine Learning* yang klasik. Pendekatan ini sangat bergantung pada metode *feature engineering* yang dirancang secara manual. Namun, pendekatan ini mulai tergantikan dengan pendekatan yang lebih baru yaitu *Deep Learning*. Pendekatan *Deep Learning* mampu mempelajari fitur-fitur yang ada secara otomatis dan langsung dari data. Beberapa penelitian menunjukkan bahwa pendekatan berbasis *Deep Learning* memiliki performa dan akurasi yang jauh lebih tinggi dibandingkan dengan pendekatan berbasis *Machine Learning* terutama dalam data dengan skala yang cukup besar [4].

Proses analisis citra medis untuk deteksi tumor otak secara manual merupakan tugas yang memakan waktu bagi para dokter. Hal ini disebabkan oleh volume dataset citra MRI yang sangat besar serta tingginya kemiripan antara jaringan tumor dengan jaringan normal, sehingga berpotensi menunda penanganan pasien [5]. Menjawab tantangan tersebut, kini, *Deep Learning* sudah menjadi *state-of-the-art* dalam pengenalan gambar khususnya citra medis. *Deep Learning* juga digunakan secara luas dalam analisis citra otak seperti klasifikasi penyakit atau tugas segmentasi lainnya [4].

Secara teknis, model *Deep Learning* (DL) bekerja dengan memanfaatkan jaringan saraf tiruan untuk mengekstraksi pola kompleks langsung dari data dalam sebuah proses yang dikenal sebagai pelatihan (*training*). Pada tahap ini, model secara berulang-ulang memperbaiki performanya dengan mengatur parameter internal untuk meminimalkan selisih antara prediksi dan label (*error*). Kemampuannya untuk menganalisis data dalam volume besar memungkinkan model ini menangkap fitur-fitur tersembunyi yang sering kali luput dari observasi manusia [6].

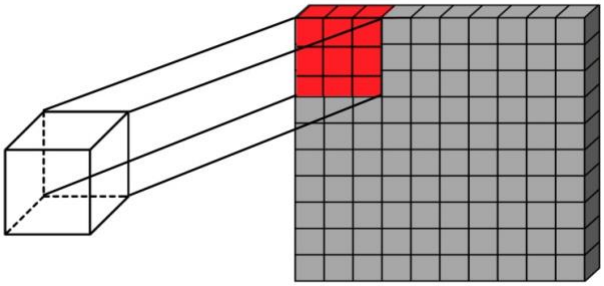
2.4 Convolutional Neural Network

Mayoritas arsitektur yang dipakai dalam *Deep Learning* untuk pemrosesan gambar adalah *Convolutional Neural Network* (CNN). CNN merupakan teknologi revolusioner yang telah meraih perhatian yang signifikan pada beberapa tahun terakhir [3]. Komponen inti dari CNN adalah *convolutional layer*, yang bekerja dengan menggeser sebuah *filter* yang dipelajari untuk melintasi gambar menggunakan mekanisme *"sliding-window"*. Pendekatan ini menggunakan parameter yang lebih sedikit dibandingkan dengan jaringan saraf *fully-connected* biasa dan mampu mengenali pola yang dipelajari tanpa terpengaruh oleh lokasinya di dalam citra. Setiap *convolutional layer* dalam jaringan memungkinkan adanya peningkatan abstraksi untuk mengidentifikasi fitur-fitur yang semakin kompleks [6].

Berbeda dengan jaringan saraf buatan yang biasa (*Multi Layer Perceptron*), data yang diproses di dalam CNN adalah data dalam bentuk dua dimensi sehingga operasinya berbeda. Operasi linear dalam CNN menggunakan operasi konvolusi dimana parameter bobot tidak lagi berbentuk satu dimensi. Karena sifat proses konvolusi, CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi seperti citra dan suara.

2.4.1 Convolutional Layer

Sebagai komponen fundamental dalam arsitektur CNN, *Convolutional Layer* bertanggung jawab untuk proses ekstraksi fitur dari data input. Lapisan ini terdiri dari sejumlah *filter* yang melakukan operasi konvolusi dengan cara bergeser di atas citra input. Secara matematis, operasi konvolusi adalah perkalian antara bobot *filter* dengan bagian-bagian dari citra, yang hasilnya kemudian dijumlahkan untuk membentuk satu nilai pada sebuah *feature map* [7]. Setiap *feature map* merepresentasikan deteksi fitur spesifik, seperti tepi, sudut, atau tekstur pada citra.



Gambar 2: Convolutional Layer

Secara matematis, operasi konvolusi itu didefinisikan sebagai

$$F(i,j) = (G * H)(i,j) = \sum_m \sum_n G(m,n)H(i-m, j-n) \quad (1)$$

dimana $F(i,j)$ mewakili output *feature map* di posisi (i,j), $G(m,n)$ adalah gambar inputnya, dan $H(i-m,j-n)$ mewakili filter yang di-*apply* ke input gambar. Pada persamaan ini, indeks m dan n merujuk pada dimensi spasial dari *filter*, yang beriterasi sepanjang lebar dan tingginya. Secara praktis, operasi konvolusi secara efektif menggeser *filter* melintasi gambar input, lalu menghitung *dot product* antara bobot *filter* dengan nilai piksel yang bersesuaian [8]. Proses ini bertujuan untuk mengekstraksi fitur, sehingga hasilnya bisa diproses di tahap selanjutnya.

2.4.2 Activation Function

Fungsi aktivasi merupakan salah satu komponen krusial dalam arsitektur jaringan saraf tiruan. Fungsi ini diterapkan pada keluaran setiap neuron untuk mentransformasi hasil menjadi output baru yang akan diteruskan ke lapisan berikutnya. Tujuannya untuk menentukan output, akurasi, dan efisiensi model pelatihan [7]. Terdapat beragam jenis fungsi aktivasi, seperti *ReLU* (Rectified Linear Unit), *Sigmoid*, *Tanh*, dan *SoftMax*. Dalam arsitektur *Deep Learning* modern, *ReLU* sering menjadi pilihan utama karena keunggulannya dalam efisiensi komputasi dan kemampuannya mengatasi masalah *vanishing gradient*. *ReLU* memiliki rumus

$$\mathrm{f_{relu}} = \max(0, \mathrm{x}) \tag{2}$$

dimana x adalah output dari neuron, yang dimasukkan ke fungsi aktivasi sehingga nilai berapapun yang di bawah nol akan di-nol kan oleh *ReLU* [4].

2.4.3 Pooling Layer

Setelah *feature map* dihasilkan oleh *Convolutional Layer*, data tersebut kemudian diteruskan ke *Pooling Layer*. Lapisan ini bertujuan untuk mereduksi dimensi spasial dari *feature map* [8]. Proses ini mengurangi kompleksitas komputasi dan jumlah parameter dengan tetap mempertahankan fitur-fitur yang paling dominan atau penting. Terdapat dua jenis operasi *pooling* yang umum digunakan, yaitu *Max Pooling* dan *Average Pooling* [4]. Dalam penelitian ini, metode yang digunakan adalah *Max Pooling*.

