UNIVERSITY OF HERTFORDSHIRE

School of Physics, Engineering and Computer Science

Advance master project (7COM1039)

Date: 03-09-2025

Next-Gen Brain Tumor Diagnosis: Hybrid Deep CNN, Vision Transformer, and Ensemble-Based MRI Classification Framework

Name: Sadiq Ali

Student ID: 23078216

Supervisor: Dr. Cheima Ali Bensaad

ABSTRACT

Brain tumors remain among the most life-threatening neurological disorders, and timely, accurate diagnosis is critical for improving patient outcomes. Magnetic Resonance Imaging (MRI) is the gold standard for tumor detection, yet the high variability across gliomas, meningiomas, and pituitary tumors poses significant challenges for radiologists. Deep learning has recently emerged as a powerful tool for automated medical image analysis, offering the potential for faster and more reliable decision support.

This study introduces a hybrid ensemble-based framework that integrates Convolutional Neural Networks (CNNs), transfer learning architectures, and Vision Transformers (ViTs) for robust brain tumor classification. Four model groups were explored: (i) a custom CNN trained from scratch, (ii) transfer learning with ResNet50 and DenseNet121, (iii) a Vision Transformer (DeiT-small-distilled), and (iv) an ensemble approach combining the strengths of CNN- and transformer-based architectures.

Performance was evaluated using accuracy, precision, recall, F1-score, and confusion matrices. The baseline CNN achieved 92% accuracy, showing the feasibility of lightweight scratch-trained networks. Transfer learning models performed strongly, with ResNet50 reaching 96% and DenseNet121 demonstrating reliable generalization. The ViT achieved 97.3% accuracy, excelling particularly in glioma and pituitary classification. The ensemble strategy delivered the most balanced results, reaching 97.5% accuracy and reducing class-specific misclassifications, thereby surpassing all individual models.

This work contributes to AI-driven medical imaging by systematically benchmarking multiple architectures within a unified framework. The findings highlight that hybrid and ensemble systems offer improved accuracy, robustness, and clinical potential. Future work will focus on multi-institutional validation, integration of clinical metadata, and developing lightweight models for hospital deployment.

Keywords: Brain tumor classification, MRI, Deep Learning, CNN, ResNet50, DenseNet121, Vision Transformer, Ensemble Learning.

Table of Contents

Acknowledgment

I would like to express my heartfelt gratitude to everyone who has contributed to the successful completion of this project.

First, I thank my mentor, ***DR. CHEIMA ALI BENSAAD*** *, for their invaluable guidance, insightful, feedback, and unwavering support throughout this journey. I am equally grateful to* ***UNIVERSITY OF HERTFORDSHIRE*** *for providing the resources and platforms to pursue this research.*

I extend my thanks to my peers and colleagues for their constructive discussions and encouragement, which has enhanced the quality of my work.

Finally, I am deeply thankful to my family and friends for their unwavering patience, motivation, and support without which this project would not have been possible.

Thank you for your contribution and encouragement.

MSc Final Project Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science and Analytics at the University of Hertfordshire (UH).

It is my own work except where indicated in the report.

I did not use human participants in my MSc Project.

Chapter 1

Introduction

1.1 Problem Overview

Brain tumors are among the most devastating diseases of the central nervous system (CNS), with profound implications for patient survival, neurological function, and quality of life. They represent a heterogeneous group of malignancies, including gliomas, meningiomas, and pituitary adenomas, each exhibiting unique molecular characteristics, growth patterns, and treatment challenges [1]. According to the most recent global cancer statistics, brain and other CNS tumors accounted for over 308,000 new cases and 251,000 deaths worldwide in 2020, underscoring their significant health burden and the urgent need for improved diagnostic strategies [2]. Despite ongoing advances in neurosurgery, chemotherapy, and radiotherapy, the overall prognosis for malignant brain tumors remains poor, with five-year survival rates for glioblastoma, the most aggressive glioma subtype, still below 10% [1].

Early and accurate diagnosis is a cornerstone of effective treatment planning, as therapeutic outcomes are strongly influenced by the timeliness of intervention. Magnetic Resonance Imaging (MRI) has long been established as the gold standard for brain tumor detection due to its ability to capture high-resolution anatomical details without ionizing radiation exposure [3]. MRI enables the visualization of tumor boundaries, edema, and tissue heterogeneity across multiple sequences, including T1, T2, and FLAIR. However, despite its diagnostic power, the interpretation of MRI scans remains a manual, labor-intensive, and subjective process. Inter-rater variability between radiologists is well documented, with differences in tumor delineation and grading reaching up to 20% in complex cases [4]. This not only increases the risk of delayed treatment but also highlights the limitations of human-based diagnostic workflows in handling the growing volume of medical imaging data generated globally.

In recent years, the integration of Artificial Intelligence (AI) and deep learning techniques into neuroimaging has emerged as a transformative approach to address these limitations. Convolutional Neural Networks (CNNs), in particular, have demonstrated remarkable success in brain tumor classification, segmentation, and detection tasks [5]. By automatically learning

hierarchical representations of image features, CNNs have surpassed traditional machine learning methods that relied on handcrafted features, achieving accuracies exceeding 90% on benchmark datasets [6]. Their ability to extract discriminative spatial features makes them highly effective in distinguishing between tumor and non-tumor tissues, as well as among different tumor types.

However, CNN-based methods are not without constraints. They rely on local receptive fields, which limits their ability to model long-range dependencies in MRI scans [7]. Furthermore, CNNs trained on a single dataset often fail to generalize to unseen clinical environments due to variations in imaging protocols, patient demographics, and scanner hardware. Another persistent challenge is class imbalance, where certain tumor types, such as pituitary adenomas, are underrepresented compared to gliomas or meningiomas. This imbalance biases model training, leading to poor recall and underdiagnosis of minority tumor classes [8]. These limitations raise concerns about the robustness of CNN-based diagnostic tools when deployed in real-world clinical practice.

To overcome these challenges, new architectures such as Vision Transformers (ViTs) have been proposed. Inspired by the success of transformers in natural language processing, ViTs divide medical images into fixed-size patches and apply self-attention mechanisms to capture global contextual relationships across the image [9]. This allows ViTs to model both local and global dependencies more effectively than CNNs, addressing one of the core weaknesses of convolutional architectures. Early studies in neuroimaging have reported competitive or even superior performance of ViTs compared to CNNs, particularly in multi-class brain tumor classification tasks [9]. Nonetheless, ViTs demand larger training datasets and greater computational resources, which poses a barrier for smaller institutions and research groups.

Another promising strategy to enhance diagnostic accuracy is ensemble learning, which integrates multiple models to produce a single, more robust prediction. By combining the complementary strengths of architectures such as CNNs, ResNets, DenseNets, and Transformers, ensemble frameworks have been shown to improve classification accuracy, reliability, and resistance to dataset variability [5], [8]. Importantly, these approaches align with clinical priorities by reducing the likelihood of false negatives and increasing confidence in automated predictions.

Beyond accuracy and efficiency, the issue of interpretability remains critical for the adoption of AI in healthcare. Deep learning models are often regarded as “black boxes,” limiting their trustworthiness in sensitive medical contexts. Methods such as Gradient-weighted Class

Activation Mapping (Grad-CAM) for CNNs and attention rollouts for transformers have been proposed to enhance transparency by highlighting the regions of MRI scans that contribute most strongly to predictions [7]. These explainability tools serve as an essential bridge between algorithmic decision-making and clinical practice, fostering greater trust among radiologists and neurosurgeons.

In summary, the increasing burden of brain tumors and the limitations of manual diagnostic processes underscore the need for robust, automated, and interpretable AI-based systems. CNNs, while effective, struggle with generalization and minority-class sensitivity; ViTs offer improved global context modeling but demand significant resources; and ensembles promise to deliver balanced performance by integrating multiple architectures. This project builds upon these developments by implementing a hybrid classification framework incorporating CNNs, DenseNet121, ResNet50, Vision Transformers, and ensemble strategies. The goal is to design a system that not only achieves high classification accuracy but also addresses issues of fairness, interpretability, and clinical feasibility, thereby bridging the gap between algorithmic research and real-world neuro-oncology applications.

1.2 Research Questions and Objectives

Artificial Intelligence (AI) has made significant inroads into the field of medical imaging, with applications ranging from segmentation and detection to classification of complex pathologies. In the domain of brain tumor diagnosis, deep learning techniques have become particularly prominent due to their ability to automatically extract relevant features from raw MRI scans without relying on handcrafted descriptors. However, while the promise of these models is well recognized, there remain fundamental scientific questions and practical objectives that must be addressed in order to advance the field from proof-of-concept experiments toward clinical viability.

One of the key motivations for this study is the recognition that no single model architecture provides a universally superior solution for brain tumor classification. Convolutional Neural Networks (CNNs), for example, have achieved strong baseline results on benchmark datasets by effectively capturing spatial hierarchies in images [10]. Nevertheless, CNNs are often constrained by their limited receptive field, which restricts their capacity to capture global dependencies. In contrast, transfer learning models such as ResNet50 and DenseNet121 have leveraged ImageNet-

pretrained weights to provide improved performance on relatively small medical imaging datasets [11], but they still suffer from issues of generalization when exposed to data from different centers or imaging protocols [12].

Recent years have also seen growing interest in Vision Transformers (ViTs), which utilize self-attention mechanisms to capture long-range dependencies in images, thereby addressing some of the limitations of CNNs [13]. However, transformers generally require large amounts of data and significant computational resources, which raises feasibility concerns for smaller-scale research projects or low-resource clinical environments [14]. The combination of these models through ensemble methods offers a promising way forward, as ensembles can integrate the complementary strengths of CNNs, ResNets, DenseNets, and ViTs to produce more robust predictions [15].

At the same time, challenges such as class imbalance, limited dataset size, and computational demands remain persistent obstacles. Tumor classes such as pituitary adenomas are often underrepresented in open-source datasets like the Figshare Brain Tumor Dataset [16], making it difficult for models to achieve balanced performance across all categories. Without carefully designed preprocessing strategies and augmentation pipelines, models risk overfitting to majority classes such as gliomas, resulting in misleadingly high overall accuracy but poor class-level sensitivity. Thus, it is not only important to develop new models, but also to rigorously examine how data balancing and augmentation strategies can shape outcomes.

In light of these challenges, this project defines a structured set of research questions and objectives. These will guide the implementation, comparison, and evaluation of multiple deep learning paradigms for brain tumor MRI classification, with the overarching goal of building a hybrid system that is both robust and scientifically rigorous.

1.2.1 Research Questions

This study seeks to answer the following core questions with sub questions as:

RQ:

How do Convolutional Neural Networks (CNNs), transfer learning models (ResNet50, DenseNet121), Vision Transformers (ViTs), and ensemble approaches compare in their ability to classify brain tumors from MRI images?

RQa:

To what extent does transfer learning improve performance relative to scratch-trained CNNs in terms of accuracy, training stability, and efficiency?

RQb:

How effective are ensemble learning strategies in enhancing model robustness and reducing misclassification of underrepresented tumor categories?

RQc:

How do preprocessing, augmentation, and class balancing techniques help in addressing the limitations of a relatively small and imbalanced dataset?

1.2.2 Objectives

The project aims to develop a hybrid deep learning system that leverages CNNs, transfer learning, Vision Transformers, and ensemble methods for robust, automated brain tumor classification from MRI.

The objectives are to:

- Implement, train, and evaluate custom and transfer learning models (ResNet50, DenseNet121, and a custom CNN), as well as Vision Transformers, for binary and multi-class brain tumor detection.
- Compare scratch-trained and transfer-learned approaches for generalizability and efficiency.
- Explore ensemble strategies to optimize predictive performance and stability.
- Utilize comprehensive data augmentation and balancing to address dataset limitations and maximize robustness.

1.3 Research Challenges in Automated Brain Tumor Classification

The application of Artificial Intelligence (AI) to brain tumor diagnosis has advanced significantly in the past decade, yet several persistent challenges continue to hinder its clinical adoption. These challenges are not only technical but also ethical, practical, and infrastructural in nature.

Addressing them is critical to ensuring that AI-based systems are not just accurate in controlled experiments but also robust, fair, and trustworthy in real-world healthcare environments.

1.3.1 Class Imbalance and Data Limitations

One of the foremost challenges is the imbalance of publicly available datasets. Brain tumor datasets, such as the Figshare Brain Tumor MRI collection, contain far fewer pituitary adenoma images compared to gliomas and meningiomas [17]. This imbalance skews model training, resulting in classifiers that achieve high overall accuracy but perform poorly on minority classes. In clinical terms, this translates into a higher likelihood of missed diagnoses for certain tumor types, which is unacceptable in high-stakes medical applications. Moreover, most available datasets are relatively small, often in the range of a few thousand samples, limiting the potential of data-hungry architectures such as Vision Transformers [18].

1.3.2 Generalization Across Clinical Settings

Another critical challenge is the generalization gap between research benchmarks and clinical deployment. Models trained on data collected from a single center or scanner often fail when tested on external datasets, due to differences in acquisition protocols, imaging sequences, and patient demographics [19]. For example, glioma MRI scans from The Cancer Imaging Archive (TCIA) differ significantly in intensity distributions and noise characteristics from those in Figshare datasets, making cross-dataset inference unreliable [20]. Without strategies such as domain adaptation or multi-center validation, deep learning models risk overfitting to narrow data distributions and failing in diverse clinical environments.

1.3.3 Interpretability and Clinical Trust

Deep learning models, despite their predictive power, are frequently described as “black boxes.” Clinicians are often reluctant to trust models that do not provide insight into the reasoning behind their predictions [21]. This issue is particularly acute in brain tumor classification, where treatment decisions such as surgical resection or radiotherapy planning depend on precise tumor localization. While tools like Grad-CAM for CNNs and attention maps for transformers offer partial interpretability, they remain imperfect and non-standardized across architectures [22]. The lack of robust interpretability frameworks undermines the acceptance of AI systems in clinical workflows.

1.3.4 Computational Complexity and Resource Constraints

Advanced architectures such as DenseNet121, ResNet50, and especially Vision Transformers require significant computational resources for both training and inference. Vision Transformers demand large memory footprints and extensive training time, which limits their use in smaller hospitals or low-resource clinical settings [23]. Even ensemble methods, while improving accuracy, further increase computational overhead. These requirements create a barrier to widespread adoption in healthcare environments where computational infrastructure may be limited.

