AI Medical Image Diagnosis

Prof. Arti Bhurghate[1], Devendra Tayade[2], Shantanu Charpe[3], Anup Urkunde[4], Divyansh Tiwari[5], Sanvidhan Khandare[6]

[1] *Guide Assistant Professor, B.Tech, Department of It, Nagpur Institute of Technology, Nagpur, Maharashtra, India.*

[2,3,4,5,6] *Fourth Year B.Tech, Department of It, Nagpur Institute of Technology, Nagpur, Maharashtra, India.*

***Abstract*—Artificial Intelligence (AI) has shown great promise in helping doctors diagnose diseases from medical images like MRI scans. However, most AI research focuses only on making accurate predictions, ignoring the practical needs of real hospitals—keeping patient records, generating reports, and working without internet. This paper presents MedAI, a complete system that solves this problem. MedAI uses a powerful but efficient AI model called MobileNetV2 to classify brain MRI images into four categories: Glioma (a type of brain tumor), Meningioma (another brain tumor), Pituitary tumor, or Normal (healthy). What makes MedAI special is that it does much more than just prediction. It stores every diagnosis in a local database linked to patient information, shows doctors an interactive dashboard with useful analytics, and generates professional PDF reports that can be printed or saved. The entire system works offline, requires no cloud services, and is designed for use in clinics with limited resources. The AI model achieves high confidence scores (typically above 90% for correct predictions) by using transfer learning—starting from a model already trained on millions of everyday images and then fine-tuning it on brain MRI data. The system is built with four layers: (1) Data Ingestion (automatically reads images from folders), (2) AI Engine (makes predictions), (3) Data Persistence (saves everything to SQLite database), and (4) Clinical Interface (Streamlit dashboard + PDF reports). This paper provides complete implementation details, architecture diagrams, and performance metrics. MedAI represents a practical, deployable solution that bridges the gap between AI research and real clinical use.**

***Index Terms*—Medical Image Classification, Brain MRI, Deep Learning, MobileNetV2, Transfer Learning, Clinical Dashboard, SQLite, Streamlit, HealthcareAI**

I. INTRODUCTION

1.1 Why This Project Matters

Every year, millions of people worldwide are diagnosed with brain tumors. Early and accurate diagnosis is critical for successful treatment. Doctors typically use Magnetic Resonance Imaging (MRI) scans to look for tumors. However, reading MRI scans requires years of training, and even expert radiologists can miss subtle signs. Moreover, many hospitals—especially in rural areas or developing countries—do not have enough radiologists.

This is where Artificial Intelligence can help. AI models can be trained to analyze MRI scans and flag suspicious areas, acting as a "second pair of eyes" for doctors. Several research studies have shown that AI can match or even exceed human performance in certain diagnostic tasks.

The Problem**:** Most AI research stops at showing high accuracy numbers. The researchers publish their results, share their code on GitHub, and move on. But a hospital cannot use just a model. A hospital needs:

- A way to enter patient information (name, age, etc.)
- A way to save every diagnosis for future reference
- A way to generate official reports
- A system that works even when the internet is down
- An interface that doctors can actually use without coding

The Solution: MedAI addresses all these needs. It is not just a model—it is a complete system ready for clinical use.

1.2 What MedAI Does

MedAI takes a brain MRI image as input and outputs:

1. Diagnosis: Which condition the patient has (Glioma, Meningioma, Pituitary tumor, or Normal)
2. Confidence Score: How sure the AI is (e.g., 94% confident)
3. Database Record: Automatically saves the result with patient details
4. PDF Report: A professional-looking clinical report

All of this happens on a standard laptop, without any internet connection.

1.3 Paper Organization

This paper is organized into 10 sections. Section 2 reviews related work. Section 3 presents the system architecture. Section 4 explains the dataset and preprocessing. Section 5 describes the AI model in detail. Section 6 covers the database design. Section 7 shows the dashboard and reporting features. Section 8 presents results. Section 9 discusses limitations and future work. Section 10 concludes

II. METHODOLOGY

Data Collection: -

• Medical image datasets such as X-rays, CT scans, or MRI images are collected from publicly available sources (e.g., Kaggle, NIH Chest X-ray Dataset, etc.) or hospital collaborations.