2.4.4 Fully Connected Layer

Lapisan *Fully Connected* adalah struktur lapisan dimana semua neuron saling terhubung. Biasanya lapisan ini diimplementasikan di akhir jaringan [7]. Lapisan ini berfungsi untuk mengintegrasikan fitur-fitur yang telah dipelajari oleh lapisan-lapisan sebelumnya untuk menghasilkan prediksi atau klasifikasi akhir. Koneksi ini mampu mempelajari hubungan-hubungan kompleks antar fitur. Output dari lapisan FC dihitung menggunakan persamaan sebagai

$$y = \sigma(Wx + b) \tag{3}$$

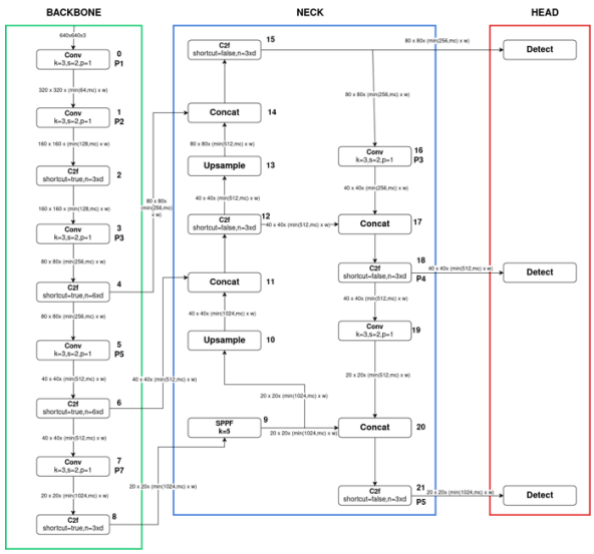
dimana y adalah hasil keluaran, x adalah vektor input, W adalah matriks bobot, b adalah vektor bias, dan σ adalah fungsi aktivasi .

2.5 You Only Look Once (YOLO)

YOLO (*You Only Look Once*) merupakan salah satu model *deep learning state-of-the-art* yang sangat populer, terutama untuk tugas deteksi objek (*object detection*). Model ini telah melalui proses pra-pelatihan (*pre-training*) pada dataset berskala besar, yang memberikannya kemampuan ekstraksi fitur yang andal dan akurasi yang tinggi. Dalam penelitian ini, digunakan YOLO versi 8 (YOLOv8), yang dikenal unggul karena kombinasi antara kecepatan pemrosesan, akurasi, dan kemudahan implementasinya [9]. YOLOv8 menyediakan kerangka kerja yang komprehensif untuk berbagai tugas komputer seperti deteksi, segmentasi, dan klasifikasi.

Arsitektur utama YOLOv8 dibagi menjadi tiga bagian utama, yaitu *backbone*, *neck*, dan *head*. Bagian *backbone* adalah bagian pertama yang bertanggung jawab untuk ekstraksi fitur dari input data di mana biasanya menggunakan CNN *pretrained* seperti *ImageNet* [10]. Terdapat *C2f module* yang berguna mempertahankan *receptive field* sambil mengurangi tekanan pada memori *bandwidth*. Fitur multi-skala digabungkan dengan mekanisme *attention* untuk mempertahankan informasi spasial yang tajam sehingga sangat penting untuk objek berukuran kecil [11].

Bagian *neck* berfungsi untuk memproses dan menyaring data yang diterima dari bagian *backbone*. Ia menggunakan beberapa teknik untuk meningkatkan representasi fitur [10]. Sebelum masuk ke bagian ini, modul SPPF (*Spatial Pyramid Poolling Fast*), tepat setelah *layer* terakhir dari bagian *backbone,* memberikan representasi dari *multi-scale feature map*. SPPF memungkinkan model untuk menangkap fitur pada berbagai tingkat abstraksi melalui proses *pooling* pada skala yang berbeda-beda. Dalam bagian *neck*, terdapat beberapa blok *concat* dan *upsample* yang berguna untuk meningkatkan resolusi dari *feature map* [12]. Bagian terakhir yaitu bagian *head* di mana bagian ini bertugas untuk melakukan deteksi terhadap fitur-fitur yang sudah diproses sebelumnya di bagian *neck* [10]. Gambar 3 adalah ilustrasi dari arsitektur *YOLOv8*.



Gambar 3: Arsitektur YOLOv8

2.6 Penelitian Terkait

Pemanfaatan *deep learning* telah menjadi fokus utama dalam analisis citra medis. Deteksi tumor otak dari citra MRI adalah salah satu bidang yang paling banyak dieksplorasi, mengingat proses interpretasi manual yang dilakukan oleh ahli radiologi membutuhkan ketelitian yang tinggi.

Studi-studi terdahulu telah membuktikan bahwa arsitektur CNN sangat efektif. Sebagai contoh, sebuah penelitian yang menerapkan arsitektur *ResNet152V2* untuk klasifikasi empat kelas tumor berhasil mencapai akurasi sebesar 94.44% [13]. Studi lain juga membuktikan bahwa arsitektur *VGG16* untuk tugas yang sama berhasil mencapai akurasi sebesar 96.01% [14]. Selain dari itu, model *pretrained* seperti YOLO juga memiliki potensi besar dalam analisis citra medis. Beberapa studi menggunakan YOLOv8 untuk deteksi dan mencapai akurasi lebih dari 90%.

Jika membandingkan hasil yang ada, baik CNN maupun YOLO menunjukkan performa yang baik. Keduanya bisa mencapai akurasi diatas 90%. Meskipun demikian, terdapat kesenjangan penelitian yang signifikan. Banyak studi berfokus pada satu arsitektur saja atau perbandingan dua arsitektur *pretrained* CNN. Perbandingan langsung model kustom CNN dengan *pretrained* model seperti YOLO untuk tugas klasifikasi tumor otak masih terbatas. Adanya perbandingan seperti ini juga penting untuk memahami secara objektif *trade-off* antara kedua pendekatan tersebut. Model CNN kustom, meskipun berpotensi dirancang secara spesifik untuk dataset medis, seringkali memerlukan data dalam jumlah besar untuk menghindari *overfitting* dan membutuhkan sumber daya komputasi yang signifikan untuk melatih dari awal. Di sisi lain, model *pretrained* seperti YOLO memungkinkan pelatihan yang lebih cepat dan berpotensi memberikan generalisasi yang lebih baik berkat pengetahuan yang didapat dari dataset berskala besar. Oleh karena itu, penelitian ini akan mengisi kesenjangan tersebut dengan melakukan perbandingan untuk memberikan panduan yang jelas dalam memilih arsitektur yang paling sesuai untuk tugas klasifikasi tumor otak.

BAB 3
METODOLOGI

3.1 Dataset

Dataset yang digunakan dalam penelitian ini merupakan *dataset* publik yang bersumber dari repositori Kaggle, yang disusun oleh Masoud Nickparvar [15]. *Dataset* tersebut terdiri dari 7023 citra MRI otak manusia yang terbagi ke dalam empat kelas berbeda, yaitu tiga jenis tumor (*glioma*, *meningioma*, *pituitary*) dan satu kelas non-tumor (tanpa tumor/sehat). Setiap citra disajikan dalam format *jpg* dengan ukuran citra yang berbeda-beda. *Dataset* ini sudah terbagi menjadi dua bagian yaitu data pelatihan dan data *testing*. Distribusi citra untuk setiap kelas disajikan pada *Tabel 1*.