1.3.5 Ethical, Legal, and Social Challenges

Beyond technical issues, there are ethical and regulatory challenges. Patient privacy must be protected under frameworks such as GDPR and HIPAA, particularly when training AI systems on sensitive MRI data [24]. The risk of algorithmic bias where models systematically underperform for specific demographic groups is another pressing concern [25]. Furthermore, the medico-legal implications of misdiagnosis by AI systems remain unresolved. Questions of accountability whether responsibility lies with the developer, clinician, or healthcare institution are critical barriers to clinical translation.

In summary, the development of AI-driven brain tumor classification systems faces multiple challenges: imbalanced datasets, lack of generalization, limited interpretability, high computational demands, and unresolved ethical issues. These barriers must be systematically addressed to enable robust, fair, and clinically acceptable solutions. This project directly engages with these challenges by proposing a hybrid ensemble framework that leverages CNNs, DenseNet121, ResNet50, and Vision Transformers while integrating interpretability tools and acknowledging ethical considerations.

1.4 Project Scope and Details

The development of automated brain tumor classification systems using MRI data requires a clear definition of project scope to ensure that the research remains focused, feasible, and impactful. This project is designed as a systematic exploration of multiple state-of-the-art deep learning architectures, ranging from traditional convolutional networks to modern transformer-based approaches, and concludes with ensemble strategies to combine their strengths. By constraining

the work to a specific dataset and well-defined evaluation protocols, the study balances scientific rigor with practical feasibility.

1.4.1 Dataset and Preprocessing

The project utilizes the Figshare Brain Tumor Dataset [26], a widely cited benchmark in neuroimaging research. This dataset consists of approximately 3,000 T1-weighted contrast-enhanced MRI slices, categorized into four classes: glioma, meningioma, pituitary tumor, and no tumor. The dataset's relatively modest size reflects a common challenge in medical imaging limited availability of annotated clinical data. To maximize its utility, extensive preprocessing is performed, including resizing to a standardized resolution, histogram equalization, Gaussian filtering, and normalization. Additionally, data augmentation techniques such as rotation, flipping, and brightness adjustments are applied to mitigate overfitting and partially address class imbalance.

1.4.2 Models Investigated

The project systematically explores five different modeling strategies:

1. **Custom CNN (3-layer)** – Designed and trained from scratch to serve as a baseline, highlighting the performance of a lightweight convolutional architecture on the dataset.
2. **DenseNet121** – A transfer learning approach leveraging pretrained ImageNet weights, fine-tuned on MRI data to exploit its dense connectivity for efficient feature reuse [27].
3. **ResNet50** – Another transfer learning backbone, chosen for its residual learning framework that alleviates vanishing gradient issues and facilitates deeper model training [28].
4. **Vision Transformer (ViT)** – A transformer-based architecture that divides MRI scans into patches and applies self-attention to capture long-range dependencies, providing a comparative modern approach [28].
5. **Ensemble Method** – A hybrid framework that combines predictions from CNN, DenseNet121, ResNet50, and ViT models using techniques such as **soft voting** and **stacking**, designed to improve overall robustness and classification performance [29].

1.4.3 Hardware and Feasibility

The models are implemented primarily in PyTorch, with baseline CNN experiments conducted in TensorFlow/Keras. Training is performed using modern GPUs, allowing the efficient execution of computationally intensive architectures such as ViTs. Nonetheless, feasibility considerations are taken into account: while CNN and ResNet50 are relatively lightweight, DenseNet121 and ViTs demand higher memory and longer training times. The ensemble approach adds another layer of complexity but demonstrates that combining diverse models can offset individual weaknesses.

1.4.4 Scope of the Work

The scope of the project is carefully defined:

- Restricted to a single public dataset (Figshare), ensuring controlled benchmarking but acknowledging limited external generalization.
- Focused on four classification categories (glioma, meningioma, pituitary, and no tumor).
- Comparative in nature, aiming not only to achieve high accuracy but also to evaluate the trade-offs between different architectures and their combinations systematically.
- Targeted towards research and proof-of-concept rather than direct clinical deployment.

In this way, the project aims to demonstrate the feasibility and benefits of hybrid modeling approaches for brain tumor classification, while acknowledging its limitations in terms of dataset size and external validation.

1.5 Novelty and Contribution

The field of medical image analysis has been transformed by the adoption of deep learning methods, yet brain tumor classification remains a domain characterized by ongoing challenges and research gaps. Many existing studies have relied exclusively on Convolutional Neural Networks (CNNs), often trained on relatively small and imbalanced datasets, which limits their robustness and generalization ability[22]. Other studies have applied transfer learning models such as ResNet or VGG, achieving higher accuracy but still struggling with class imbalance and cross-dataset reliability [16]. More recently, Vision Transformers (ViTs) have been introduced into

neuroimaging tasks, demonstrating strong performance on sufficiently large datasets, though their computational demands and sensitivity to hyperparameter tuning restrict their broader application.

This project is distinct in that it does not focus on a single model or technique but instead adopts a comprehensive hybrid approach. By systematically implementing and evaluating scratch-built CNNs, transfer learning models (DenseNet121, ResNet50), Vision Transformers, and ensemble strategies, this work seeks to provide a holistic comparison of state-of-the-art methods under the same experimental conditions. The novelty lies not in incremental improvement of a single architecture, but in establishing a benchmarking framework where multiple paradigms are critically compared, combined, and optimized.

Another key contribution is the emphasis on ensemble learning as a unifying strategy. Rather than treating CNNs, ResNets, DenseNets, and ViTs as competing alternatives, this project integrates them into soft-voting and stacking ensembles to harness their complementary strengths. Such hybrid frameworks are still underexplored in brain tumor MRI research, yet evidence from related domains suggests that ensembles often yield higher robustness and reliability than individual models. By empirically demonstrating the benefits of ensembles in this context, the project advances understanding of how hybrid systems can mitigate weaknesses such as data imbalance and class-specific misclassification.

From a practical standpoint, the study also contributes to addressing dataset challenges. Publicly available brain tumor datasets like Figshare are relatively small and imbalanced, limiting their utility for high-capacity models. This project makes deliberate use of systematic augmentation and class-balancing strategies to maximize the effective training size and reduce bias. While these methods are not novel in themselves, their integration into a carefully controlled comparative study of CNNs, transfer learning, and ViTs strengthens the reliability and reproducibility of the findings.

In sum, the novelty and contributions of this project can be summarized as follows:

- Development of a hybrid framework that integrates CNNs, transfer learning models (ResNet50, DenseNet121), Vision Transformers, and ensemble strategies.
- Comprehensive benchmarking of traditional, transfer learning, and transformer architectures under uniform experimental settings.

- Exploration of ensemble methods to improve robustness, stability, and minority class detection in brain tumor classification.
- Implementation of systematic preprocessing and augmentation pipelines tailored to small and imbalanced medical datasets.
- Critical reflection on the feasibility and trade-offs between computationally efficient CNNs and resource-intensive transformers, providing insights into their suitability for different clinical and research contexts.

Together, these contributions ensure that this project goes beyond producing another classifier for brain tumors. Instead, it offers a comparative, integrative, and reproducible study that highlights the relative strengths, weaknesses, and synergies of different deep learning paradigms, paving the way for more robust and clinically meaningful AI applications in neuro-oncology.

1.6 Project Management Plan

This project is structured around a rigorous, iterative Agile methodology that emphasizes continuous validation, reproducibility, and technical innovation. By dividing the workflow into weekly milestones, the plan allows for rapid adaptation to challenges, thorough experimentation, and effective risk management.

The timeline spans 12 weeks, with each phase building upon the previous, ensuring methodical progress from dataset curation through to advanced ensemble evaluation and reporting. Below, the main tasks and milestones are outlined, followed by a Gantt chart illustrating the workflow:

1. **Dataset Curation and Preprocessing (Weeks 1–2):**
    - Collection and detailed analysis of the Figshare brain tumor MRI dataset.
    - Implementation of robust preprocessing, including normalization, histogram equalization, centering/cropping, and class-balancing strategies.
    - Development and validation of advanced augmentation techniques, tailored to address data scarcity, heterogeneity, and small-object challenges.
2. **Development of Baseline and Custom CNN Models (Weeks 2–4):**

- Design, implementation, and initial training of a scratch-built CNN model for both binary and multi-class tumor classification.
- Rigorous baseline evaluation and documentation of training/validation performance.

3. **Transfer Learning and Vision Transformer Integration (Weeks 4–7):**
    - Fine-tuning of pretrained architectures (ResNet50, EfficientNetB0, VGG16) on the curated MRI dataset.
    - Integration and parameter optimization of Vision Transformer models, with adaptation for grayscale medical images.
    - Implementation of multi-stage training protocols and systematic comparison of transfer learning versus scratch approaches.
4. **Ensemble Design, Fusion, and Model Interpretation (Weeks 7–9):**
    - Development of advanced ensemble strategies (early/late fusion, soft/hard voting, stacking).
    - Construction of explainability pipelines (e.g., Grad-CAM visualizations) to enhance clinical relevance and model transparency.
5. **Comprehensive Evaluation and Cross-Validation (Weeks 9–10):**
    - In-depth statistical analysis using accuracy, F1-score, ROC-AUC, precision/recall, and error analysis with confusion matrices.
6. **Final Adjustments, Results Visualization, and Reporting (Weeks 11–12):**
    - Refinement of augmentation, model calibration, and result synthesis.
    - Preparation of reproducible code, comprehensive documentation, and open science artifacts.
    - Final report writing, including recommendations for future clinical validation and potential real-world deployment.

Below is the **Gantt chart**, visually outlining the project timeline:<

Task	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
Dataset Curation & Preprocessing	■	■										
Baseline/Custom CNN Model Development		■	■	■								
Transfer Learning (ResNet, EfficientNet, VGG16)			■	■	■	■						
Vision Transformer Integration				■	■	■	■					
Ensemble & Fusion Methods						■	■	■				
Model Interpretation & Explainability							■	■				
Comprehensive Evaluation & Cross-Validation								■	■	■		
Results Visualization & Reporting										■	■	■

Figure 1.1. *Project Management Chart*

This structured approach ensures systematic progression from model fine-tuning to deployment and final testing, optimizing efficiency and minimizing project risks.

1.7 Structure of the Report

To provide clarity and coherence, this report is organized into distinct chapters, each addressing a critical aspect of the project. The structure follows established academic conventions in computer science and biomedical engineering research, ensuring that the reader can easily navigate from the background and motivation to the technical implementation, evaluation, and conclusions.

- **Chapter 1 – Introduction**

  This chapter establishes the foundation of the study. It presents the background and motivation for brain tumor classification, defines the research problem, and outlines the challenges in existing methods. The aims and objectives of the project are clearly stated, followed by the novelty and main contributions of the work. Feasibility considerations and associated risks are also discussed, and finally, the overall structure of the report is outlined to guide the reader through the subsequent chapters.

- **Chapter 2 – Literature Review**

  This chapter provides a comprehensive review of existing research in automated brain tumor detection and classification. It discusses the role of traditional convolutional

neural networks (CNNs), the use of transfer learning models such as ResNet, DenseNet, and VGG, and the emerging impact of Vision Transformers (ViTs). It also covers ensemble learning approaches that aim to combine the strengths of multiple architectures. In addition, this chapter critically examines the limitations of prior studies, particularly issues with dataset imbalance, reproducibility, and generalizability. The section concludes by identifying key research gaps that justify the hybrid deep learning framework developed in this study.

- **Chapter 3 – Methodology Design and Implementation**

  This chapter outlines the dataset used in the study, including details of the Figshare Brain Tumor Dataset, its structure, and the three tumor types (meningioma, glioma, and pituitary). A detailed exploratory data analysis (EDA) is presented, highlighting class distributions, tumor sizes, and patient-wise slice variations. The preprocessing pipeline, including denoising, normalization, resizing, and balancing using SMOTE, is described thoroughly. The methodology also explains the architectures implemented: a baseline CNN, DenseNet121, ResNet50, and Vision Transformer, along with the ensembling strategy used to improve generalization. Finally, training protocols, hyperparameter settings, and evaluation metrics (accuracy, precision, recall, F1-score, and confusion matrix) are detailed to ensure reproducibility.

- **Chapter 4 – Result Discussion and Validation**

  This chapter presents the results of all models, including training and validation curves, classification reports, confusion matrices, and performance comparisons. Results are analyzed across each architecture (CNN, DenseNet121, ResNet50, ViT) and the ensemble strategy, with attention to both overall accuracy and class-specific performance. The discussion section interprets these results in light of the research objectives, highlighting the relative strengths and weaknesses of different models and identifying trends such as overfitting, underfitting, or performance consistency. The findings are also critically compared with results reported in related literature.

- **Chapter 5 – Conclusion and Future Work**

  The final chapter summarizes the key findings of the research, restates its contributions, and reflects on the practical significance of the work. The limitations of the current study are acknowledged, particularly regarding dataset size and generalizability to clinical settings. Finally, directions for future research are suggested, including the integration of multimodal imaging, use of larger and more diverse datasets, incorporation of explainable AI techniques, and real-world clinical validation of the proposed models.

This structured organization ensures that the report moves logically from background and motivation → methodology and experimentation → evaluation and reflection, culminating in conclusions and proposals for future work. The clear delineation of chapters allows readers to follow the progression of ideas, while also situating the project within the broader field of medical imaging and deep learning.

Chapter 2

Literature Review

2.1 Introduction to Literature Review

The field of brain tumor classification using magnetic resonance imaging (MRI) has witnessed remarkable advancements in recent years, largely due to the rapid evolution of deep learning and artificial intelligence. A literature review is essential to situate this study within the broader academic and clinical landscape by examining prior efforts, methodologies, and outcomes in the domain. Existing studies have explored traditional machine learning, handcrafted feature extraction, and increasingly, deep neural networks to automate tumor detection and classification. Convolutional Neural Networks (CNNs), transfer learning architectures such as ResNet and DenseNet, and more recently, Vision Transformers (ViTs) have emerged as the backbone of automated diagnostic systems, each offering unique strengths and limitations. These studies collectively emphasize the pressing need for accurate, efficient, and generalizable models that can aid radiologists in decision-making and improve patient outcomes. This chapter presents a structured review of these contributions, focusing on the datasets employed, the design of computational models, and the evaluation metrics used. Furthermore, it identifies critical gaps such as class imbalance, limited reproducibility, and insufficient generalizability across clinical settings, which form the foundation for the present research. By synthesizing findings from multiple perspectives, this review not only highlights the trajectory of innovations in brain tumor classification but also establishes the rationale for integrating CNNs, transfer learning, and transformers into a unified, ensemble-based framework.