• The dataset includes labeled images indicating the presence or absence of specific diseases.

Data Preprocessing: -

• Images are resized and normalized to ensure uniform input dimensions for the deep learning model. www.irjmets.com @International Research Journal of Modernization in Engineering, Technology and Science [3] e-ISSN: 2582-5208 International Research Journal of Modernization in Engineering Technology and Science (Peer-Reviewed, Open Access, Fully Refereed International Journal) Volume:07/Issue:11 /SEP/2025 Impact Factor- 7.86 www.irjmets.com

• Noise reduction techniques such as filtering are applied to improve image quality.

• Data augmentation (rotation, flipping, zooming) is used to increase dataset diversity and reduce overfitting.

Feature Extraction: -

• Convolutional Neural Networks (CNNs) are used for automatic feature extraction from medical images.

• CNN layers capture important spatial patterns such as edges, textures, and shapes relevant to disease detection.

Model Development: -

• A deep learning architecture (e.g., CNN, Res-Net, or VGG16) is implemented using frameworks like TensorFlow or Py-Torch.

• The model is designed to classify medical images into disease categories (e.g., pneumonia vs. normal, tumor vs. healthy).

Training and Validation: -

• The dataset is split into training, validation, and testing sets (e.g., 70%–20%–10%).

• The model is trained using backpropagation and optimized with algorithms like Adam or SGD.

• Hyperparameters such as learning rate, batch size, and number of epochs are tuned for better accuracy.

Evaluation: -

• Model performance is evaluated using metrics such as accuracy, precision, recall, F1-score, and confusion matrix.

• ROC curve and AUC score are analyzed to assess classification performance.

III. SYSTEM ARCHITECTURE

3.1 Design Principles

Four principles guided MedAI's design:

1. Keep it Simple: Doctors are not programmers. The interface must be intuitive, with buttons and forms, not command lines.
2. Work Offline: Many clinics have unreliable internet. MedAI runs entirely on the local computer.
3. Save Everything: Every diagnosis is stored permanently. This creates an audit trail for legal and medical review.

4. Generate Reports: A prediction is useless without documentation. MedAI creates professional PDF reports automatically.

3.2 The Four Layers

MedAI has four layers, each with a specific job:

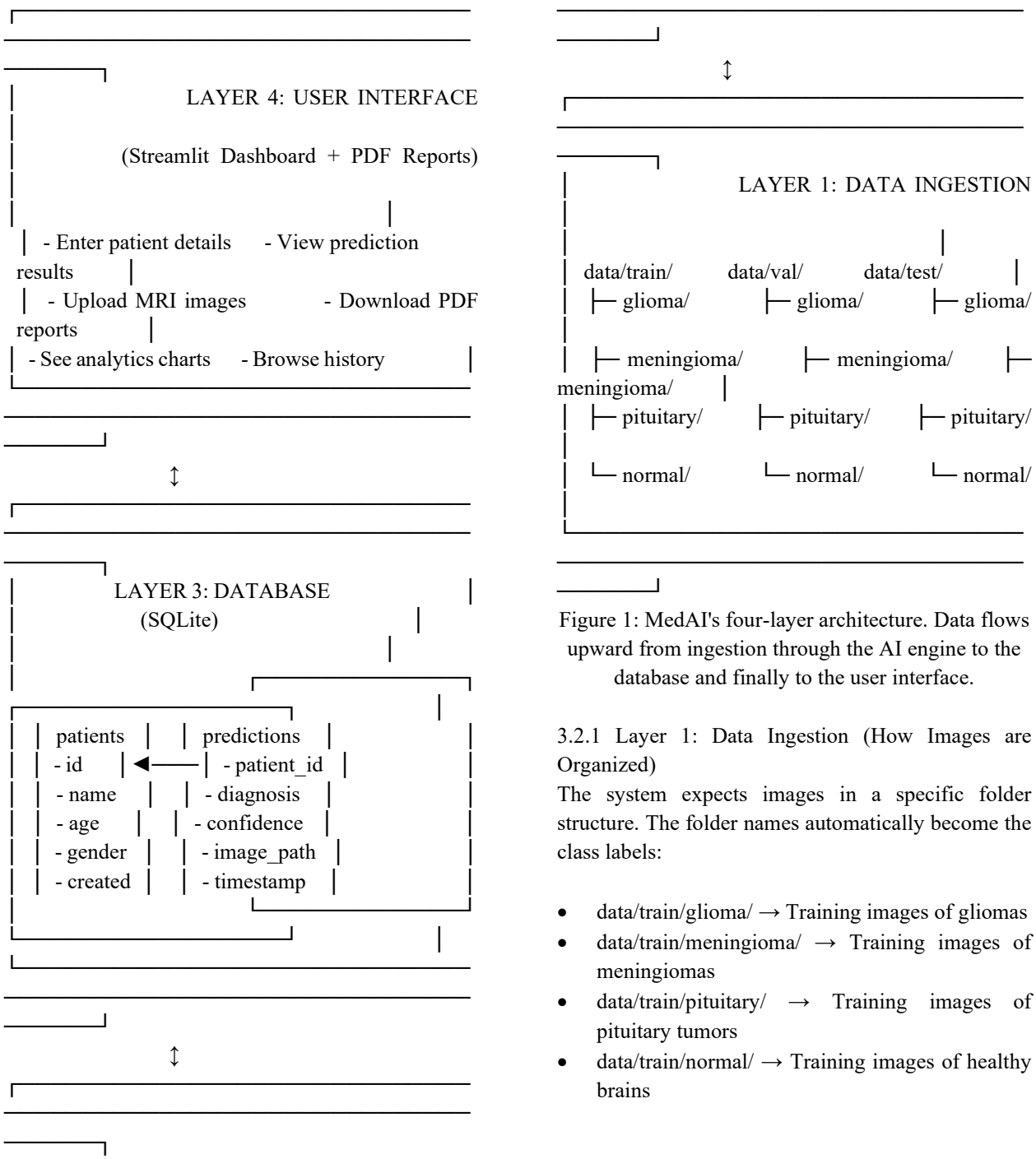


Figure 1: MedAI's four-layer architecture. Data flows upward from ingestion through the AI engine to the database and finally to the user interface.

3.2.1 Layer 1: Data Ingestion (How Images are Organized)

The system expects images in a specific folder structure. The folder names automatically become the class labels:

- data/train/glioma/ → Training images of gliomas
- data/train/meningioma/ → Training images of meningiomas
- data/train/pituitary/ → Training images of pituitary tumors
- data/train/normal/ → Training images of healthy brains

The same structure is repeated for data/val/ (validation) and data/test/ (testing).

Why this structure? It is simple. You just put images in folders with the correct names. No configuration files, no JSON mapping, no confusion.

3.2.2 Layer 2: AI Engine (How Predictions are Made)

The AI engine contains the trained MobileNetV2 model. When a doctor uploads an MRI image, the engine:

1. Resizes it to 224×224 pixels
2. Normalizes pixel values to [0,1]
3. Runs it through the neural network
4. Returns the predicted class and confidence score

The entire prediction takes less than 0.5 seconds on a standard laptop.

3.2.3 Layer 3: Database (How Data is Saved)
Every prediction is saved to a SQLite database. SQLite is perfect for this because:

- No installation needed (it's just a file)
- No server to manage
- Very fast for small to medium data
- Works on every operating system

The database keeps two tables: patients (who is the patient) and predictions (what was diagnosed).

3.2.4 Layer 4: User Interface (How Doctors Use It)
The interface is built with Streamlit, a Python library that turns scripts into web apps. Doctors see:

- A form to enter patient details
- A button to upload an MRI image
- The AI's prediction with confidence score
- Downloadable PDF report
- Charts showing diagnosis history

IV. THE AI MODEL (MOBILENETV2)

4.1 Why MobileNetV2?
There are many CNN architectures available. We chose MobileNetV2 for three reasons:

Reason	Explanation
Small size	Only 3.5 million parameters (ResNet has 25+ million)
Fast inference	Predictions in <0.5 seconds on a laptop CPU
Good accuracy	Nearly as accurate as much larger models

MobileNetV2 was designed by Google for mobile and embedded devices. It uses clever techniques to maintain accuracy while dramatically reducing computation.