Kelas	Training	Testing
Glioma	1321	300
Meningioma	1339	306
Pituitary	1457	300
No Tumor	1595	405

Tabel 1: Distribusi data

3.2 Pra-pemrosesan Data

Pra-pemrosesan data merupakan tahap paling penting atau fundamental dalam pengembangan model *deep learning* yang bertujuan untuk mengubah data mentah menjadi data yang bersih, seragam, dan optimal untuk proses pelatihan. Tahapan ini merupakan tahapan yang penting karena kualitas data yang input secara langsung mempengaruhi hasil dari prediksi model. Tujuan dari tahapan ini adalah untuk mencegah *overfitting*, memperkaya variasi data latih, serta menyeragamkan format data.

Dataset citra MRI otak manusia tersebut memiliki ukuran atau dimensi yang berbeda-beda. Sementara itu model CNN atau YOLO memerlukan ukuran dataset yang seragam. Oleh karena itu, semua citra dalam dataset diubah menjadi ukuran yang sama yaitu 256x256 piksel. Langkah ini memastikan bahwa ukuran input selalu konsisten supaya lebih mudah untuk diprediksi oleh model.

Setiap piksel dalam satu gambar memiliki rentang nilai 0 sampai 255. Nilai ini cukup besar yang dapat memperlambat proses pembentukan model yang optimal dan minimnya *error* yang dihasilkan. Normalisasi adalah proses untuk mengubah skala rentang nilai piksel tersebut menjadi rentang yang lebih kecil yaitu antara 0 dan 1. Proses ini bisa dilakukan dengan membagi setiap piksel dengan 255. Dengan minimnya rentang nilai piksel, hal ini bisa mempercepat model menuju bentuk yang optimal.

Keterbatasan jumlah data juga menjadi tantangan yang cukup penting dalam membentuk model yang optimal. Jika data memiliki pola yang terlalu mudah, model akan cenderung

menghafal pola sehingga bisa menyebabkan *overfitting*. Untuk mengatasi hal ini, diterapkan teknik augmentasi data. Teknik ini bertujuan untuk memperbanyak variasi dari dataset yang sudah ada. Augmentasi dilakukan dengan menerapkan beberapa transformasi geometri secara acak pada citra MRI. Transformasi yang diterapkan yaitu:

1. *Random Flipping*: Membalik citra secara acak baik vertikal maupun horizontal
2. *Random Rotation*: Memutar citra secara acak dari berbagai sudut
3. *Random Zoom*: Memperbesar atau memperkecil citra secara acak

Model klasifikasi juga memerlukan data label dalam format numerik supaya dapat diproses. Oleh karena itu label-label dalam dataset perlu diubah menjadi representasi numerik. Proses ini sering disebut dengan istilah *Label Encoding*. Perubahan format ini memungkinkan model untuk menghitung *loss function* seperti *Sparse Categorical Crossentropy* supaya bisa mencapai hasil yang maksimal.

3.3 Arsitektur Model

Pada tahap ini, arsitektur model *deep learning* dirancang untuk tugas klasifikasi citra. Penelitian ini menggunakan dua pendekatan model yang berbeda untuk dibandingkan, yaitu arsitektur *Convolutional Neural Network* yang dibangun dari awal dan model YOLOv8 yang memanfaatkan arsitektur *pre-trained*.

3.3.1 Arsitektur *Custom* CNN

Model pertama yang digunakan yaitu membangun model CNN sendiri. Model ini dirancang secara khusus untuk klasifikasi citra dalam penelitian ini. Arsitektur yang dibangun menggunakan *framework Tensorflow* dengan *API Keras*. Arsitektur CNN ini terdiri dari beberapa lapisan utama yang bekerja secara *sequential* yaitu:

1. *Input Layer*: Lapisan ini menerima data citra dengan dimensi 256x256 piksel.
2. *Data Augmentation Layer*: Lapisan augmentasi data yang telah dijelaskan pada sub-bab sebelumnya diintegrasikan langsung ke dalam model. Ketika model dalam mode *inferensi* (prediksi tanpa *update* parameter), lapisan ini tidak digunakan.
3. *Normalization Layer*: Lapisan normalisasi yang juga telah dijelaskan pada sub-bab sebelumnya diintegrasikan langsung ke dalam model. Sama dengan lapisan augmentasi data, lapisan ini tidak digunakan ketika model dalam mode *inferensi*.
4. *Convolutional Layers*: Terdapat tiga lapisan konvolusi yang berfungsi untuk mengekstrak fitur pada citra.
    - Conv2D Lapisan 1: Menggunakan 32 filter dengan ukuran *kernel* 3x3
    - Conv2D Lapisan 2: Menggunakan 64 filter dengan ukuran *kernel* 3x3
    - Conv2D Lapisan 3: Menggunakan 64 filter dengan ukuran *kernel* 3x3

    Setiap lapisan konvolusi menggunakan fungsi aktivasi *ReLU* (Rectified Linear Unit) untuk menghindari *vanishing-gradient problem*, serta inisialisasi bobot menggunakan *he_normal* dan regularisasi L2 untuk mengurangi *overfitting*.

5. *Pooling Layers*: Setelah setiap lapisan konvolusi, diterapkan lapisan *MaxPooling2D* dengan ukuran 2x2. Lapisan ini berfungsi untuk mengurangi dimensi spasial dari *feature map* tanpa menghilangkan fitur penting, sehingga komputasi menjadi lebih efisien.
6. *Flatten Layer*: Lapisan ini mengubah output dari lapisan konvolusi yang berbentuk matriks dua dimensi menjadi vektor satu dimensi supaya dapat diproses oleh lapisan selanjutnya.
7. *Fully-Connected (Dense) Layers*: Terdapat beberapa lapisan *Dense* yang berfungsi untuk melakukan klasifikasi berdasarkan fitur yang telah diekstraksi. Untuk mencegah *overfitting*, lapisan *Dropout* diterapkan dengan *rate* 20% yang disisipkan setelah lapisan *Dense* pertama.
8. *Output Layer*: Lapisan *Dense* terakhir merupakan lapisan output dengan 4 neuron (sesuai jumlah kelas yang akan diprediksi) dan menggunakan fungsi aktivasi *softmax*. Fungsi *softmax* akan menghasilkan probabilitas untuk setiap kelas di mana kelas dengan probabilitas tertinggi akan menjadi prediksi akhir model.

Untuk proses pelatihan, model di-*compile* dengan:

- *Optimizer: Adam* (sebuah algoritma optimisasi yang efisien secara komputasi)
- *Loss Function: Sparse Categorical Crossentropy* (fungsi loss yang umum digunakan untuk masalah klasifikasi multi kelas.
- Parameter Pelatihan: Pelatihan dilakukan dengan *learning rate* sebesar 0.001 selama 30 *epochs*.