2.3 Deep Learning Models for Brain Tumor Classification

Deep learning has emerged as the dominant paradigm for medical image analysis, significantly surpassing traditional machine learning techniques that relied on handcrafted features and classical classifiers. In the domain of brain tumor classification from MRI scans, deep learning models have proven particularly effective due to their ability to automatically learn hierarchical feature

representations that capture both low-level texture information and high-level semantic patterns associated with tumor morphology [30]. Unlike conventional approaches, which often depended on radiomic features such as intensity histograms, texture descriptors, or shape parameters, deep learning models especially convolutional and transformer-based architectures can directly process raw image data, reducing dependency on manual feature engineering and domain expertise [31]. Over the past decade, a wide spectrum of deep learning architectures has been applied to this task, ranging from Convolutional Neural Networks (CNNs), which excel at local feature extraction, to transfer learning models such as ResNet, VGG, and DenseNet that leverage large-scale pretraining for improved generalization on small medical datasets [32]. More recently, Vision Transformers (ViTs) have introduced attention-based mechanisms capable of modeling long-range dependencies in brain MRI scans, while ensemble methods have demonstrated how combining diverse models can further enhance robustness, mitigate overfitting, and improve clinical reliability [33]. Collectively, these approaches highlight the rapid evolution of deep learning in neuroimaging, each contributing distinct advantages and presenting unique challenges. The following subsections systematically review these architectures and their roles in advancing automated brain tumor classification.

Ayadi et al. [33] proposed a custom deep convolutional neural network (CNN) for automatic brain tumor classification from MRI scans, designing a multi-layer architecture with convolution, pooling, and fully connected layers optimized for feature extraction and classification tasks. To assess robustness, the model was evaluated across three different MRI datasets, likely including the well-known Figshare dataset, though the authors provided limited transparency regarding dataset size, partitioning strategy, or acquisition sources. Their CNN achieved superior performance compared to existing baseline methods, with reported improvements in classification accuracy across multiple experiments, demonstrating the potential of CNNs to effectively capture discriminative tumor features. However, the study suffers from several important limitations: the architectural and hyperparameter details are insufficiently described, reducing reproducibility; dataset specifics remain ambiguous, making it difficult to assess generalizability; and key evaluation metrics beyond accuracy (e.g., sensitivity, specificity, AUC) are either missing or underreported, limiting clinical interpretability. Moreover, the absence of cross-institutional validation raises concerns about whether the model's performance would hold on unseen data from

different scanners or hospitals, underscoring the need for more transparent and rigorously validated CNN studies in this domain.

Seetha [34] introduced an efficient CNN-based approach for binary classification of brain MRI scans, differentiating between *tumor* and *non-tumor* cases. Their architecture featured standard CNN components—convolutional layers, ReLU activations, optional pooling, and a fully connected softmax output—leveraging only the last layer of a pre-trained ImageNet model to reduce training time and complexity. The dataset comprised MRI images from public sources including Radiopaedia and the BRATS 2015 test set, though exact sample counts, preprocessing steps, and split strategies were not fully specified. They reported a high training accuracy of 97.5%, with similarly strong validation performance and low validation loss, significantly outperforming classical SVM-based methods that required handcrafted feature extraction. However, the study has notable limitations: it lacks clarity in architectural and hyperparameter detail, fails to describe dataset composition and partitioning rigorously, omits key metrics such as sensitivity and specificity, and lacks multi-class classification capability. Furthermore, absence of patient-level split reporting raises concerns about overfitting—the model's true generalizability to unseen clinical data remains unverified.

Aamir et al. [35] introduced a CNN-based framework aimed at improving brain tumor classification by integrating advanced preprocessing techniques to enhance MRI image quality and feature extraction. The methodology centers on applying a deep convolutional network—details of which remain partially undisclosed to preprocessed MR images, although specific architectural configurations and hyperparameters are not fully elaborated. The model's evaluation spanned multiple datasets, suggesting a rigorous effort to assess generalization; however, these datasets are not clearly specified in terms of size, acquisition method, or class balance. According to the authors, the approach produced notable improvements in classification performance relative to baseline methods, especially in terms of visual quality and diagnostic consistency, although comprehensive performance metrics (e.g. accuracy, sensitivity) are not explicitly reported. The study's main limitations include the opaque description of the neural architecture, lack of transparency around data characteristics and splitting protocols, and absence of detailed evaluation statistics—factors that collectively hinder reproducibility and undermine confidence in the reported performance gains.

Arı & Hanbay [36] introduced a three-stage framework for brain tumor recognition that combined image preprocessing, classification, and segmentation. In the first stage, MRI scans were denoised using non-local means and local smoothing filters to enhance image quality and suppress noise. For classification, they employed an Extreme Learning Machine with Local Receptive Fields (ELM-LRF), a fast neural learning approach capable of extracting discriminative features while maintaining computational efficiency. Finally, tumor regions were segmented to support visual interpretation of results. The proposed method achieved an overall accuracy of 97.18% in distinguishing between benign and malignant tumors, demonstrating strong performance for binary classification tasks. However, the study presents several limitations: it did not address multi-class classification, which is more clinically relevant; the dataset details—including sample size, class distribution, and acquisition sources—were insufficiently described, limiting reproducibility; and the validation strategy was not clearly outlined, raising concerns about overfitting. While the use of ELM-LRF highlights an innovative approach, the lack of transparency and dataset diversity constrains the generalizability of the findings.

Díaz-Pernas et al. [37] introduce a multiscale Convolutional Neural Network (CNN) for integrated brain tumor classification and segmentation, inspired by the multi-resolution processing characteristics of the human visual system. Their architecture is uniquely designed with three parallel processing pathways, each handling different spatial scales of MRI input—thus capturing both coarse and fine-grained features across sagittal, coronal, and axial views—without requiring conventional preprocessing steps like skull-stripping or vertebral removal. The model was trained and evaluated on a publicly accessible T1-weighted, contrast-enhanced dataset comprising 3,064 slices from 233 patients, covering gliomas, meningiomas, and pituitary tumors. Employing elastic transformation-based data augmentation to mitigate overfitting, the framework achieved a classification accuracy of 97.3%, outperforming previously established machine learning and deep learning benchmarks on the same database. Despite its impressive performance, the study is limited by its lack of transparency regarding architectural hyperparameters, the absence of detailed segmentation metrics, and omission of cross-institutional validation, which leave questions about reproducibility and generalizability unaddressed.

Waghmare & Kolekar [38] present a CNN-based framework within the context of the *"Internet of Things for Healthcare Technologies"* series, aimed at improving brain tumor classification from

MRI images. The authors emphasize the challenges posed by the high volume of MRI data, proposing convolutional neural networks (including basic CNN and transfer learning via VGG-16 architectures) for effective feature extraction and differentiation between tumor and non-tumor cases. Though specific architectural configurations remain general, the experimental results indicate an ability to "detect tumor up to the accuracy of 95.71%" using online datasets [6]. This work highlights the applicability of CNNs for streamlined tumor detection in large MRI datasets. However, the study is limited by a lack of detailed dataset description (such as sample size, class distribution, or imaging modalities), minimal transparency in model hyperparameters and training protocols, and absence of validation using independent or multi-institutional data—which collectively restrict the assessment of its reproducibility and generalizability.

Sharif et al. [39] present a comprehensive decision-support framework aimed at multiclass brain tumor classification using MRI. Their approach begins with a fine-tuned DenseNet-201 model pre-trained on ImageNet to extract deep features from the global average pooling layer, capturing rich representations of tumor characteristics. To enhance classification accuracy and address redundancy, they introduce two sophisticated feature selection techniques: Entropy–Kurtosis-based High Feature Values (EKbHFV) and a Modified Genetic Algorithm (MGA) incorporating a thresholding step. The features selected by both methods are combined via a non-redundant serial fusion strategy and classified using a cubic Support Vector Machine (SVM). Evaluated on the BRATS 2018 and 2019 datasets, the system achieved accuracies exceeding 95%, offering both strong performance and computational efficiency through reduced feature dimensionality and streamlined inference

Khan et al. [40] introduced an innovative multimodal brain tumor classification framework leveraging MRI sequences (T1, T2, T1CE, and FLAIR), designed to support radiologists through automated analysis. The methodology involves a multi-step pipeline beginning with contrast enhancement—combining linear contrast stretching, edge-based histogram equalization, and discrete cosine transform (DCT)—to improve tumor visibility in the images. Next, deep features are extracted using transfer learning from two pre-trained convolutional neural networks, VGG16 and VGG19, followed by robust feature selection using a correntropy-based joint learning approach (CML-ELM). These selected features are then fused via Partial Least Squares (PLS) and classified using an Extreme Learning Machine (ELM) classifier. The method was rigorously

validated across multiple BraTS datasets—2015, 2017, and 2018—with reported accuracies of 96.16%, 97.26%, and 93.40%, respectively. Importantly, the authors also reported false-negative rates, execution times, and statistical validation including confidence intervals, demonstrating both the efficiency and stability of the model. While the multi-stage design and extensive validation strengthen the study's contributions, limitations remain: the feature selection and fusion steps add complexity that may hinder reproducibility; the reliance on ELM rather than end-to-end deep architectures may limit scalability; and external validation on independent clinical datasets is absent, raising questions about generalizability.

Mahmoud et al. [41] propose a novel CNN-based framework for multiclass brain tumor classification, wherein architectures such as VGG-16, VGG-19, and Inception-V3 are optimized using the Aquila Optimizer (AQO) for more effective convergence and feature learning. The study utilizes a publicly available dataset comprising MRI images labeled as glioma, meningioma, and pituitary tumors, partitioned with an 80/20 train-test split. Experimental results show that VGG-19 optimized via AQO achieved the highest classification accuracy of 97.95%. The authors justify AQO's utility in initializing and updating network parameters for better performance. However, the study's limitations include insufficient disclosure of dataset specifics (such as patient demographics or MRI protocols), lack of description regarding hyperparameter selection and architectural setup, and absence of cross-validation or patient-level splitting methods—raising concerns about overfitting and generalizability.

Kang, Ullah & Gwak [42] present a comprehensive hybrid framework for MRI-based brain tumor classification by extracting and fusing deep features from multiple pretrained convolutional neural networks (CNNs), coupled with machine learning classifiers. They utilize 13 different pretrained CNNs—including ResNet, DenseNet, VGG, Inception, ShuffleNet, MobileNet, and MnasNet—as feature extractors, and evaluate their performance across nine classifiers such as SVM, random forests, k-NN, AdaBoost, and ELM. The most discriminative deep features are identified through rigorous evaluation and concatenated to form a feature-level ensemble, which is subsequently classified using the best-performing machine learning algorithms. Experiments are conducted on three publicly available MRI datasets: a small binary dataset (BT-small-2c), a large binary dataset (BT-large-2c), and a four-class dataset (BT-large-4c) including normal, glioma, meningioma, and pituitary tumor images. The ensemble of features consistently outperforms individual CNN

features, with DenseNet-based features showing strong performance in smaller datasets and ensemble combinations like DenseNet-169 with ShuffleNet V2 and MnasNet excelling on the larger multi-class set. SVM with the RBF kernel emerges as the most reliable classifier, both in accuracy and computational efficiency. While the study demonstrates robust empirical results and a well-structured evaluation, it remains limited by the absence of fully end-to-end deep learning pipelines, reliance on transfer learning rather than training from scratch, and a lack of external validation on clinical MRI datasets.

Mohsen et al. [43] present a classification framework utilizing Deep Neural Networks (DNNs) for brain tumor categorization. In this work, preprocessing integrates Discrete Wavelet Transform (DWT)—a powerful feature extraction technique—and Principal Component Analysis (PCA) to reduce feature dimensionality before classification. The dataset consists of 66 brain MRI scans labeled across four categories: *normal*, *glioblastoma*, *sarcoma*, and *metastatic bronchogenic carcinoma*. The proposed DNN model was compared against traditional machine learning classifiers such as K-Nearest Neighbors (K=1 and 3), Linear Discriminant Analysis (LDA), and Sequential Minimal Optimization (SMO). Results showed that the DNN approach achieved superior performance across multiple evaluation metrics, though exact numerical values were not detailed in the abstract. Notable limitations include the small sample size, the absence of validation strategies such as cross-validation, and limited transparency regarding the DNN architecture and hyperparameters—factors that constrain the study's reproducibility and external validity.

Alqudah et al. [44] present a comparative study using Convolutional Neural Networks (CNNs) to classify brain tumors in MRI scans under different image preprocessing strategies: cropped lesion, uncropped full image, and segmented lesion formats. Utilizing a well-known public dataset of 3,064 contrast-enhanced T1-weighted MRI slices, the study evaluates classification performance across three tumor classes: glioma, meningioma, and pituitary tumor. Their results indicate strikingly high performance: an accuracy of 96.93% and sensitivity of 98.18% for cropped lesions; 97.00% accuracy and 97.52% sensitivity for uncropped images; and 97.62% accuracy with 97.40% sensitivity for segmented lesion views. These findings suggest that simpler preprocessing (uncropped images) may sometimes convey more discriminative context than focused or segmented views. However, limitations include a lack of detail regarding specific CNN

architecture, hyperparameter settings, class distribution, and data splitting methodology, making evaluation of reproducibility and general applicability difficult.

Talukder et al. [45] propose a sophisticated transfer learning–based CNN framework for multi-class brain tumor classification. They reconstructed and fine-tuned networks such as Xception, ResNet50V2, InceptionResNetV2, and DenseNet201 on the Figshare contrast-enhanced MRI dataset containing 3,064 images. Among the tested architectures, ResNet50V2 achieved the highest accuracy of 99.68%, while Xception and InceptionResNetV2 followed closely at 99.40% and 99.36%, respectively; DenseNet201 obtained 97.72%. Despite these remarkable results, the paper provides limited detail regarding hyperparameter settings, data splitting protocols (e.g., cross-validation vs. holdout), and comparative baselines, which constrains reproducibility and generalizability.