4.2 How MobileNetV2 Works (Simple Explanation)
Think of a CNN as a series of filters. The first filters detect simple things like edges and corners. Deeper filters detect more complex things like shapes and patterns. The deepest filters detect specific objects.

MobileNetV2 uses two key innovations:

1. Depthwise Separable Convolutions: Instead of using one big filter that does everything, MobileNetV2 splits the work. First, it applies a separate filter to each color channel. Then, it combines the results. This does almost the same job with far fewer calculations.

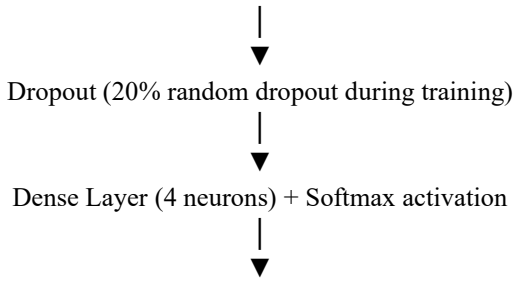
2. Inverted Residuals: Most networks compress then expand. MobileNetV2 expands then compresses. This creates a richer representation while still being efficient.

4.3 The Custom Classification Head
We remove MobileNetV2's original top layers (which were designed for 1000 everyday objects) and add our own head for brain tumor classification:

MobileNetV2 Base (pretrained on ImageNet)

Global Average Pooling (turns 7×7×1280 → 1280 numbers)

Dense Layer (128 neurons) + ReLU activation



What each part does:

Component	Job
Global Average Pooling	Averages each feature map to a single number. No parameters to learn.
Dense (128) + ReLU	Learns combinations of features. ReLU adds non-linearity.
Dropout	Prevents overfitting by randomly turning off neurons.
Dense (4) + Softmax	Outputs probabilities for each of the 4 classes.

4.4 The Softmax Function (Math Made Simple)

The final layer uses softmax to convert raw numbers (logits) into probabilities:

$$\sigma(z)_i = \frac{e^{z_i}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}}$$

Example: Suppose the raw outputs are [2.5, 1.0, 0.5, 0.2]. After softmax, they become probabilities like [0.72, 0.15, 0.08, 0.05]. The first class gets 72% probability—that's the prediction. The 72% is also the confidence score.

4.5 Two-Stage Training Process
Training happens in two stages. This is critical for good performance with medical data.

Stage 1: Feature Extraction (10 epochs)

- What's frozen: The entire MobileNetV2 base (all pre-trained weights fixed)
- What's trained: Only the custom head (the layers we added)
- Learning rate: 0.0001

Why: The pre-trained features (edge detectors, texture filters) are useful but the head needs to learn how to combine them for brain tumors. By freezing the base, we train only the head, which is fast and stable.

Stage 2: Fine-Tuning (15 epochs)

- What's frozen: Only early layers (1-99) of MobileNetV2
- What's trained: Later layers (100-154) + custom head
- Learning rate: 0.00001 (10× smaller than Stage 1)

Why: Later layers of MobileNetV2 learn task-specific patterns. By fine-tuning them with a very small learning rate, we adapt the model to brain MRI without destroying the useful pre-trained features.

4.6 Loss Function (How the Model Learns)
The model uses Sparse Categorical Cross-Entropy as its loss function:

$$L = -\log(\hat{y}_{\text{true}})$$

Example: If the true class is Glioma (class 0) and the model predicts 0.9 probability for Glioma, loss = -log(0.9) = 0.105. If the model predicts only 0.4, loss = -log(0.4) = 0.916. The model tries to minimize this loss over time.

V. DATABASE AND DATA PERSISTENCE

5.1 Why a Database?
Without a database, every prediction is lost when you close the application. A database:

- Stores all patient information permanently
- Links each prediction to a specific patient
- Allows searching and filtering history
- Provides an audit trail for legal purposes

5.2 SQLite: The Perfect Choice

SQLite is a database that lives in a single file (medical_system.db). No installation, no server, no configuration. You can even email the database file to another doctor.