3.3.2 Model YOLOv8

Dalam penelitian ini, model YOLOv8 yang sudah dilatih pada dataset besar seperti COCO (*Common Object in Context*) digunakan sebagai titik awal. Model ini kemudian dilatih secara langsung pada dataset citra medis yang telah disiapkan. Dengan kata lain, kami menerapkan *transfer learning* di mana pengetahuan dasar dari model YOLOv8 diadaptasi untuk secara spesifik mengenali kelas-kelas yang ada dalam penelitian ini. Parameter yang digunakan dalam model YOLOv8 ini yaitu ukuran *batch* sebesar 32 dan 20 *epochs*.

3.4 Setup Pelatihan

Seluruh proses komputasi, mulai dari pra-pemrosesan data hingga pelatihan model, dijalankan pada platform *cloud computing Google Colaboratory*. Lingkungan ini menyediakan akses ke sumber daya komputasi yang memiliki kinerja tinggi yang memungkinkan untuk melatih model *deep learning* secara efisien. Spesifikasi perangkat keras yang digunakan yaitu GPU T4 yang melakukan akselerasi proses operasi matriks dan *tensor* secara signifikan. Memori dialokasikan secara dinamis oleh *Google Colaboratory* untuk menangani proses yang ada dalam RAM dengan maksimal kapasitas mencapai 12GB.

Lingkungan pengembangan untuk penelitian ini dibangun di atas serangkaian *library* dan *framework* yang berbasis bahasa pemrograman *Python*. Rincian perangkat lunak yang digunakan sebagai berikut:

- Lingkungan Eksekusi: *Google Colaboratory Notebook*, sebuah lingkungan yang berbasis *Jupyter Notebook* yang interaktif dan terintegrasi dengan sumber daya *Google Cloud*.
- Bahasa Pemrograman: *Python* versi 3.12
- *Framework Deep Learning*
    - *Tensorflow* versi 2.19 dengan *API Keras* yang digunakan untuk membangun, melatih, dan mengevaluasi model kustom CNN
    - *YOLOv8* dari *Ultralytics* digunakan untuk implementasi model YOLO
- *Library* Pendukung: *NumPy, Matplotlib, Shutil*

Durasi pelatihan merupakan faktor penting dalam eksperimen. Dengan adanya akselerasi dari GPU, waktu yang dibutuhkan untuk melatih setiap model dapat dioptimalkan. Pelatihan model kustom CNN yang dilakukan selama 30 *epochs* membutuhkan waktu sekitar 2 hingga 3 menit. Sementara pelatihan model YOLOv8 yang dilakukan selama 20 *epochs* membutuhkan waktu sekitar 9 hingga 10 menit. Total waktu pelatihan disesuaikan dengan jumlah *epochs* yang telah ditentukan untuk mencapai model yang optimal.

3.5 Metrik Evaluasi

Metrik evaluasi yang objektif, diperlukan untuk mengukur kinerja model klasifikasi secara kuantitatif. Metrik-metrik ini dihitung berdasarkan hasil dari *confusion matrix*, yang membandingkan prediksi model dengan label yang sebenarnya. Dalam penelitian ini, empat metrik utama yang digunakan yaitu akurasi, presisi, *recall*, dan *F1-score*. Sebelum merinci untuk setiap metrik evaluasi, penting untuk memahami konsep dasar dari *confusion matrix*:

- *True Positive* (TP): Jumlah kasus di mana model memprediksi kelas positif dengan benar
- *True Negative* (TN): Jumlah kasus di mana model memprediksi kelas negatif dengan benar
- *False Positive* (FP): Jumlah kasus di mana model salah memprediksi kelas positif
- *False Negative* (FN): Jumlah kasus di mana model salah memprediksi kelas negatif

3.5.2 Akurasi (*Accuracy*)

Akurasi adalah metrik yang paling intuitif di mana ia mengukur rasio total prediksi yang benar terhadap keseluruhan jumlah data. Metrik ini memberikan gambaran tentang seberapa sering model membuat prediksi yang tepat. Namun, akurasi memiliki kekurangan di mana ia bisa menyesatkan jika dataset tidak seimbang. Oleh karena itu, diperlukan metrik lain agar lebih jelas.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

3.5.3 Presisi (*Precision*)

Presisi mengukur seberapa akurat prediksi positif yang dibuat oleh model. Metrik ini menunjukkan dari semua kasus yang diprediksi sebagai tumor, berapa persen yang sebenarnya benar-benar tumor. Presisi tinggi menunjukkan bahwa model memiliki tingkat *false positive* yang rendah. Metrik ini sangat penting terutama dalam bidang medis, di mana prediksi positif yang salah dapat menyebabkan kecemasan yang tidak perlu pada pasien.

$$Presisi = \frac{TP}{TP + FP} \tag{5}$$

3.5.4 Recall (*Sensitivity*)

Metrik ini mengukur kemampuan model untuk menemukan semua kasus positif yang sebenarnya. *Recall* menunjukkan dari semua kasus tumor yang ada di dataset, berapa persen yang berhasil diidentifikasi oleh model. Metrik ini sangat penting dalam dunia medis di mana tingkat *False Negative* yang tinggi sangat berbahaya karena berarti ada kasus penyakit yang terlewatkan oleh model yang dapat menunda penanganan medis.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

3.5.5 F1-Score

F1-Score adalah rata-rata dari Presisi dan *Recall*. Metrik ini memberikan keseimbangan antara kedua metrik tersebut, sehingga menjadi indikator yang baik untuk menilai kinerja model secara keseluruhan. *F1-Score* memiliki rentang nilai dari nol hingga satu di mana satu menunjukkan nilai presisi dan recall yang sempurna.

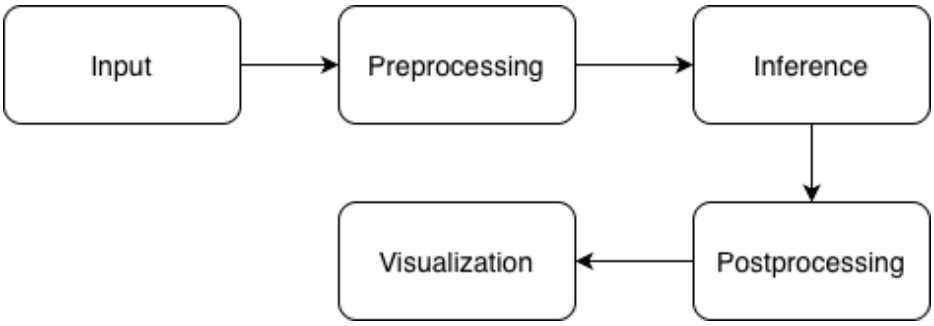
$$F1 - Score = 2 \times \frac{Presisi \times Recall}{Presisi + Recall} \tag{7}$$