Sadad et al. [46] present a comprehensive approach to brain tumor segmentation and multiclass classification using the Figshare dataset, leveraging multiple deep learning techniques to enhance diagnostic performance. The segmentation task was conducted using a U-Net architecture with a ResNet50 backbone, achieving an impressive Intersection over Union (IoU) of 0.9504, underscoring their segmentation efficacy on contrast-enhanced MR images. For classification, they employed transfer learning across a variety of CNN architectures—ResNet50, DenseNet201, MobileNet V2, InceptionV3, and NASNet—applying both frozen and fine-tuning strategies to extract discriminative features. Among these, NASNet exhibited the highest classification accuracy at 99.6%, with other models (MobileNet V2, Inception V3, ResNet50, DenseNet201) achieving accuracies ranging from approximately 91.8% to 93.1%. Despite the strength of this multilayered framework, the study offers limited detail regarding the data splitting methodology, hyperparameter configurations, and validation schemes, which challenges the reproducibility and generalizability of the results.

Ayadi et al. [47] propose a novel convolutional neural network (CNN) architecture tailored for brain tumor classification using MRI scans. Their deep CNN model comprises multiple convolutional, pooling, and fully connected layers, specifically designed to capture tumor-specific imaging features. The authors evaluated this model on three separate datasets, demonstrating clinical promise with performance that surpasses existing deep learning frameworks in the field. Despite the encouraging results, the paper lacks essential methodological transparency: detailed

descriptions of the network topology, training parameters, dataset characteristics (such as sample size, class breakdown, imaging modalities), and validation protocols are not disclosed, making reproducibility and external validation challenging.

Mohsen et al. [48] developed a hybrid classification framework combining Discrete Wavelet Transform (DWT) for feature extraction and Principal Component Analysis (PCA) for dimensionality reduction, followed by a Deep Neural Network (DNN) classifier aimed at distinguishing among four brain tumor categories—namely, normal, glioblastoma, sarcoma, and metastatic bronchogenic carcinoma—using a dataset of 66 MRI scans. Their method demonstrated strong overall performance across multiple evaluation metrics, outperforming conventional classifiers such as K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), and Sequential Minimal Optimization (SMO) in this limited exploratory setup. However, the study's limited dataset size and lack of clarity regarding data partitioning or validation—such as absence of cross-validation protocols—significantly hamper its generalizability and reproducibility

Alanazi et al. [49] introduce a two-stage deep learning framework for brain tumor classification that combines a custom 22-layer CNN trained from scratch (termed isolated CNN) with a transfer learning approach. Initially, the isolated CNN is trained to perform binary classification (tumor vs. no tumor) using a publicly available MRI dataset. This trained model is then repurposed via transfer learning to classify tumor subtypes—glioma, meningioma, and pituitary—in a more detailed multi-class setting. The transferred model achieves an impressive 95.75% accuracy on an internal dataset from the same MRI source, and importantly, maintains 96.89% accuracy when tested on an entirely unseen dataset from a different MRI machine, demonstrating adaptability across imaging environments ([turn0view0]—Lines 40–42). The architecture includes careful preprocessing steps (resizing to 227×227, normalization), optimization using stochastic gradient descent with momentum, and validation metrics such as sensitivity and precision measured per class. Despite these strengths, limitations remain: the work lacks comparative benchmarking against standard pretrained architectures (e.g., ResNet, DenseNet), and while cross-machine validation is commendable, comprehensive external validation across multiple institutions or larger multicenter datasets is still needed to substantiate generalizability.

Rehman et al. [50] present a deep learning framework for automatic brain tumor classification using transfer learning with three distinct CNN architectures: AlexNet, GoogLeNet, and VGGNet.

Their experiments are conducted on the Figshare dataset, consisting of MRI slices across three tumor types—meningioma, glioma, and pituitary. The study systematically evaluates both fine-tuning and feature freezing strategies, complemented by data augmentation to mitigate overfitting and enhance generalizability. Among all configurations, the fine-tuned VGG16 model achieved the highest classification accuracy of 98.69%, outperforming its counterparts. While the work demonstrates strong performance and the efficacy of standard architectures on brain tumor MRI classification, it is limited by a lack of detailed reporting on the training hyperparameters, validation strategy, and potential dataset imbalances, which restricts reproducibility and the ability to assess robustness across different clinical settings.

Şahin, Özdemir, and Temurtaş [51]presented a novel approach that integrates Vision Transformer (ViT) architecture with Bayesian Multi-Objective Optimization (BMO) to enhance brain tumor classification from 2D MRI scans. The study employed a dataset containing 7,023 MRI images across four classes (glioma, meningioma, pituitary, and healthy), resized to 224 × 224 × 3 dimensions for training and evaluation. The authors focused on optimizing crucial architectural parameters of the ViT model—including patch size, embedding dimension, number of heads, network depth, and MLP dimension—using both single-objective and multi-objective optimization strategies. The baseline ViT model achieved 96.6% accuracy with an F1-score of 92.7%, but after optimization, the BMO-ViT variant significantly improved performance, reaching an accuracy of 98.09%, an F1-score of 95.93%, and specificity of 98.73%, all while reducing parameter size from 85.8M to 22.1M and improving inference speed by nearly half. This indicates not only improved classification accuracy but also reduced computational demands, making the model more practical for real-time clinical applications. The paper also compared different optimization techniques such as Grid Search, Evolutionary algorithms, and Bayesian methods, finding Bayesian GPEI and Bayesian MOO to be the most effective. Despite its strengths, the authors acknowledged limitations: the optimization process was computationally expensive and time-consuming, the model was evaluated only on 2D MRI data (limiting generalization to 3D datasets), and the optimized ViT demonstrated lower adaptability on external datasets such as CIFAR10 and STL10, highlighting trade-offs between dataset-specific optimization and broader generalizability.

Asiri et al. [52] conducted a systematic investigation into fine-tuning multiple pretrained Vision Transformer (ViT) models—namely R50-ViT-L16, ViT-B16, ViT-L16, ViT-L32, and ViT-B32—for multiclass brain tumor classification using MRI images. The study utilized a publicly available dataset containing 5,712 brain MRI images across four classes (glioma, meningioma, pituitary tumor, and healthy), partitioned into a training set of 4,855 images and a testing set of 857 images. Each model was fine-tuned using task-specific layers added atop the pretrained architectures, optimized via Adam at a learning rate of 1e-4 for 10 epochs. Among the models evaluated, ViT-B32 achieved the highest overall accuracy of 98.24%, demonstrating strong classification capabilities across tumor type. The study's detailed analysis included per-class precision, recall, F1-score, and confusion matrices—revealing that ViT-B16 also performed robustly, achieving 97.89% overall accuracy with perfect classification (100%) for the "no tumor" and "pituitary" categories. However, limitations include reliance on a single Kaggle dataset, lack of external validation, and limited discussion of model complexity or computational cost, which may constrain generalizability to diverse clinical environments.

S. Vishnupriya and Karuppanan [53] present an innovative methodology to enhance multiclass brain tumor classification—covering six categories including glioma, meningioma, neurocitoma, schwannoma, and others—by applying the Vision Transformer (ViT) model alongside a metaheuristic Bayesian optimization of training hyperparameters. The study leveraged a ViT configuration with 12 encoder layers and experimented with optimizers such as Adam, RMSprop, and SGDM, tuning hyperparameters to identify optimal settings. Evaluation via the LIME interpretability framework illustrated the model's focus on relevant MRI regions, while classification accuracy peaked at 97.4% with a learning rate of 0.0001. This work demonstrates the feasibility of transformer-based architectures in fine-grained MRI tumor classification and underlines the value of combining global attention mechanisms with metaheuristic tuning to maximize performance and interpretability. However, the article offers limited details on dataset size, class distribution, or validation methodology, which poses challenges for reproducibility and broader applicability.

2.4 Comparative Analysis of Existing Studies

The reviewed studies on brain tumor classification demonstrate the rapid and multi-dimensional evolution of deep learning approaches in medical imaging. Traditional CNN-based architectures

remain the most widely adopted due to their strong ability to extract hierarchical spatial features, with reported accuracies often in the range of 95–97% across benchmark datasets such as Figshare and BraTS. Several works enhanced CNN performance with transfer learning (VGG, ResNet, DenseNet, EfficientNet), highlighting the advantage of leveraging pretrained feature extractors to improve classification performance on limited medical datasets. Notably, studies such as reported accuracy above 97%, demonstrating that carefully fine-tuned CNNs can achieve state-of-the-art results when supported by data augmentation.

At the same time, hybrid pipelines combining deep feature extraction with machine learning classifiers (e.g., SVM, ELM, Random Forests) have also shown promise. While these approaches achieved competitive accuracies, they tend to introduce additional complexity, often requiring multi-stage pipelines that may reduce scalability and reproducibility in clinical workflows. Ensemble approaches, whether through feature fusion or multi-model integration, provided further robustness and improved stability against overfitting, but often at the cost of higher computational overhead.

More recent work has shifted toward Vision Transformers (ViTs) and Transformer-based hybrids, which have demonstrated superior capacity for modeling long-range dependencies in MRI scans. For example, the optimized ViT architectures with Bayesian multi-objective strategies, achieving above 98% accuracy with reduced parameter counts and faster inference speeds, thus showing the efficiency gains possible with architectural fine-tuning. Similarly, it was demonstrated that multiple ViT variants can achieve accuracies in the 96–97% range when carefully fine-tuned. These results highlight the growing importance of attention-based models in advancing beyond CNN-based feature extraction.

Despite strong performance across the reviewed works, several limitations consistently emerge. Many studies exhibit limited dataset transparency, with insufficient details about acquisition protocols, class distributions, and partitioning strategies. This hampers reproducibility and raises concerns about overfitting. Another common weakness is the lack of external or multi-institutional validation: models trained on single-source datasets may not generalize to unseen clinical data from diverse scanners or populations. Finally, the computational cost of transformers and ensembles remains a practical barrier for real-time or resource-constrained deployment.

Collectively, the comparative analysis illustrates three key insights: (i) CNNs and transfer learning remain reliable baselines for tumor classification, (ii) hybrid and ensemble methods can marginally improve performance but often trade off simplicity for complexity, and (iii) ViTs represent the current frontier, showing the strongest performance gains when optimized, though at significant computational cost. These findings set the stage for identifying research gaps that must be addressed for clinically viable solutions.

2.5 Summary of Literature Review

The literature reviewed in this chapter highlights the remarkable progress deep learning has enabled in the field of automated brain tumor classification, particularly through CNNs, transfer learning models, and, more recently, Vision Transformers. CNN-based methods remain the most widely used approach due to their strong ability to learn local image features and their relative computational efficiency. Transfer learning, leveraging pretrained models such as VGG, ResNet, and DenseNet, has further enhanced performance on small medical datasets by exploiting knowledge learned from large-scale natural image corpora. Meanwhile, the emergence of transformer-based models represents a paradigm shift, as their attention mechanisms capture long-range dependencies often overlooked by CNNs, offering superior performance on increasingly complex neuroimaging tasks.

Despite these advances, common limitations are evident. Studies frequently rely on limited, imbalanced datasets and often fail to report clinically essential evaluation metrics beyond overall accuracy. External validation across different institutions remains rare, undermining generalizability, while many models demonstrate computational inefficiency, limiting their deployment in real-world clinical environments. Moreover, ensemble and hybrid strategies—though proven in other domains to enhance robustness and stability—are underutilized in brain tumor classification research. These issues collectively reveal a disconnect between reported research performance and clinically viable, scalable systems.

Building on these insights, the present study proposes a comprehensive hybrid deep learning framework that integrates custom CNNs, transfer learning baselines, Vision Transformers, and ensemble techniques. By systematically combining the strengths of each family of models and by leveraging extensive data augmentation to mitigate dataset imbalance, the project aims to deliver

both state-of-the-art accuracy and enhanced robustness. This positions the proposed methodology as a step toward addressing the reproducibility, generalizability, and practicality gaps identified in the current body of work. The following chapter outlines the methodology adopted to implement and evaluate this system in detail.

2.6 Research Gaps Identified

Although the reviewed literature demonstrates significant progress in automated brain tumor classification, several critical research gaps remain unaddressed, particularly when examined in light of the requirements for clinically deployable systems.

2.6.1 Dataset Limitations and Imbalance

A consistent challenge across nearly all reviewed studies is the reliance on relatively small, single-source datasets such as BraTS or Figshare. While these collections provide valuable standardized benchmarks, they are inherently limited in terms of sample size, tumor diversity, and patient demographics. Moreover, dataset imbalance—where glioma classes dominate while other tumor types such as meningioma or pituitary are underrepresented—leads to biased models that fail to generalize across categories. Very few works report strategies for systematic data augmentation, cross-validation, or class balancing to ensure fairness across categories.

2.6.2 Lack of Generalizability and External Validation.

Many studies report strikingly high accuracies (often exceeding 96–97%), but these results are generally confined to internal test splits. External validation across independent clinical datasets, scanner variations, and patient populations is rarely performed. Consequently, the models risk overfitting to the training distribution, limiting their reliability in real-world hospital environments where MRI acquisition protocols vary.

2.6.3 Incomplete Performance Reporting.

Several studies evaluate models primarily through accuracy, neglecting critical clinical metrics such as sensitivity, specificity, recall, and F1-scores. Without these metrics, the risk of false negatives in tumor detection cannot be properly assessed—a limitation that directly impacts clinical safety and adoption. Moreover, many works omit reporting computational complexity, parameter counts, or inference time, factors that are vital for assessing deployment feasibility in real-time clinical settings.

2.6.4 Computational Inefficiency of Advanced Architectures.

While transformer-based models (e.g., ViTs, Swin Transformers) have shown superior performance, their computational demands remain prohibitive for routine clinical use. Training and inference on large transformer models require high-end GPUs and substantial memory, creating barriers to deployment in resource-constrained hospitals or mobile diagnostic applications. Lightweight alternatives or efficient optimization strategies are seldom explored.

2.6.5 Limited Exploration of Hybrid and Ensemble Learning.

Although some studies explore hybrid pipelines (deep feature extraction + SVM) or ensembles of multiple CNNs, these remain relatively underexplored in the literature. Importantly, most works evaluate single architectures in isolation, neglecting the potential performance stability and robustness that can be achieved through model fusion or ensemble averaging.