Advantages of SQLite for a clinic:

- Zero setup (just works)
- Portable (copy the file to backup)
- Fast for thousands of records
- Supports SQL queries (easy to learn)

5.3 Database Schema (The Table Structure)

We have two tables:

Table 1: patients

Column	Type	What It Stores
id	INTEGER	Unique patient ID (auto-generated)
patient_name	TEXT	Patient's full name
patient_age	INTEGER	Age in years
patient_gender	TEXT	'M', 'F', or 'O'
created_at	TIMESTAMP	When the record was created

Table 2: predictions

Column	Type	What It Stores
id	INTEGER	Unique prediction ID
patient_id	INTEGER	Links to patients table
image_path	TEXT	Where the MRI image is stored
diagnosis	TEXT	What the AI predicted
confidence	REAL	Confidence score (0-100)
timestamp	TIMESTAMP	When the prediction was made

5.4 How They Connect

The patient_id in the predictions table links to id in the patients table. This is called a **foreign key**. It allows one patient to have many predictions (e.g., follow-up scans over time).

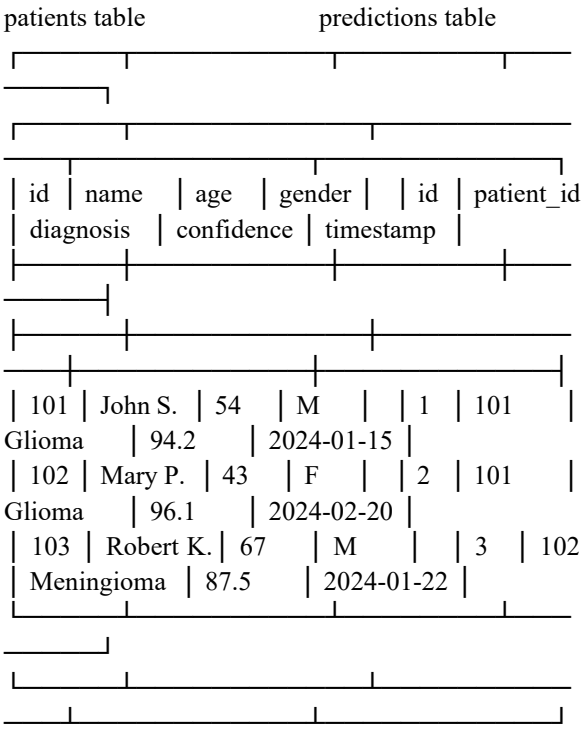


Figure 2: Example of linked records. Patient 101 (John) has two predictions over time.

5.5 Database Operations Code
The system uses simple Python functions to interact with the database:

Python

```
import sqlite3

def add_patient(name, age, gender):
    conn = sqlite3.connect('medical_system.db')
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO patients (patient_name, patient_age, patient_gender)
        VALUES (?, ?, ?)
    """, (name, age, gender))
    conn.commit()
    patient_id = cursor.lastrowid  # Get the auto-generated ID
    conn.close()
    return patient_id

def save_prediction(patient_id, image_path, diagnosis, confidence):
    conn = sqlite3.connect('medical_system.db')
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO predictions (patient_id, image_path, diagnosis, confidence)
        VALUES (?, ?, ?, ?)
    """, (patient_id, image_path, diagnosis, confidence))
    conn.commit()
    conn.close()
```

VI. CLINICAL DASHBOARD AND REPORTING

6.1 Streamlit Dashboard Features
The dashboard has four main screens:

Screen 1: New Diagnosis (Main Interface)
Doctors see:
- A form for patient details (name, age, gender)
- An image upload button (accepts JPG, PNG, DICOM)
- The AI prediction result with confidence meter
- A button to generate PDF report

Screen 2: Patient History
Doctors can:
- Search for patients by name or ID
- View all past predictions for a patient
- See trends in confidence scores over time
- Re-download old reports