BAB 4
IMPLEMENTASI DAN HASIL

4.1 Detail Sistem

Sistem yang dikembangkan pada penelitian ini diberi nama *TumorSense AI*, sebuah aplikasi berbasis web untuk deteksi dan klasifikasi tumor otak pada citra *Magnetic Resource Imaging* (MRI). Sistem ini dibangun dengan *framework Streamlit* dengan mengintegrasikan model antara CNN atau YOLO, dilihat dari performanya yang terbaik. Secara garis besar, arsitektur sistem dirancang untuk menerima *input* berupa *file* citra digital, memprosesnya melalui model *deep learning* yang sudah dilatih, dan menghasilkan *output* berupa prediksi kelas tumor. Mekanisme kerja sistem dari sisi pengguna hingga mendapatkan hasil diagnosa dapat diuraikan dalam tahapan sebagai berikut:

1. Input Data: Pengguna mengunggah citra MRI melalui antarmuka web
2. Prapemrosesan: Citra input secara otomatis diubah ukurannya menjadi 256x256 piksel dan dinormalisasi nilai pikselnya
3. Inferensi Model: Citra yang telah diproses dimasukkan ke dalam model yang telah memuat bobot hasil pelatihan. Model melakukan kalkulasi matriks untuk menentukan probabilitas kemiripan citra dengan keempat kelas target
4. Pasca pemrosesan: Sistem mengambil kelas dengan probabilitas tertinggi dan menampilkannya sebagai hasil prediksi
5. Visualisasi: Hasil yang didapat ditampilkan ke pengguna



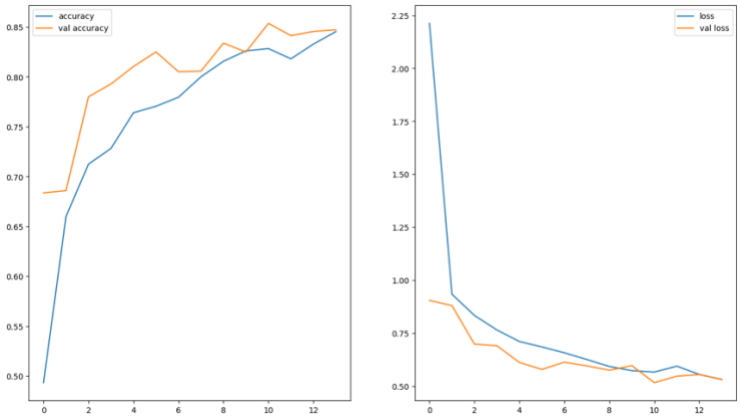
Gambar 4: Flowchart alur kerja sistem

4.2 Eksperimen

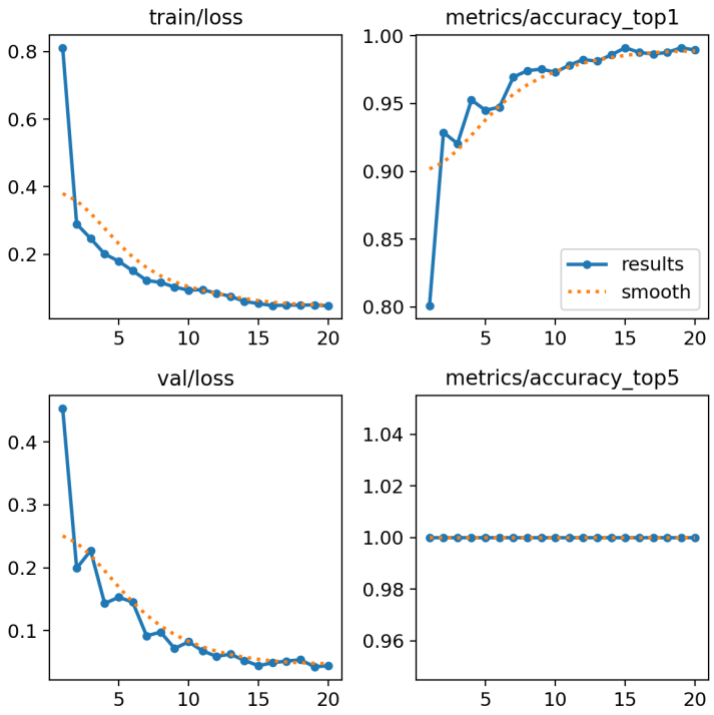
Pada tahap ini dilakukan serangkaian eksperimen untuk melatih dua model yang dibandingkan, yaitu *Custom* CNN dan YOLO *pretrained* model, sesuai metodologi yang telah dijelaskan pada Bab 3. Eksperimen dilakukan untuk memperoleh performa model secara nyata, serta mengevaluasi perbedaan hasil yang muncul dari masing-masing pendekatan. Proses pelatihan dilakukan pada

lingkungan komputasi yang sama untuk memastikan hasil yang adil. Seluruh eksperimen dijalankan pada platform *Google Colaboratory* yang mendukung GPU sehingga proses pelatihan dapat berjalan secara efisien. Proses pelatihan dilakukan menggunakan data yang telah melalui *preprocessing* dan *augmentation*. Selama pelatihan, dilakukan pemantauan terhadap *training loss* serta *validation accuracy* untuk melihat kestabilan model dan potensi *overfitting*. Proses pelatihan YOLO menggunakan *transfer learning*, di mana sebagian besar bobot model dipertahankan dari hasil *pretraining*, dan hanya *layer* tertentu yang disesuaikan dengan dataset MRI. Selama eksperimen, proses pelatihan *Custom* CNN selalu terkena *early stopping* (proses pelatihan diberhentikan lebih awal agar tidak terjadi *overfitting*). Sementara model YOLO, tetap berjalan sesuai dengan konfigurasi yang sudah ditentukan.

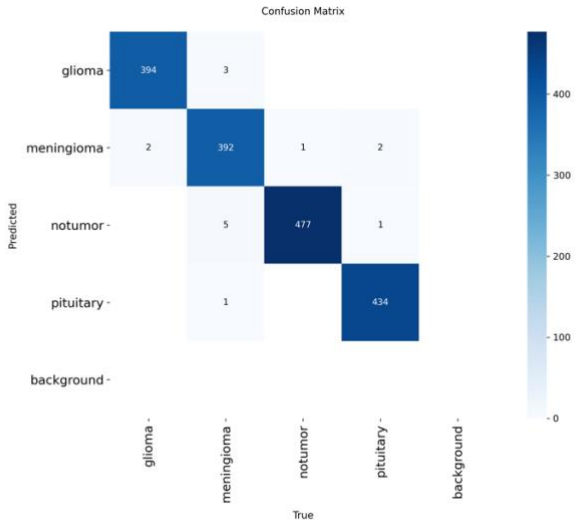
4.3 Visualisasi



Gambar 5: CNN Train History



Gambar 6: YOLO Train History



Gambar 7: Confusion Matrix YOLO

4.4 Hasil yang Diperoleh

Setelah proses testing, didapat bahwa YOLO merupakan model yang lebih baik jika dibandingkan dengan CNN *Custom*. Perbandingan hasil bisa dilihat pada Tabel 2.

Metrics	CNN	YOLO
Accuracy	0.805	0.983
Precision	0.798	0.983
Recall	0.792	0.981
F1-Score	0.791	0.982

Tabel 2: Tabel Perbandingan Hasil Model

BAB 5
PEMBAHASAN DAN KETERBATASAN

5.1 Analisis Performa

Hasil evaluasi menunjukkan bahwa model YOLO *pretrained* memberikan hasil yang konsisten lebih baik dibandingkan dengan CNN *Custom*. Berdasarkan pengukuran metrik yang sudah ditentukan sebelumnya, model YOLO lebih unggul pada keempat metrik tersebut. Hal ini menunjukkan bahwa model *pretrained* yang telah dilatih pada dataset besar mampu melakukan generalisasi lebih baik ketika diaplikasikan pada tugas deteksi tumor otak.