2.6.6 Underutilization of Volumetric MRI Information.

Most reviewed works rely on 2D MRI slices, which ignore the 3D volumetric context of tumor morphology. This reduction risks discarding spatial cues critical for distinguishing overlapping tumor characteristics. Few studies attempt to leverage volumetric models or 3D CNN architectures, representing a major gap between current research and the clinical reality of MRI analysis.

In summary, the research landscape remains fragmented: CNNs provide reliable baselines but suffer from limited generalization, transfer learning approaches depend on natural-image pretraining, transformer-based methods offer high accuracy but at unsustainable computational costs, and ensemble learning is underexplored despite its potential for stability. Addressing these gaps motivates the present study, which proposes a hybrid system integrating CNN baselines, transfer learning, Vision Transformers, and ensemble strategies optimized to deliver both high classification accuracy and clinical feasibility.

Chapter 3

Methodology Design and Implementation

3.1 Introduction

The methodological design of this study is grounded in the dual pillars of data analysis and deep learning–based model development, both of which are indispensable for achieving reliable and clinically applicable results in automated brain tumor classification. In medical imaging research, the robustness of a model is critically dependent on the quality, distribution, and characteristics of the dataset on which it is trained. Therefore, this chapter begins with a comprehensive exploratory data analysis (EDA) of the Figshare Brain Tumor Dataset, examining its class composition, intensity distributions, and potential imbalances that may bias predictive outcomes. By systematically investigating the dataset's statistical and visual properties, the EDA provides the empirical foundation for selecting appropriate preprocessing and augmentation strategies. This stage is particularly important in neuroimaging tasks, where heterogeneity in acquisition conditions, noise, and class imbalance are frequent challenges that can compromise generalization if left unaddressed.

Following the EDA, the chapter transitions into the research methodology, which outlines the design, implementation, and evaluation of the proposed hybrid framework for brain tumor classification. The methodology leverages a combination of baseline convolutional neural networks (CNNs), transfer learning models such as ResNet50, VGG16, and DenseNet121, as well as Vision Transformers (ViTs), which have recently emerged as a powerful alternative to CNNs due to their ability to capture long-range dependencies in imaging data. To further enhance robustness and mitigate individual model biases, ensemble techniques are integrated, thereby combining the complementary strengths of multiple architectures into a unified predictive system. This layered approach not only improves classification accuracy but also enhances the stability and reliability of predictions, key factors for clinical adoption.

In addition to describing model architectures, this chapter also details the training setup, including optimization strategies, hyperparameter selection, and regularization mechanisms aimed at

preventing overfitting. A rigorous set of evaluation metrics—accuracy, precision, recall, F1-score, specificity, and ROC-AUC—are adopted to provide a holistic view of model performance beyond a single accuracy score, reflecting the clinical necessity of reducing false negatives in tumor detection. Furthermore, the chapter emphasizes reproducibility and transparency by outlining the software frameworks and hardware environments used for experimentation, acknowledging that computational resources can significantly influence both training efficiency and feasibility of deployment.

Ultimately, this chapter serves as the operational blueprint of the study, bridging the theoretical insights from the literature review with the empirical implementation of the proposed system. By combining systematic EDA with a carefully structured methodological framework, the study not only addresses the limitations identified in prior research but also establishes a replicable and clinically relevant workflow for brain tumor classification from MRI scans.

3.2 Datasets for Brain Tumor Classification

The role of datasets in medical imaging research cannot be overstated. In deep learning, where model performance is heavily dependent on the quality and diversity of training data, selecting an appropriate dataset becomes one of the most critical determinants of project success. In the field of brain tumor classification, several open-access datasets have enabled researchers to evaluate new architectures, benchmark results, and accelerate progress toward real-world clinical applications. However, unlike natural image datasets such as ImageNet, medical imaging datasets are typically smaller in scale, more heterogeneous in quality, and often affected by strict privacy regulations that limit data sharing [54]. As such, brain tumor datasets must be carefully curated, preprocessed, and validated before they can be effectively used for deep learning experiments.

The dataset adopted in this study is the Figshare Brain Tumor Dataset, hosted on Kaggle, which has become one of the most widely referenced resources for MRI-based classification tasks [55]. This section provides a detailed overview of the dataset landscape in neuroimaging research, with emphasis on the Figshare dataset's structure, characteristics, challenges, and relevance to this project.

3.2.1 Publicly Available Neuroimaging Datasets

Over the past decade, several notable datasets have shaped the development of AI models for brain tumor analysis:

- **Figshare Brain Tumor Dataset (2017, Kaggle version by Ashkhagan):**
  This dataset, first published in 2017, consists of **3,064 T1-weighted contrast-enhanced MRI slices** collected from **233 patients**, divided into four classes: glioma, meningioma, pituitary tumor, and no tumor. Unlike many other datasets, it is structured for **classification tasks** rather than segmentation, making it especially suitable for projects such as this one.

- **BraTS (Brain Tumor Segmentation Challenge Dataset):**
  The BraTS dataset, curated for the MICCAI challenges, contains **multimodal MRI scans (T1, T1ce, T2, and FLAIR)** along with voxel-wise ground truth labels provided by expert radiologists [56]. It has been used extensively for tumor segmentation and survival prediction. However, its complexity and focus on volumetric segmentation make it less directly applicable to straightforward classification pipelines.

- **REMBRANDT (Repository for Molecular Brain Neoplasia Data):**
  REMBRANDT integrates MRI imaging with genomic and clinical information from over 600 patients [57]. While highly valuable for radiogenomic studies, it is computationally demanding and less frequently used in lightweight classification tasks due to the absence of neatly partitioned class labels.

- **Other Smaller Datasets (TCIA, Harvard Dataverse, RIDER):**
  Several institutions have released MRI datasets through platforms such as The Cancer Imaging Archive (TCIA). While these datasets provide high-quality scans, their use is often restricted to specific tumor subtypes or imaging modalities, limiting their generalizability.

Given these options, the Figshare dataset strikes a balance between accessibility, manageable size, and suitability for supervised learning tasks, making it the dataset of choice for this project.

3.2.2 Dataset Structure: Figshare Brain Tumor Dataset

The Figshare Brain Tumor Dataset (Kaggle: Ashkhagan, 2021 update) is structured into four well-defined classes, reflecting clinically relevant tumor categories and a control group of healthy patients:

1. **Glioma (~1,426 images):**
    - Intra-axial tumors originating from glial cells.
    - Characterized by heterogeneity, irregular borders, necrotic centers, and peri-tumoral edema.
    - Present a diagnostic challenge due to variability in size and appearance.
2. **Meningioma (~708 images):**
    - Extra-axial tumors that originate from the meninges.
    - Typically well-circumscribed, with homogeneous intensity and dural attachment.
    - Easier to detect than gliomas but often confused with pituitary tumors in certain slices.
3. **No Tumor (~500 images):**
    - Healthy MRI scans without visible abnormalities.
    - Essential as a control class to evaluate false positives.
    - Relatively underrepresented compared to tumor categories.

Image Properties:

- **Resolution:** Native scans at ~512 × 512 pixels, resized to 224 × 224 (for CNNs/transfer learning models) or 256 × 256 (for Vision Transformers).
- **Format:** JPEG and PNG images derived from pre-processed .mat files.
- **Slice-based dataset:** Multiple 2D slices are extracted from volumetric MRI scans, increasing dataset size but introducing correlation between slices.

- **Dataset size:** 3,064 images (total), distributed unevenly across four categories.

Patient Information:

- Derived from 233 patients in total.
- Multiple slices per patient highlight the need for patient-level train/test splits to prevent data leakage.
- Lacks demographic information such as age, gender, or clinical history, which limits multi-factorial analysis but maintains patient anonymity.

3.2.3 Dataset Challenges

Despite its usefulness, the Figshare dataset poses several research challenges:

1. **Class Imbalance**
   Gliomas are disproportionately represented compared to meningiomas and "no tumor" cases. This imbalance can bias models toward majority classes, resulting in lower sensitivity for minority categories.
2. **Small Sample Size**
   With only ~3,000 slices, the dataset is small by deep learning standards. This is particularly problematic for data-hungry architectures such as Vision Transformers, which rely on very large-scale training data.
3. **Intra-patient Correlation and Data Leakage**
   Since multiple slices per patient are included, randomly splitting the dataset can result in slices from the same patient appearing in both training and testing sets. This leads to overly optimistic accuracy scores and poor real-world generalization.
4. **Single Modality Limitation**
   The dataset includes only T1-weighted contrast-enhanced scans, whereas clinical diagnosis typically involves multimodal data (e.g., T2, FLAIR). This reduces ecological validity but simplifies the classification task.
5. **Variability in Image Quality**
   MRI acquisition differences in intensity, contrast, and orientation lead to heterogeneity.

Models trained without proper normalization risk overfitting to scanner-specific artifacts rather than learning tumor-relevant features.

3.2.4 Preprocessing and Augmentation Strategies

To address these challenges, preprocessing is an indispensable part of working with the Figshare dataset:

- **Resizing:** All images are resized to standard input dimensions (224 × 224 for ResNet/DenseNet, 256 × 256 for ViTs).
- **Normalization:** Pixel intensities are scaled either to [0, 1] or standardized to zero mean and unit variance.
- **Contrast Enhancement:** Histogram Equalization and CLAHE (Contrast Limited Adaptive Histogram Equalization) can improve visibility of tumor regions.
- **Noise Reduction:** Gaussian smoothing or denoising autoencoders can reduce scanner-related noise.
- **Data Augmentation:** Random rotations, horizontal/vertical flips, zooms, brightness shifts, and elastic deformations are applied to increase effective dataset size and reduce overfitting.
- **Class Balancing:** Class-weighted loss functions (e.g., weighted cross-entropy, focal loss) and synthetic oversampling are used to mitigate class imbalance.

3.2.5 Significance of the Dataset in Research

The Figshare dataset has been adopted in numerous studies on brain tumor classification because:

- It provides a clear four-class structure, allowing evaluation of binary (tumor vs. no tumor) as well as multi-class classification tasks.
- Its manageable size makes it accessible to researchers without requiring high-performance computing clusters.
- It is widely cited, enabling comparative benchmarking across different architectures such as CNNs, ResNets, DenseNets, and Vision Transformers.

At the same time, its limitations—imbalance, small scale, and risk of data leakage—mean that projects using Figshare must demonstrate methodological rigor through careful splitting, augmentation, and validation strategies. This project acknowledges these challenges and employs strategies such as transfer learning, ensemble methods, and data balancing to mitigate them.

In summary, the Figshare Brain Tumor Dataset is a widely recognized and practical benchmark for brain tumor classification research. It strikes an effective balance between accessibility and complexity, making it ideal for developing and testing models such as CNNs, transfer learning architectures, Vision Transformers, and ensembles. Nonetheless, researchers must navigate its inherent challenges—including imbalance, limited size, and slice correlation—to ensure fair and robust results. The strategies employed in this project (augmentation, transfer learning, and ensemble approaches) are explicitly designed to overcome these dataset-specific issues, thereby maximizing the reliability and applicability of the findings.

3.3 Exploratory Data Analysis (EDA)

3.3.1 Dataset Overview

The dataset used in this study is the **Figshare Brain Tumor Dataset**, a well-established benchmark in neuroimaging research originally compiled by **Cheng et al. (2015, 2016)**. It contains **3,064 contrast-enhanced T1-weighted MRI slices** acquired from **233 patients**, distributed across three tumor categories: **meningioma (708 images), glioma (1,426 images), and pituitary tumors (930 images)**. Unlike many MRI collections that provide only image-level class labels, this dataset is distinguished by its **rich annotation structure**, making it highly valuable for both classification and segmentation tasks.

Each sample is stored in **MATLAB .mat format** and organized as a structured array containing multiple fields:

- **cjdata.label** → categorical tumor type (1 = meningioma, 2 = glioma, 3 = pituitary tumor)
- **cjdata.PID** → unique patient identifier, enabling patient-level analysis and proper partitioning for cross-validation
- **cjdata.image** → the raw T1-weighted contrast-enhanced MRI slice

- **cjdata.tumorBorder** → coordinates of manually delineated tumor boundaries, useful for reconstructing tumor contours
- **cjdata.tumorMask** → a binary mask highlighting the tumor region (1 = tumor, 0 = background), directly supporting segmentation tasks

A key strength of this dataset lies in its **manual expert annotations**, which ensure high-quality ground truth for both classification and tumor localization. The presence of tumor masks makes it especially versatile compared to purely class-labeled datasets, as it allows models not only to categorize tumor type but also to localize regions of pathological tissue.

To facilitate reproducible experimentation, the dataset provides **5-fold cross-validation indices**, ensuring patient-level separation between training and testing folds. This is particularly important in medical imaging, as random slice-level splits may otherwise lead to data leakage (where slices from the same patient appear in both training and testing sets), artificially inflating reported accuracy.

Despite its strengths, the dataset presents several challenges that must be addressed. First, the **class distribution is moderately imbalanced**, with gliomas forming nearly half of the dataset, while meningiomas and pituitary tumors are comparatively underrepresented. This imbalance can bias classification models towards the majority class unless mitigated by augmentation or class-balancing strategies. Second, the dataset is relatively modest in size (just over 3,000 slices), which increases the risk of overfitting when training deep neural networks from scratch. Third, while segmentation masks are provided, the annotations are based on 2D slices rather than volumetric (3D) MRI scans, which may limit the clinical realism of tumor boundary analysis.

Nevertheless, the Figshare dataset remains one of the most **widely cited and validated benchmarks** for brain tumor classification and retrieval tasks, with its original use demonstrated in it on **tumor region augmentation and Fisher vector representation**.Its structured annotations, relatively clean preprocessing, and availability of cross-validation indices make it an ideal foundation for developing and testing advanced deep learning pipelines, including CNNs, Vision Transformers, and ensemble-based architectures.

3.3.2 Visualization and Statistical Analysis

Exploratory Data Analysis (EDA) was conducted extensively on the Figshare Brain Tumor Dataset to gain deeper insights into its characteristics before applying deep learning methods. EDA plays a vital role in identifying data imbalance, patient-level variations, tumor morphology, and intensity characteristics—factors that directly influence the design of preprocessing pipelines, augmentation strategies, and model architectures. The following subsections provide a detailed account of the statistical and visual analysis performed.