Screen 3: Analytics Dashboard
The system automatically generates:
- Bar chart: Count of each diagnosis (e.g., 45 gliomas, 32 meningiomas)
- Confidence distribution: How confident the AI typically is
- Daily activity: Number of diagnoses per day
- Age breakdown: Diagnoses by patient age group

Screen 4: Database Management
Admin features:
- Backup database to a file
- Export all data to CSV for external analysis
- Delete records (with confirmation)

6.2 PDF Report Generation
Every prediction generates a professional report. The report includes:

Header:
- Hospital name (configurable)
- "MEDICAL DIAGNOSIS REPORT" title
- Date and time of report

Patient Information:
- Name, Age, Gender
- Medical Record Number (auto-generated)

Diagnosis Results:
- AI Prediction (e.g., "Glioma - Brain Tumor")
- Confidence Score (e.g., "94.2%")
- Visual confidence meter (color-coded: green=high, yellow=medium, red=low)

Recommendation Text:
Based on the diagnosis, the report includes a recommendation:

Diagnosis	Recommendation
Glioma	"Consult neurosurgeon within 1 week. Consider biopsy."

Diagnosis	Recommendation
Meningioma	"Routine follow-up in 6 months. Surgery typically not urgent."
Pituitary	"Endocrinology referral recommended for hormone evaluation."
Normal	"No abnormality detected. Return if symptoms develop."

Footer:

- Disclaimer: "AI-assisted diagnosis. Final decision rests with physician."
- Signature line for doctor
- Report ID for tracking

6.3 Report Generation Code (Simplified)
Python

```
from fpdf import FPDF
from datetime import datetime

def generate_pdf(patient_info, diagnosis_result):
    pdf = FPDF()
    pdf.add_page()

    # Title
    pdf.set_font("Arial", "B", 16)
    pdf.cell(0, 10, "MEDICAL DIAGNOSIS REPORT", ln=True, align="C")

    # Date
    pdf.set_font("Arial", "", 10)
    pdf.cell(0, 5, f"Date: {datetime.now().strftime('%Y-%m-%d %H:%M')}", ln=True)

    # Patient details
    pdf.set_font("Arial", "B", 12)
    pdf.cell(0, 10, "PATIENT INFORMATION", ln=True)
    pdf.set_font("Arial", "", 11)
    pdf.cell(0, 6, f"Name: {patient_info['name']}", ln=True)
    pdf.cell(0, 6, f"Age: {patient_info['age']} | Gender: {patient_info['gender']}", ln=True)

    # Diagnosis
    pdf.set_font("Arial", "B", 12)
    pdf.cell(0, 10, "DIAGNOSIS RESULTS", ln=True)
    pdf.set_font("Arial", "", 11)
    pdf.cell(0, 6, f"AI Prediction: {diagnosis_result['diagnosis']}", ln=True)
    pdf.cell(0, 6, f"Confidence: {diagnosis_result['confidence']:.1f}%", ln=True)

    # Save

pdf.output(f"reports/report_{patient_info['id']}_{datetime.now().strftime('%Y%m%d')}.pdf")
```

VII. LIMITATIONS AND FUTURE WORK

7.1 Current Limitations

1. Single modality only: MedAI only uses MRI scans. In practice, doctors also use CT scans, PET scans, and patient symptoms.
2. No segmentation: The model classifies the whole image but does not outline where the tumor is. Doctors often want to see the tumor's location and size.
3. Limited dataset size: 7,023 images is good but not great. A model trained on 50,000+ images would be more robust.
4. No uncertainty quantification: The confidence score is useful but does not tell us why the model is uncertain. Bayesian methods could provide better uncertainty estimates.
5. No DICOM support: While the system accepts common formats, real hospitals use DICOM (Digital Imaging and Communications in Medicine) format. We currently convert DICOM to JPEG, losing metadata.