Di sisi lain, CNN *Custom* menunjukkan bahwa performanya masih kalah dengan model YOLO terutama pada metrik *recall*. Hal ini berarti bahwa CNN cenderung gagal pada saat mendeteksi sebagian besar kasus tumor otak. Salah satu penyebabnya dikarenakan keterbatasan arsitektur serta keterbatasan dataset sehingga model tidak mampu untuk menangkap pola tumor yang diberikan. Selain dari itu, CNN juga memiliki kelebihannya, yaitu biaya komputasi yang jauh lebih murah dibandingkan dengan YOLO. Waktu yang digunakan untuk proses pelatihan dan memori yang digunakan jauh lebih kecil dibandingkan dengan YOLO.

5.2 Tantangan yang Dihadapi

Selama proses eksperimen, ada beberapa tantangan yang muncul pada studi ini, diantaranya:

1. Variasi kualitas citra MRI
   Citra MRI memiliki perbedaan ukuran, tingkat *noise*, dan teknik pencitraan. Hal ini menyebabkan model sering kesulitan untuk menangkap pola secara konsisten.
2. Struktur dataset yang berbeda
   Model YOLO membutuhkan struktur dataset yang sesuai. Sehingga proses *splitting* tidak bisa dilakukan secara langsung sebelum pelatihan. Struktur dataset harus disesuaikan dengan yang dibutuhkan oleh model YOLO.

5.3 Keterbatasan

Penelitian ini memiliki beberapa keterbatasan yaitu:

1. Dataset terbatas
   Data citra MRI memiliki jumlah yang cenderung sedikit. Model CNN *Custom* yang dilatih dari nol, membutuhkan data yang banyak supaya model bisa lebih konvergen.
2. Arsitektur CNN sederhana
   Model CNN *Custom* pada penelitian ini cenderung memiliki arsitektur dengan kedalaman yang terbatas. Arsitektur yang lebih kompleks mungkin dapat memberikan hasil yang lebih baik.
3. Tidak menguji model pada data klinis nyata
   Seluruh evaluasi hanya menggunakan dataset publik. Dalam kondisi klinis, format dan kualitas gambar biasanya lebih beragam.
4. Evaluasi hanya mencakup metrik dasar
   Penelitian ini hanya menggunakan metrik dasar untuk evaluasi sehingga memerlukan metrik yang lebih spesifik lagi supaya lebih baik.

BAB 6
KESIMPULAN DAN SARAN

Penelitian ini bertujuan untuk membandingkan performa dua pendekatan *deep learning* yaitu model CNN *Custom* dan YOLO *pretrained* pada tugas mendeteksi tumor otak pada citra MRI. Berdasarkan eksperimen dan analisis hasil, beberapa kesimpulan utama diperoleh sebagai berikut:

1. YOLO menunjukkan performa yang jauh lebih tinggi dibandingkan CNN pada seluruh metrik evaluasi dengan nilai di atas 0.97. Hal ini menunjukkan bahwa kemampuan model yang sudah dilatih pada dataset umum, memberikan keuntungan yang signifikan dalam tugas deteksi tumor otak.
2. *Preprocessing* dan *data augmentation* memberikan dampak yang besar pada kedua model. Namun, augmentasi harus dilakukan secara hati-hati karena data medis tidak boleh mengubah struktur anatomi secara berlebihan.
3. Berdasarkan seluruh hasil, model YOLO merupakan pendekatan yang lebih sesuai untuk kasus deteksi tumor otak.

BAB 7
DAFTAR PUSTAKA

[1] S. Anantharajan, S. Gunasekaran, T. Subramanian, and V. R. Venkatesh, "MRI brain tumor detection using deep learning and machine learning approaches," *Measurement: Sensors*, vol. 31, p. 101026, 2024, doi: 10.1016/j.measen.2024.101026.

[2] M. N. Putri, K. Adi, M. I. Katili, S. Hariyanto, and D. Rochmayanti, "Comparison Measurement and Calculation of Brain Tumor in MRI Modalities Utilising Spin-Echo Pulse Sequence T1-Weighted Contrast and Digital Image Processing Applications," *Journal of Vocational Health Studies*, vol. 6, pp. 151–157, 2022, doi: 10.20473/jvhs.V6.I2.2022.151-157.

[3] Y. B. Nainggolan, R. S. Perdana, and A. A. Soebroto, "Klasifikasi Penyakit Tumor Otak berdasarkan Citra MRI Menggunakan Metode Convolutional Neural Network EfficientNetV2-S," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 9, no. 10, pp. x–x, Oct. 2025, [Online]. Available: http://j-ptiik.ub.ac.id

[4] S. A. Y. Al-Galal, I. F. T. Alshaikhli, and M. M. Abdulrazzaq, "MRI Brain Tumor Medical Images Analysis Using Deep Learning Techniques: A Systematic Review," *Health Technol (Berl)*, vol. 11, pp. 267–282, Jan. 2021, doi: 10.1007/s12553-020-00514-6.

[5] R. Zhang, H. Luo, W. Chen, and Y. Bai, "Review of deep learning-driven MRI brain tumor detection and segmentation methods," *Advances in Computer, Signals and Systems*, vol. 7, no. 8, pp. 17–28, 2023, doi: 10.23977/acss.2023.070803.

[6] F. J. Dorfner, J. B. Patel, J. Kalpathy-Cramer, E. R. Gerstner, and C. P. Bridge, "A review of deep learning for brain tumor analysis in MRI," *NPJ Precis Oncol*, vol. 9, no. 2, Jan. 2025, doi: 10.1038/s41698-024-00789-2.

[7] D. Gunawan and H. Setiawan, "Convolutional Neural Network dalam Analisis Citra Medis," *KONSTELASI: Konvergensi Teknologi dan Sistem Informasi*, vol. 2, no. 2, Dec. 2022, [Online]. Available: https://ojs.uajy.ac.id/index.php/konstelasi/article/download/5367/2568/16298

[8] I. D. Mienye, T. G. Swart, G. Obaido, M. Jordan, and P. Ilono, "Deep Convolutional Neural Networks in Medical Image Analysis: A Review," *Information*, vol. 16, no. 3, 2025, doi: 10.3390/info16030195.

[9] A. Alshahrani, J. Mustafa, M. Almatrafi, L. Albaqami, R. Aljabri, and S. Almuntashri, "A Comparative Study of Deep Learning Techniques for Alzheimer's disease Detection in Medical Radiography," *IJCSNS International Journal of Computer Science and Network Security*, vol. 24, no. 5, pp. 53–63, May 2024, doi: 10.22937/IJCSNS.2024.24.5.6.

[10] M. L. Ali and Z. Zhang, “The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection,” *Computers*, vol. 13, no. 12, p. 336, Dec. 2024, doi: 10.3390/computers13120336.