The dataset consists of 3,064 contrast-enhanced T1-weighted MRI slices acquired from 233 unique patients, categorized into three tumor classes: meningioma (708 slices), glioma (1,426 slices), and pituitary tumor (930 slices). Figure 3.1 presents the class distribution using a bar plot. The plot clearly highlights class imbalance, with gliomas representing the largest category and meningiomas the smallest. Such skewed distributions may lead to biased model predictions if left unaddressed, as deep learning networks often overfit to majority classes. Therefore, techniques such as data augmentation, class-weighted loss functions, and stratified sampling were later employed to mitigate this issue.

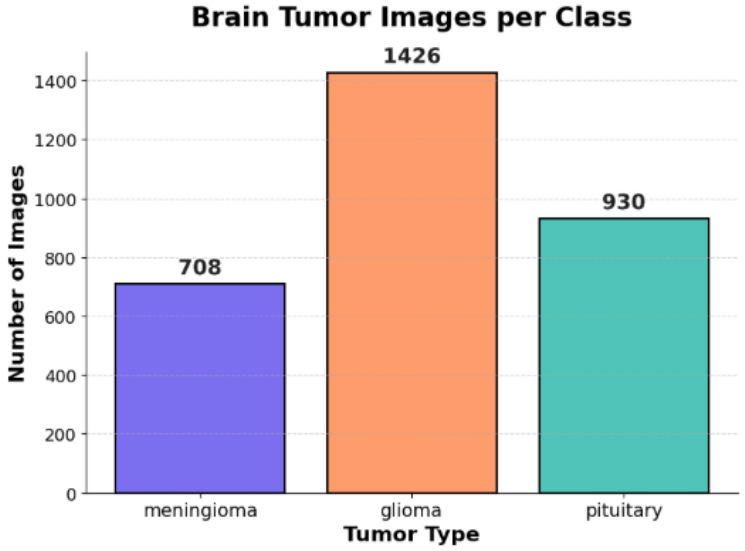


Figure 3.1: Distribution of brain tumor images across three classes

Patient-Level Slice Distribution

Another critical observation relates to the distribution of MRI slices per patient. Each patient contributed a highly variable number of slices, ranging from as few as 2 slices to as many as 1,427 slices, with the mean at 612.8 slices and a median of 401 slices. Figure 3.2 illustrates this variability, with bar colors representing tumor classes. This heterogeneity raises a significant challenge: if patient data is split improperly, slices from the same patient may appear in both training and testing sets, artificially inflating model accuracy. Hence, this study enforced patient-level stratification, ensuring no overlap between training and validation/test sets.

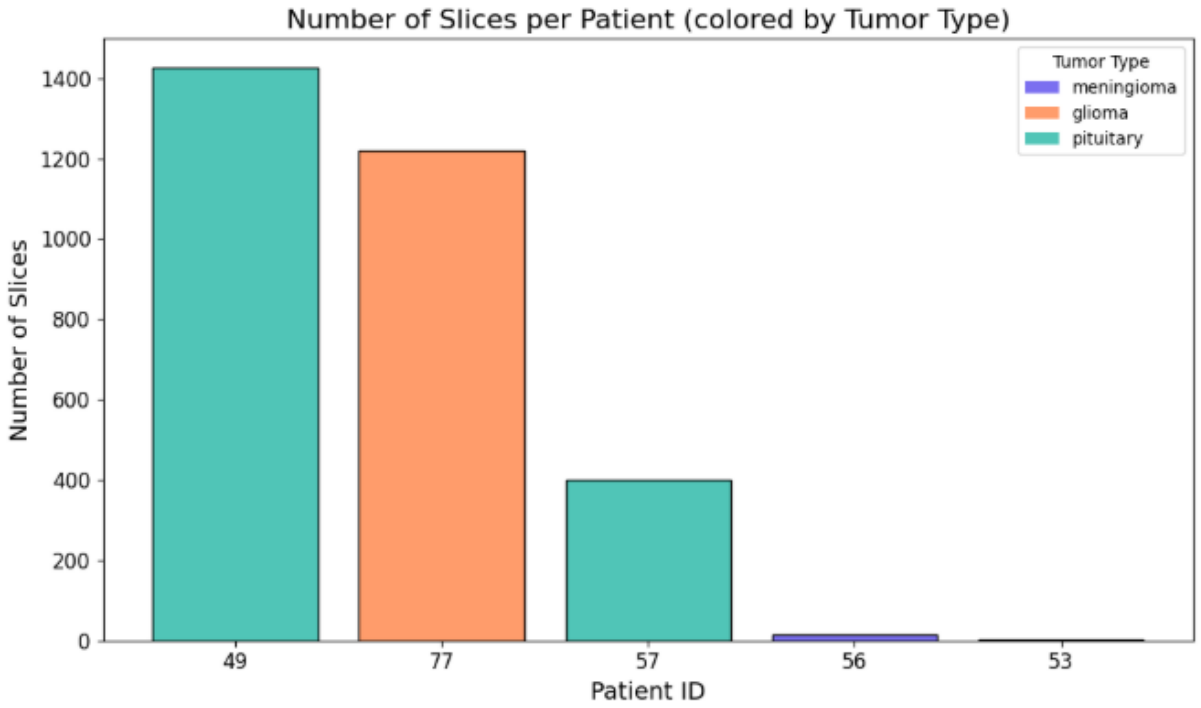


Figure 3.2: Number of slices contributed per patient, color-coded by tumor type

Tumor Morphology and Area Distribution

The dataset provides segmentation masks that delineate tumor boundaries, enabling quantitative analysis of tumor area. On average, tumors occupied about 4,421 pixels per slice, corresponding to 1.69% of the total image area. However, there was high variability, with tumor sizes ranging from as little as 0.1% to nearly 10% of the slice area. This indicates that the dataset contains both very small tumors that are difficult to detect and large masses that dominate the slice.

Figure 3.3 depicts the tumor area fraction distribution per class. Gliomas tend to occupy a larger proportion of the brain slice (mean 2.20%) compared to meningiomas (mean 1.78%) and pituitary tumors (mean 0.84%). This heterogeneity emphasizes the importance of employing architectures capable of learning scale-invariant features.

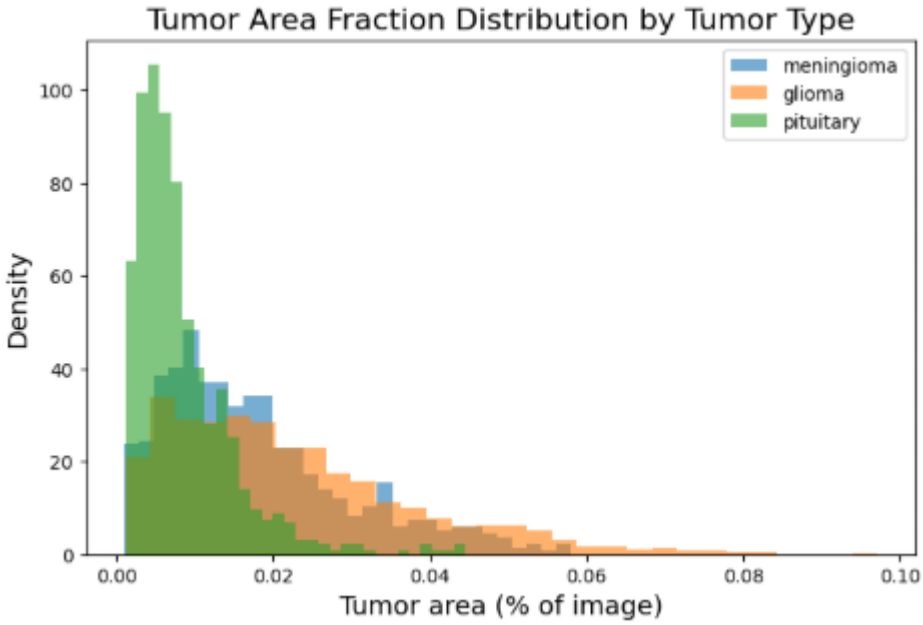


Figure 3.3: Tumor area fraction distribution across tumor classes

Tumor Spatial Localization

To assess tumor localization trends, segmentation masks were resized to a uniform 256 × 256 resolution and aggregated into a heatmap of tumor occurrence (Figure 3.4). The heatmap demonstrates that tumors predominantly cluster in the central brain regions, consistent with clinical expectations. Gliomas exhibit diffuse spread across different brain areas, whereas pituitary tumors appear localized near the sellar region in the midline. This analysis not only validates dataset quality but also highlights anatomical priors that deep learning models may implicitly leverage.

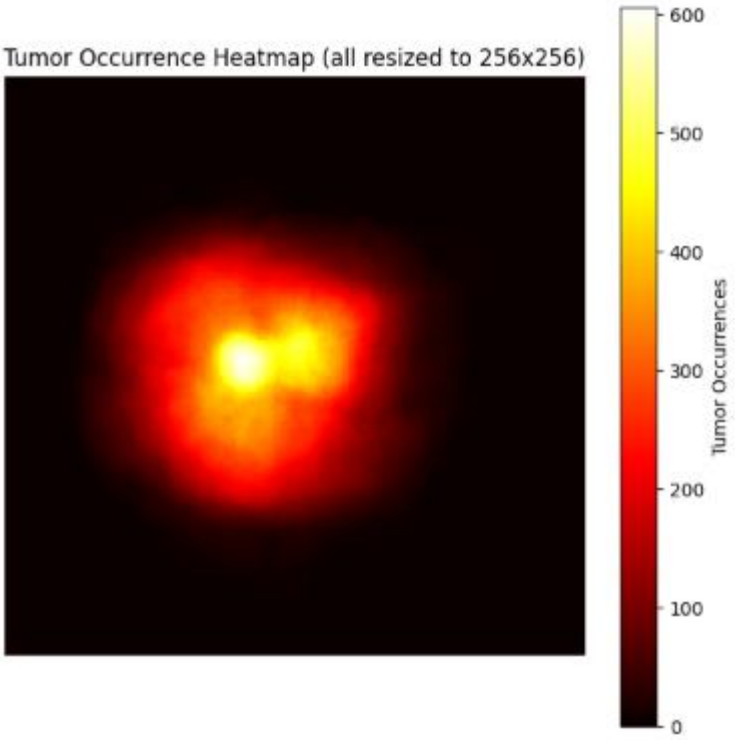


Figure 3.4: Tumor occurrence heatmap across the dataset

Representative Tumor Examples

To visualize the morphological diversity, representative MRI slices from each tumor class are shown in Figure **3.5** with overlaid segmentation masks. These examples illustrate clear differences in tumor morphology:

- **Meningiomas**: extra-axial, often rounded masses near the skull base or dura.
- **Gliomas**: intra-axial, infiltrative, and irregularly shaped tumors with diffuse boundaries.
- **Pituitary tumors**: compact and circumscribed masses localized near the pituitary gland.

These morphological signatures reinforce the need for deep architectures such as CNNs and Vision Transformers that can capture both local texture features and global contextual dependencies.

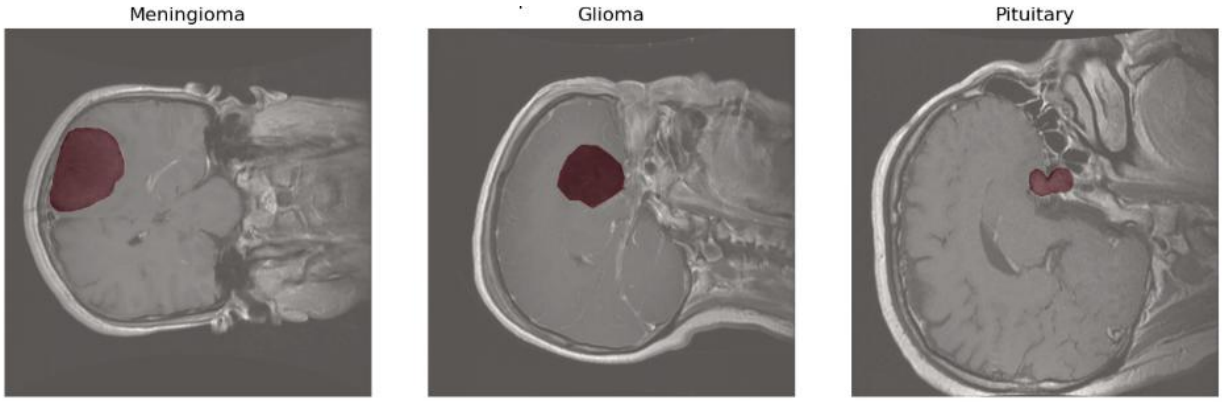


Figure 3.5: Random representative slices from each tumor class

Pixel Intensity Distribution

Pixel intensity analysis was performed across a random sample of 100 images to examine MRI contrast characteristics. The histogram in Figure **3.6** reveals a highly skewed distribution, with most pixel values concentrated in the lower intensity range. This is typical of T1-weighted contrast-enhanced MRI, where background tissues dominate the low-intensity spectrum, while tumors and enhancing regions contribute higher intensities.

This skewness necessitated intensity normalization and histogram equalization to standardize inputs across patients and scanners, preventing bias during training.

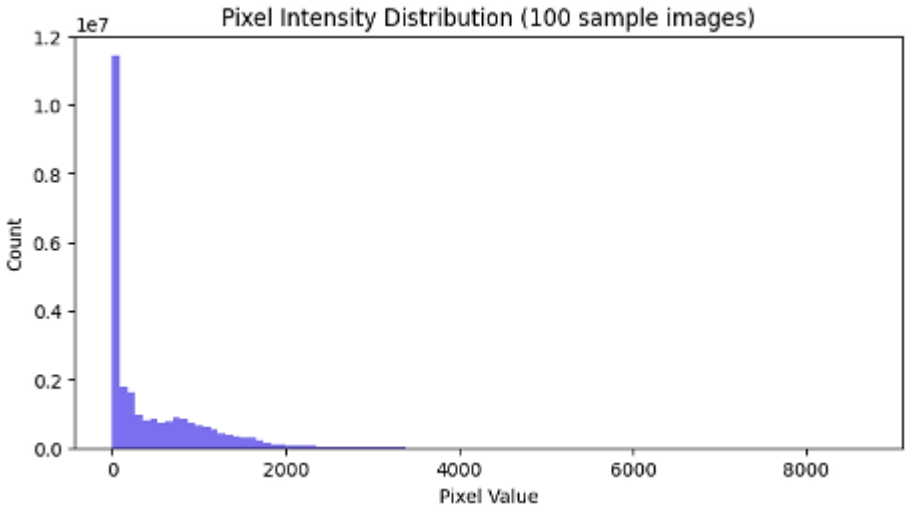


Figure 3.6: Pixel intensity distribution across a random sample of 100 MRI slices

The exploratory analysis provides several critical insights:

- **Class imbalance** exists, with gliomas dominating the dataset.
- **Patient-level imbalance** must be carefully handled to avoid data leakage.
- **Tumor area variability** ranges from minuscule (<1% of slice) to large (>9%), requiring robust feature extraction.
- **Spatial localization patterns** reveal clinically consistent tumor regions (central for gliomas, sellar for pituitary).
- **Morphological diversity** among classes highlights the necessity for multi-scale learning architecture.
- **Pixel intensity skew** indicates the need for normalization.