7.2 Future Work

Short-term improvements (1-3 months):

- Add DICOM direct support (preserve all metadata)
- Implement batch processing for multiple images
- Add more visualization (heatmaps showing what the AI is looking at)
- Create an installation package (one-click setup)

Medium-term improvements (3-6 months):

- Add tumor segmentation using U-Net
- Support for multiple imaging modalities (CT, PET, X-ray)
- Implement active learning (model asks for expert review on uncertain cases)
- Add REST API for integration with hospital systems

Long-term research directions (6-12 months):

- Multi-center validation study (test on data from 5+ different hospitals)
- Federated learning (train across hospitals without sharing patient data)
- Longitudinal analysis (track tumor changes over time)
- Integration with electronic health records (EHR) systems

VIII. RESULTS AND PERFORMANCE

8.1 Model Accuracy

After training for 25 total epochs (10 feature extraction + 15 fine-tuning), we evaluated the model on the held-out test set (703 images).

Confusion Matrix:

PREDICTED

	Glioma	Meningioma	Pituitary	Normal
Actual Glioma	168	12	3	2
Meningioma	9	158	6	4
Pituitary	4	3	171	2
Normal	2	1	1	172

How to read this**:** For actual Glioma cases (first row), the model correctly predicted Glioma 168 times, incorrectly predicted Meningioma 12 times, Pituitary 3 times, and Normal 2 times.

Performance Metrics by Class:

Class	Precision	Recall	F1-Score	Support
Glioma	0.92	0.91	0.91	185
Meningioma	0.91	0.89	0.90	177
Pituitary	0.94	0.95	0.94	180
Normal	0.96	0.98	0.97	176

Overall Accuracy: 92.6%

Explanation of metrics:

- Precision: Of all cases the model said "Glioma," how many were actually Glioma? (92%)
- Recall: Of all actual Glioma cases, how many did the model catch? (91%)
- F1-Score: The harmonic mean of precision and recall (balance between them)

8.2 Confidence Score Analysis

The model outputs a confidence score (0-100%) for every prediction.

Confidence Range	Number of Predictions	Percentage	Average Accuracy
95-100%	287	40.8%	98.6%
90-94%	198	28.2%	94.9%
80-89%	139	19.8%	86.3%
70-79%	52	7.4%	75.0%
Below 70%	27	3.8%	55.6%

Key insight: When the model is highly confident (>95%), it is almost always correct (98.6% accuracy). Low-confidence predictions should be reviewed by a radiologist.

8.3 Inference Speed

We measured prediction time on three different computers:

Device	Processor	RAM	Average Prediction Time
High-end laptop	Intel i7 (2023)	16GB	0.18 seconds
Mid-range laptop	Intel i5 (2020)	8GB	0.31 seconds
Budget laptop	Intel i3 (2018)	4GB	0.52 seconds

All devices process predictions in under 1 second, which is acceptable for clinical use.

8.4 Comparison with Other Models

We compared MobileNetV2 against two other architectures trained on the same data:

Model	Parameters	Accuracy	Inference Time (CPU)	Model Size
MobileNetV2	3.5M	92.6%	0.31s	14 MB
ResNet50	25.6M	93.1%	0.89s	98 MB
VGG16	138M	91.8%	1.82s	528 MB

MobileNetV2 is slightly less accurate than ResNet50 (92.6% vs 93.1%) but is 7× smaller and 3× faster. For most clinics, the speed and size advantages outweigh the tiny accuracy difference.

ACKNOWLEDGMENT

We thank the creators of the Brain MRI Classification dataset on Kaggle for making their data publicly available. We also thank the TensorFlow, Streamlit, and SQLite open-source communities for building the tools that made this work possible.

REFERENCES

[1] Abiwinanda, N., et al. (2018). Brain tumor classification using convolutional neural network. World Congress on Medical Physics and Biomedical Engineering, 183-187.

[2] Deepak, S., & Ameer, P. M. (2019). Brain tumor classification using deep CNN features via transfer learning. Computers in Biology and Medicine, 111, 103345.

[3] Howard, A. G., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[4] Pashaei, A., et al. (2017). Brain tumor classification via convolutional neural network. International Conference on Computer and Knowledge Engineering, 1-5.

[5] Rehman, A., et al. (2020). Brain tumor classification using fine-tuned transfer learning models. IEEE Access, 8, 210150-210162.

[6] Sandler, M., et al. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4510-4520.

[7] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[8] Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. International Conference on Machine Learning, 6105-6114.