[11] R. Sapkota and M. Karkee, “Ultralytics YOLO Evolution: An Overview of YOLO26, YOLO11, YOLOv8 and YOLOv5 Object Detectors for Computer Vision and Pattern Recognition,” Oct. 2025.

[12] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, “YOLOv8 to YOLO11: A Comprehensive Architecture In-depth Comparative Review,” Apr. 2025.

[13] S. S. Rhomadhon and D. R. Ningtias, “Developing a classification system for brain tumors using the ResNet152V2 CNN model architecture,” *Journal of Soft Computing Exploration*, vol. 5, no. 2, pp. 173–182, Jun. 2024, doi: 10.52465/joscex.v5i2.372.

[14] Y. B. A. Sembiring and E. Indra, “MRI Image Classification Analysis of Brain Cancer Using ResNet18 and VGG16 Deep Learning Architectures,” *INFOKUM*, vol. 13, no. 5, pp. 1537–1547, 2025, doi: 10.58471/infokum.v13i05.

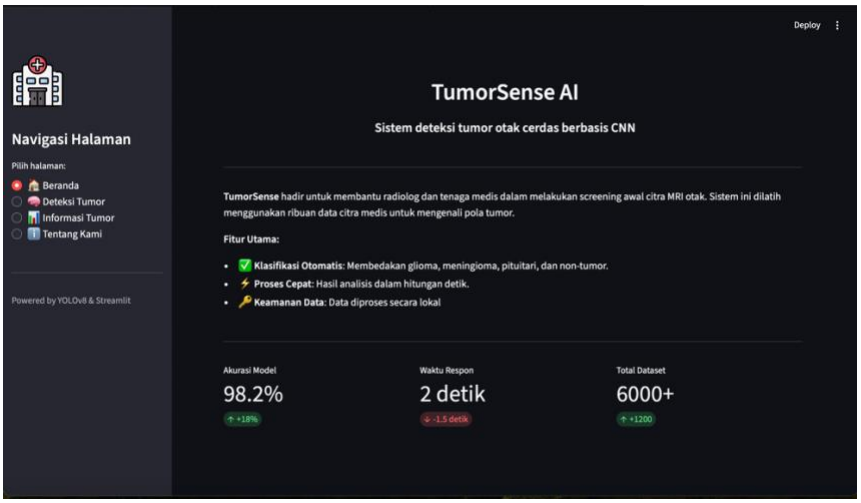
[15] M. Nickparvar, “Brain Tumor MRI Dataset,” 2021, *Kaggle*. doi: 10.34740/KAGGLE/DSV/2645886.

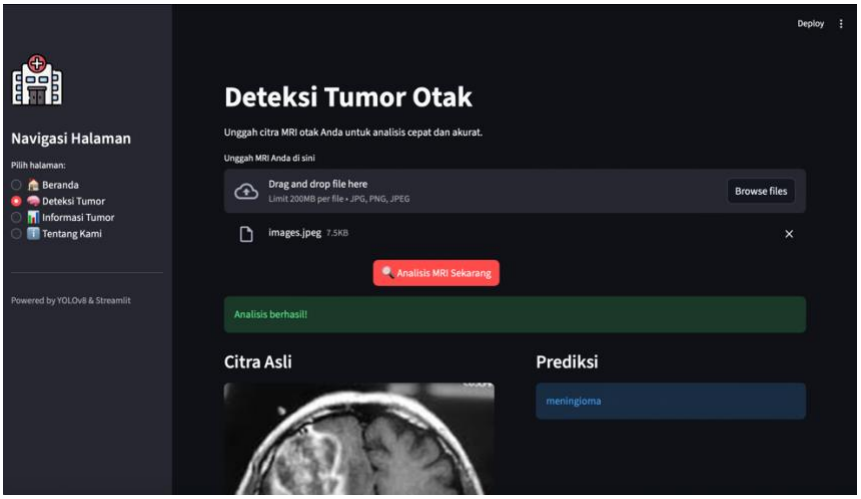
BAB 8
LAMPIRAN

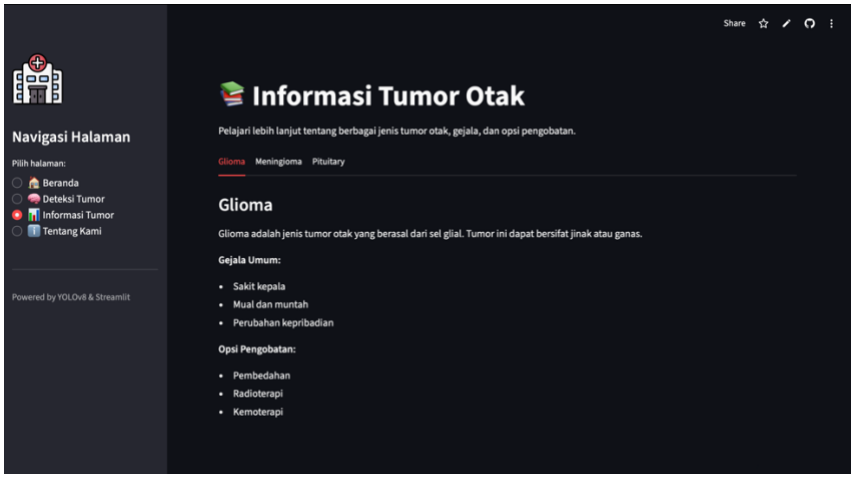
8.1 Pernyataan Kontribusi

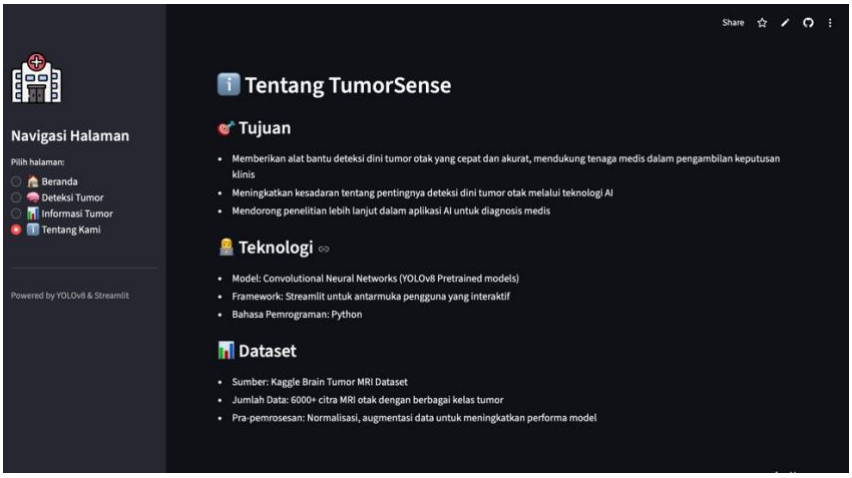
Nama	Kontribusi
Jason Evansaputra Sudargo	Membuat model dan membuat laporan
Moeh Adji Anggalaksana	Membuat *frontend* dan membuat laporan