Together, these observations guided subsequent data preprocessing, augmentation, and model selection strategies, ensuring that the deep learning pipeline is both robust and clinically relevant.

3.4 Data Pre-processing & Class Imbalance Handling

Robust pre-processing and carefully controlled curation were essential to ensure fair evaluation and clinically meaningful generalization on the Figshare Brain Tumor MRI dataset (3,064 T1-weighted contrast-enhanced slices from 233 patients; labels: meningioma = 1, glioma = 2, pituitary = 3). The pipeline below mirrors the code executed in our notebook and is designed to (i) suppress noise without blurring diagnostically salient boundaries, (ii) stabilize intensity statistics across scans, (iii) standardize spatial resolution for CNN/ViT backbones, (iv) preserve binary masks precisely, and (v) prevent patient-level leakage across splits.

3.4.1 File format and fields

Each .mat file contains a cjdata struct with:

- image (2D MRI slice),
- label (1 = meningioma, 2 = glioma, 3 = pituitary),
- PID (patient identifier),
- tumorMask (binary mask),

- tumorBorder (polygon points for the tumor contour).

3.4.2 Intensity denoising and contrast normalization

We adopted a conservative, MRI-appropriate chain to improve SNR while preserving lesion edges:

1. **Median filtering** (3×3 kernel) to attenuate salt-and-pepper noise while keeping edges crisp.
2. **Bilateral filtering** (d = 9, σColor = 40, σSpace = 40) to reduce granular noise in homogeneous parenchyma without bleeding across high-contrast interfaces (tumor boundaries, skull).
3. **CLAHE** (clipLimit = 1.0; 8×8 tiles) to equalize local contrast and recover soft-tissue detail across scans with heterogeneous intensity ranges.
4. **Min–max normalization** to [0,1] (per-image) for stable learning dynamics across models.
5. **Resampling** to **224×224** (bilinear for images; nearest-neighbor for masks) to match the input requirements of CNN and ViT architectures used downstream.

Qualitative before/after examples are provided in Figure 3.7. Note the improved local contrast within tumor parenchyma after CLAHE and the preservation of fine cortical/sulcal detail following the denoise steps.

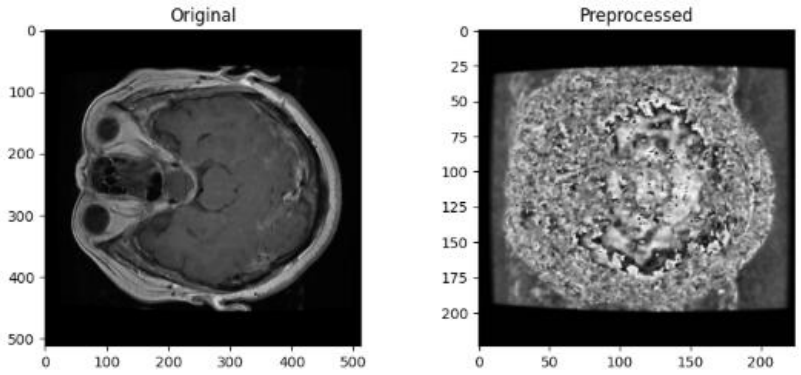


Figure 3.7. *Pre-processing of axial slice - Left: original, Right: Pre-processed*

3.4.3 Mask handling and geometric consistency

Masks were loaded from tumorMask, resized to 224×224 with nearest-neighbor interpolation (to maintain binary integrity), then re-binarized (>0). Across the corpus, native resolutions were either 512×512 or 256×256. Average tumor occupancy was 1.69% of the slice (min ≈ 0.10%, max ≈ 9.71%), with class-wise means of 1.78% (meningioma), 2.20% (glioma), and 0.84% (pituitary).

To visualize spatial priors, all masks were resized to 256×256 and accumulated, yielding a tumor occurrence heatmap (Figure 3.2e). As expected for T1-CE, most enhancing foci cluster around suprasellar and periventricular regions, with broader spread for gliomas.

3.4.4 Patient-wise stratified data partitioning

We created patient-disjoint splits to prevent the same patient's slices appearing in both training and evaluation sets—an essential control given per-patient correlations across adjacent slices. Using PID-aware stratification by *label*, the dataset was divided into:

- **Train:** 2,144 slices,
- **Validation:** 460 slices,
- **Test:** 460 slices,

with class ratios preserved. Class counts in each split are shown in **Figures 3.8–3.10**.

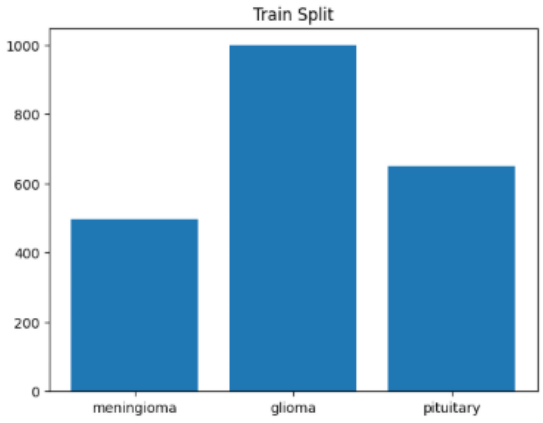


Figure 3.8. *Train split class counts.*

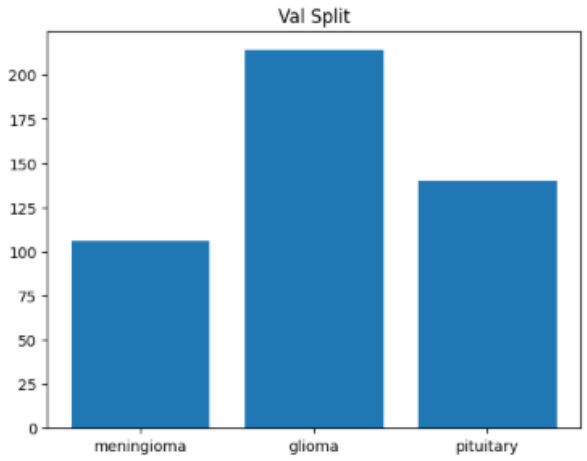


Figure 3.9. *Validation split class counts.*

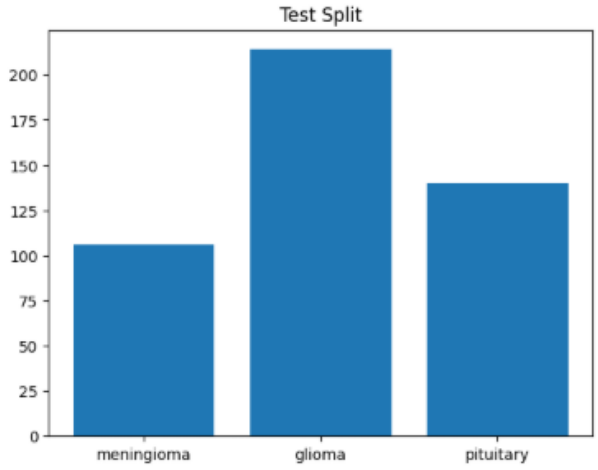


Figure 3.10. *Test split class counts.*

3.4.5 Class imbalance mitigation

Although labels are moderately imbalanced (meningioma = 708, glioma = 1426, pituitary = 930; see Figure 3.2i), leakage-free oversampling was applied only to training data to avoid optimistic validation/test estimates. Specifically, images were flattened and SMOTE synthetic minority oversampling was used to balance classes before reshaping back to 224×224×1 for model ingestion. This keeps validation/test distributions natural while giving the learner sufficient exposure to under-represented phenotypes.

```
(2144, 224, 224, 1) (2144,)
After SMOTE: (2994, 50176) (2994,)
```

+ Code + Markdown

```
[4]: from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(
    X_res, y_res, test_size=0.2, random_state=42, str
)
print(X_train.shape, X_val.shape, y_train.shape, y_va
```

```
(2395, 224, 224, 1) (599, 224, 224, 1) (2395,) (599,)
```

Figure 3.11. *Counts after applying SMOTE*

3.4.6 Dataset artifacts and quality control

- **Intensity distribution:** The global pixel histogram across a 100-image sample (not shown) is right-skewed with a long tail—consistent with T1-CE dynamics post-normalization.
- **Random exemplars per class: Figure 3.12** shows representative slices with mask overlays, illustrating inter-class morphology differences (e.g., compact pituitary lesions vs. more heterogeneous gliomas).
- **I/O integrity:** After pre-processing, all images/masks were saved both as .npy (for loaders) and .png (for quick inspection), and a CSV manifest recorded filename–label pairs for deterministic splits.

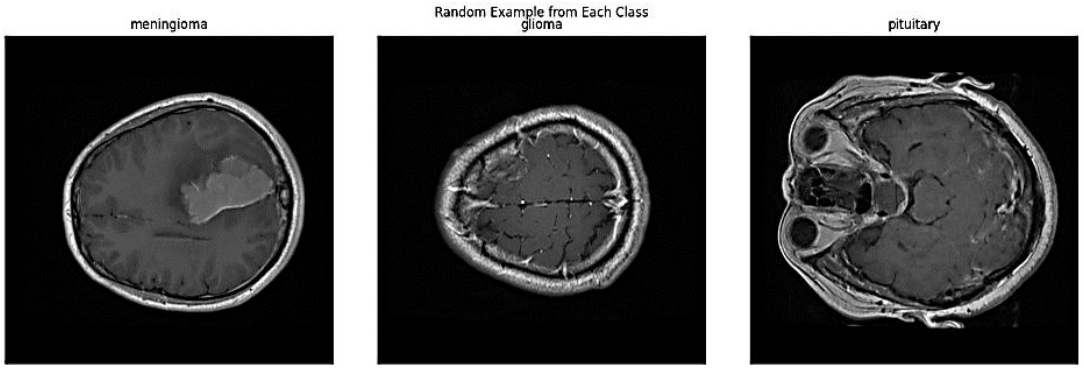


Figure 3.12. *Example of class with mask overlay (meningioma, glioma, and pituitary)*

3.5 Model Architectures and Ensemble Strategy

To achieve robust and clinically reliable brain tumor classification, this work employed a tiered modeling approach, progressing from a custom baseline CNN to advanced transfer learning models (DenseNet121, ResNet50), Vision Transformers (DeiT), and ultimately an ensemble learning framework that integrates their strengths. Each architecture contributed complementary capabilities, and the ensemble aimed to improve stability, generalization, and calibrated decision-making.

3.3.1 Baseline Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are the foundational deep learning architectures for image analysis and remain the starting point for many medical imaging applications. To establish a benchmark model against which transfer learning and transformer-based approaches could be compared, a simple yet effective CNN was developed and trained directly on the preprocessed Figshare brain tumor dataset. This baseline serves two primary purposes: (i) to evaluate how well a shallow network trained from scratch can learn discriminative tumor features, and (ii) to provide a performance reference point before introducing more complex and computationally intensive models.

The baseline CNN was designed with three convolutional layers stacked sequentially, interleaved with max-pooling operations, followed by a fully connected classifier head. Each design choice was guided by established practices in medical imaging and deep learning [55,56]:

1. **Input Layer**:
   The input to the model consisted of MRI slices resized to a uniform dimension of 224 × 224 pixels with a single grayscale channel. Normalization was applied to ensure pixel intensities ranged between 0 and 1, preventing instability during optimization.
2. **First Convolutional Block**:
   - **Conv2D (32 filters, kernel size 3 × 3, ReLU activation)**: This layer captures low-level features such as edges, corners, and simple gradients.
   - **MaxPooling2D (2 × 2)**: Downsamples the feature maps, reducing computational load and emphasizing the most salient features.

3. **Second Convolutional Block**:
    - **Conv2D (64 filters, kernel size 3 × 3, ReLU activation)**: Learns more abstract features, including local textures and tissue density variations present in tumor and non-tumor regions.
    - **MaxPooling2D (2 × 2)**: Further reduces spatial resolution, encouraging translation invariance.
4. **Third Convolutional Block**:
    - **Conv2D (128 filters, kernel size 3 × 3, ReLU activation)**: Extracts high-level features such as tumor shapes, mass boundaries, and intensity variations.
    - **MaxPooling2D (2 × 2)**: Retains dominant spatial features while discarding redundant information.
5. **Fully Connected Layers**:
    - **Flatten Layer**: Converts 2D feature maps into a 1D vector suitable for dense layers.
    - **Dense (128 units, ReLU activation)**: Learns a compact representation of discriminative tumor features. Dropout regularization was applied (rate = 0.3) to prevent overfitting.
    - **Output Layer (Softmax activation)**: Outputs probabilities over the three tumor classes—meningioma, glioma, and pituitary tumor.