8.2 Screenshoot



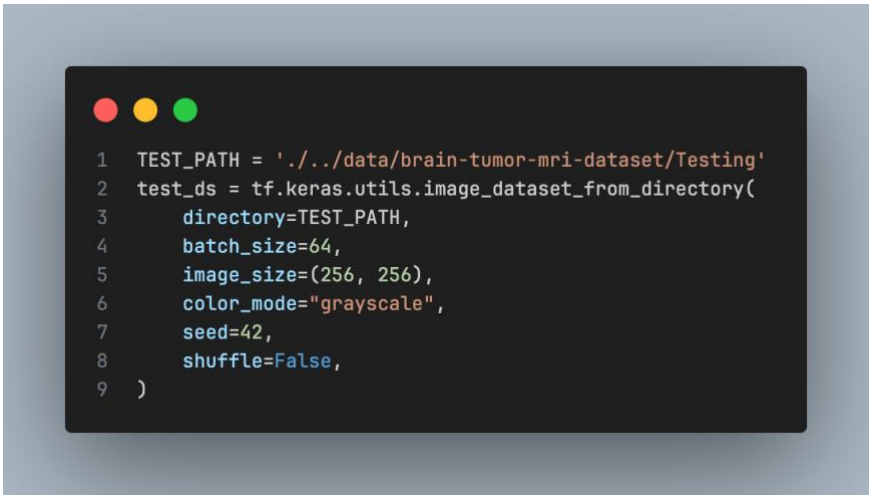






8.3 Code Snippets

```python
TRAIN_PATH = f'./../data/brain-tumor-mri-dataset/Training'
train_ds, val_ds = tf.keras.utils.image_dataset_from_directory(
    directory=TRAIN_PATH,
    batch_size=64,
    image_size=(256, 256),
    validation_split=0.3,
    subset="both",
    seed=42,
    color_mode="grayscale"
)
```



```python
TEST_PATH = './../data/brain-tumor-mri-dataset/Testing'
test_ds = tf.keras.utils.image_dataset_from_directory(
    directory=TEST_PATH,
    batch_size=64,
    image_size=(256, 256),
    color_mode="grayscale",
    seed=42,
    shuffle=False,
)
```

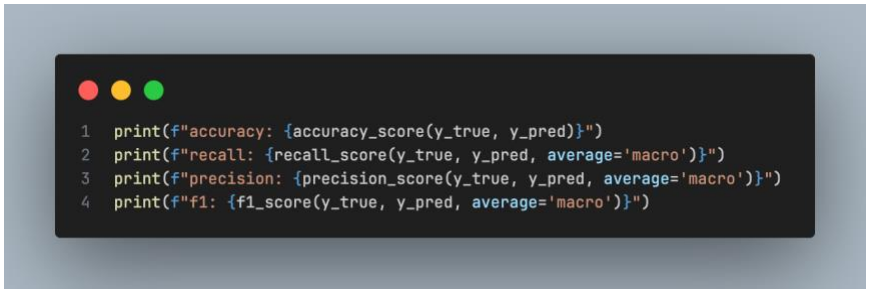
```python
norm_layer = Normalization()
norm_layer.adapt(data=train_ds.map(lambda x, y: x))
```

```python
data_augmentation = Sequential([
    RandomFlip("horizontal_and_vertical"),
    RandomRotation(0.2),
    RandomZoom(0.1)
])

model = Sequential([
    InputLayer(input_shape),
    data_augmentation,
    norm_layer,
    Conv2D(32, 3, activation="relu", kernel_initializer="he_normal", kernel_regularizer=l2(0.0001)),
    MaxPooling2D(),
    Conv2D(64, 3, activation="relu", kernel_initializer="he_normal", kernel_regularizer=l2(0.0001)),
    MaxPooling2D(),
    Conv2D(64, 3, activation="relu", kernel_initializer="he_normal", kernel_regularizer=l2(0.0001)),
    MaxPooling2D(),
    Flatten(),
    Dense(128, activation="relu", kernel_initializer="he_normal", kernel_regularizer=l2(0.0001)),
    Dropout(0.2),
    Dense(64, activation="relu", kernel_initializer="he_normal", kernel_regularizer=l2(0.0001)),
    Dense(32, activation="relu", kernel_initializer="he_normal", kernel_regularizer=l2(0.0001)),
    Dense(len(train_ds.class_names), activation="softmax")
])
```

```python
yolo_model = YOLO("yolov8n-cls.pt")

results = yolo_model.train(
    data="./../data/brain-tumor-mri-dataset-yolo",
    epochs=20,
    imgsz=256,
    batch=32,
    augment=True
)
```



```python
print(f"accuracy: {accuracy_score(y_true, y_pred)}")
print(f"recall: {recall_score(y_true, y_pred, average='macro')}")
print(f"precision: {precision_score(y_true, y_pred, average='macro')}")
print(f"f1: {f1_score(y_true, y_pred, average='macro')}")
```

```python
def home_page():
    st.markdown("""
        <div style="text-align: center;">
            <h2>TumorSense AI</h2>
            <h5>Sistem deteksi tumor otak cerdas berbasis CNN</h5>
        </div>
    """, unsafe_allow_html=True)

    st.markdown("---")

    st.markdown("""
    **TusorSense** hadir untuk membantu radiolog dan tenaga medis dalam melakukan screening awal citra MRI otak. Sistem ini dilatih menggunakan ribuan data citra medis untuk mengenali pola tumor.

    **Fitur Utama:**
    - ✅ **Klasifikasi Otomatis**: Membedakan glioma, meningioma, pituitari, dan non-tumor.
    - ⚡ **Proses Cepat**: Hasil analisis dalam hitungan detik.
    - 🔑 **Keamanan Data**: Data diproses secara lokal
    """)

    st.markdown("---")

    col1, col2, col3 = st.columns(3)
    col1.metric("Akurasi Model", "98.2%", "+18%")
    col2.metric("Waktu Respon", "2 detik", "-1.5 detik")
    col3.metric("Total Dataset", "6000+", "+1200")
```

```python
def detect_tumor_page():
    st.markdown("""
    # Deteksi Tumor Otak
    Unggah citra MRI otak Anda untuk analisis cepat dan akurat.
    """)

    f = st.file_uploader('Unggah MRI Anda di sini', ['jpg', 'png', 'jpeg'], accept_multiple_files=False)

    if f is not None:
        img = Image.open(f)

        c1, c2, c3 = st.columns([1, 2, 1])
        with c2:
            analyze_button = st.button("🔍 Analisis MRI Sekarang", type='primary')

        if analyze_button:
            predicted_class = predict(model, img)
            st.success("Analisis berhasil!")

            res_col1, res_col2 = st.columns(2, gap="large")
            with res_col1:
                st.markdown("### Citra Asli")
                st.image(img, width="stretch", caption="Input MRI")

            with res_col2:
                st.markdown("### Prediksi")
                st.info(predicted_class)

    st.markdown("---")
```

```python
from ultralytics import YOLO
from PIL import ImageFile

def predict(model: YOLO, input_data: ImageFile):
    results = model.predict(input_data)

    top_class_idx = results[0].probs.top1
    top_class_name = results[0].names[top_class_idx]

    return top_class_name
```