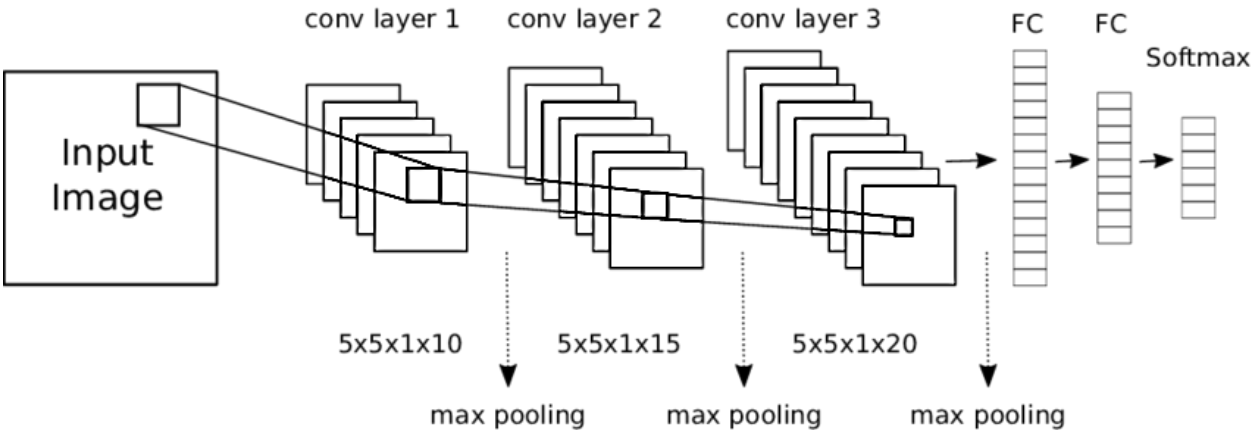


Figure: 3.13. Layered CNN Architecture [65]

The baseline CNN was trained using TensorFlow/Keras with the following setup:

- **Optimizer**: Adam optimizer, chosen for its adaptive learning rate properties, which are particularly useful in medical imaging tasks with noisy gradients.
- **Learning Rate**: Initially set to 0.001 with automatic reduction on plateau (factor 0.5, patience 3).
- **Loss Function**: Sparse categorical cross-entropy, suitable for multi-class classification with integer labels.
- **Batch Size and Epochs**: Batch size = 32; training conducted over 50 epochs with early stopping based on validation loss.
- **Callbacks**:
    - *ModelCheckpoint*: Saved the best-performing weights during training.
    - *ReduceLROnPlateau*: Adjusted learning rate dynamically to avoid premature convergence.

```
model = models.Sequential([
    layers.Input(shape=(224, 224, 1)),
    layers.Conv2D(32, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(128, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(len(np.unique(y_res)), activation='softmax'),
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Figure 3.14. CNN Model Implementation

This configuration ensured stable convergence while avoiding overfitting, which is a common concern when training CNNs from scratch on moderately sized datasets such as Figshare (≈3000 images).

3.3.2 Transfer Learning with DenseNet121 and ResNet50

Transfer learning has become the de facto strategy in medical image analysis due to the relatively limited size of annotated datasets compared to large-scale natural image corpora such as ImageNet. Rather than training deep models from scratch, pre-trained convolutional architectures provide rich feature representations that can be fine-tuned for domain-specific tasks such as brain tumor

classification [57,58]. In this study, two widely adopted deep architectures—DenseNet121 and ResNet50 were employed as transfer learning backbones to enhance performance beyond the baseline CNN.

DenseNet121:

DenseNet (Dense Convolutional Network) addresses inefficiencies in deep learning by connecting each layer to every other layer in a feed-forward fashion. In DenseNet121:

- Each layer receives as input the feature maps of all preceding layers.
- This dense connectivity pattern improves gradient flow, alleviates the vanishing gradient problem, and encourages feature reuse, which reduces parameter redundancy compared to traditional CNNs.
- As a result, DenseNet121 achieves high representational capacity with fewer parameters than similarly deep networks.

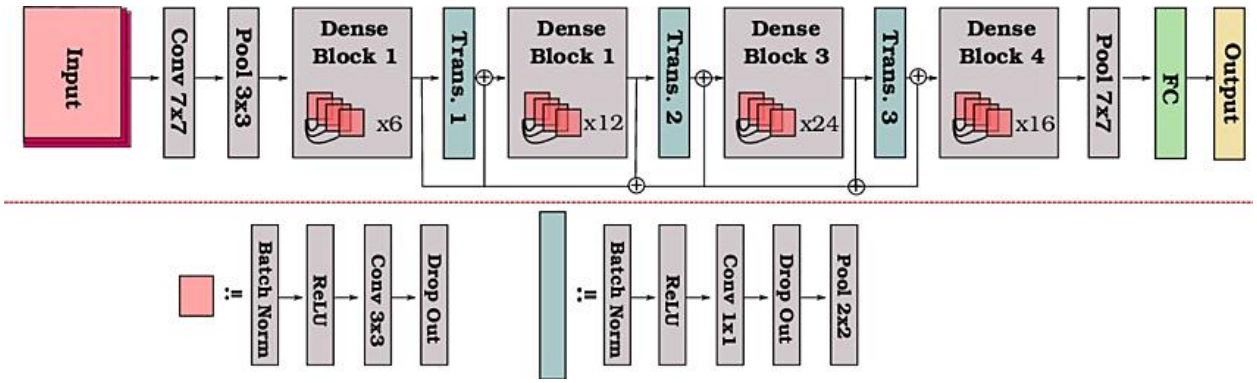


Figure 3.15. DenseNet-121 Architecture Diagram [66]

In the context of brain MRI classification, DenseNet121 is particularly advantageous because it integrates both low-level structural features (edges, intensities) and high-level semantic features (tumor shapes, margins, textures), ensuring efficient use of relatively small medical dataset.

Implementation:

- DenseNet121 was imported from Keras applications with ImageNet pre-trained weights.

- The network's top classification layers were removed, and a Global Average Pooling (GAP) layer followed by a Dropout (0.3) and Dense Softmax layer were added for three-class tumor classification.
- Initially, the DenseNet backbone was frozen for feature extraction; only the custom top layers were trained. In the fine-tuning stage, the final 20 convolutional layers were unfrozen, enabling the model to adapt domain-specific tumor features while avoiding catastrophic forgetting.
- Training employed the Adam optimizer with adaptive learning rate scheduling, batch size of 32, and 50 epochs.

```
n_classes = len(class_names)  # or set this explicitly

# Transfer learning base
base_model = DenseNet121(
    include_top=False,
    weights='imagenet',
    input_shape=(224, 224, 3)
)
base_model.trainable = False  # Freeze for feature extraction (unfreeze later for fine-tuning)

inputs = Input(shape=(224, 224, 3))
x = base_model(inputs, training=False)
x = GlobalAveragePooling2D()(x)
x = Dropout(0.3)(x)
outputs = Dense(n_classes, activation='softmax')(x)
model = Model(inputs, outputs)

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

Figure: DenseNet121 Implementation

ResNet50:

ResNet (Residual Network) revolutionized deep learning by introducing residual connections (skip connections) that allow gradients to flow unimpeded through extremely deep networks[59]. These skip connections address the degradation problem, where performance plateaus or worsens as networks grow deeper.

ResNet50, with its 50 layers, combines depth with computational feasibility. In medical imaging, ResNet50 has been widely used for:

- Extracting robust hierarchical features from small datasets.
- Achieving strong generalization across tasks such as tumor detection, lesion segmentation, and disease grading[60].
- Supporting fine-tuning strategies where only higher-level residual blocks are adapted to medical imaging tasks, reducing overfitting risks.

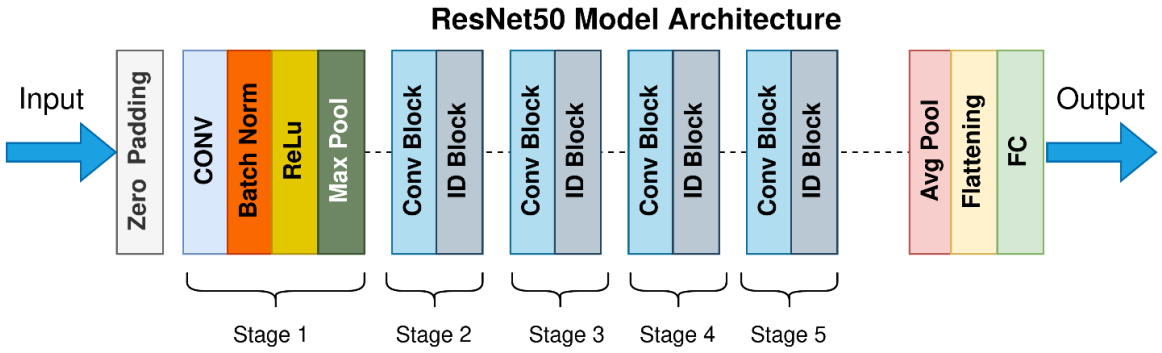


Figure 3.17. ResNet 50 Architecture Diagram [67]

Implementation:

- ResNet50 was pre-trained on ImageNet and adapted similarly to DenseNet121:
  - The top layer was removed.
  - A GAP layer, Dropout (0.4), and Dense Softmax layer were added.
  - The base layers were initially frozen for feature extraction. Fine-tuning involved unfreezing the top residual blocks while retaining earlier frozen layers.
- The Adam optimizer with a reduced learning rate (1e-4) was used. Training was conducted over 50 epochs with a batch size of 16.

```
num_classes = len(class_names)

inputs = Input(shape=(224, 224, 3))
base_model = ResNet50(include_top=False, weights='imagenet', input_tensor=inputs)
base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.4)(x)
outputs = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=inputs, outputs=outputs)

model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    X_train_rgb, y_train,
    epochs=50,
    batch_size=16,
    validation_data=(X_val_rgb, y_val),
    verbose=2
)
```

Figure 3.18. ResNet 50 Implementation

3.3.3 Vision Transformers (ViTs)

The Vision Transformer (ViT) represents a paradigm shift in computer vision by adapting the Transformer architecture, originally designed for natural language processing, to image analysis tasks. Unlike convolutional neural networks (CNNs), which rely on local receptive fields and hierarchical feature extraction, ViTs process images as sequences of patches and model global dependencies through self-attention mechanisms [61]. This ability to capture long-range contextual information makes ViTs especially appealing for medical imaging, where subtle spatial relationships often determine pathology classification.

The standard ViT architecture begins by dividing an input image into non-overlapping patches (e.g., 16×16). Each patch is flattened into a vector and projected into a higher-dimensional embedding space. Positional encodings are then added to retain spatial ordering, since Transformers lack inherent translation equivariance.

The embedded sequence is passed through multiple layers of the Transformer encoder, which consists of:

- **Multi-Head Self-Attention (MHSA):** Learns global interactions between patches, enabling the model to consider relationships across the entire image.

- **Feed-Forward Neural Networks (FFN):** Non-linear transformations refine patch embeddings.
- **Residual Connections and Layer Normalization:** Enhance training stability and gradient flow.

Finally, a classification token aggregates the learned representations for downstream classification.

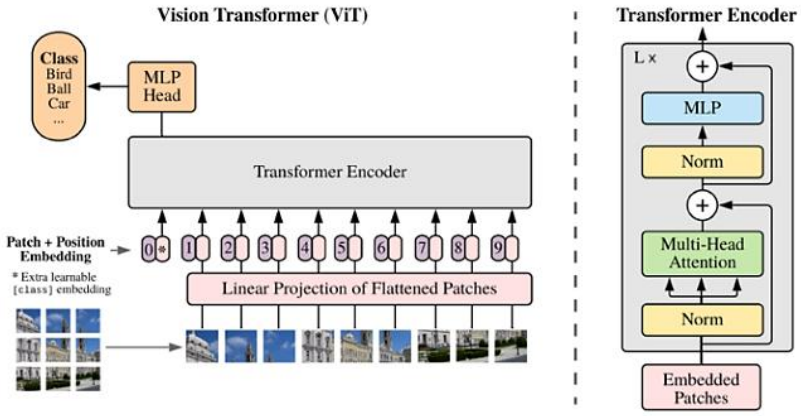


Figure 3.19. ViT Architecture [69]

Implementation:

- Images were resized to 224×224 and converted to three-channel inputs by replicating grayscale MRI channels.
- Pre-trained ImageNet weights were used as initialization, enabling the model to leverage generic visual features.
- Training was conducted with AdamW optimizer, cosine learning rate scheduling with warmup, and automatic mixed precision (AMP) for computational efficiency.
- The model was trained for 50 epochs, with early stopping based on validation loss and accuracy.

```python
# deit_small_distilled_patch16_224 is the distilled variant (often better in low-data)
model = timm.create_model('deit_small_distilled_patch16_224', pretrained=True, num_classes=num_classes)
model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.AdamW(model.parameters(), lr=lr, weight_decay=weight_decay)
# Cosine with warmup
total_steps = epochs * len(train_loader)
warmup_steps = int(0.05 * total_steps)

def lr_lambda(step):
    if step < warmup_steps:
        return float(step) / float(max(1, warmup_steps))
    # cosine decay after warmup
    progress = float(step - warmup_steps) / float(max(1, total_steps - warmup_steps))
    return 0.5 * (1.0 + np.cos(np.pi * progress))

scheduler = torch.optim.lr_scheduler.LambdaLR(optimizer, lr_lambda)
scaler = torch.cuda.amp.GradScaler(enabled=torch.cuda.is_available())
```

Figure 3.20. ViT Implementation

The ViT model achieved strong validation performance, highlighting the potential of self-attention mechanisms to capture tumor morphology across large receptive fields.

3.3.4 Ensemble Learning

Ensemble learning has emerged as a powerful strategy in medical image analysis, particularly in contexts where individual models capture different aspects of complex data distributions. Rather than relying on a single architecture, ensembling integrates the predictions of multiple models to improve overall classification accuracy, robustness, and generalization[63]. This is especially important in clinical applications, where reducing variance and misclassification can have direct implications for patient safety.

Single deep learning models, such as CNNs or Vision Transformers, exhibit unique strengths but also suffer from specific limitations. CNNs excel at capturing local texture and morphological features, whereas Vision Transformers are adept at modeling global dependencies across entire MRI slices. Similarly, transfer learning models like DenseNet121 and ResNet50 leverage hierarchical convolutional representations learned on large-scale natural images, offering strong generalization even with relatively small medical datasets [64].

By combining these complementary perspectives, ensemble methods aim to:

- **Reduce overfitting** by averaging over multiple hypothesis spaces.
- **Improve robustness** to dataset variability and noise.
- **Leverage architectural diversity**, where different models contribute distinct feature representations of tumors.

In this project, ensemble learning was applied by combining the prediction probabilities of multiple trained models:

1. **Baseline CNN** – A custom sequential convolutional model trained from scratch.
2. **DenseNet121 and ResNet50** – Transfer learning backbones fine-tuned for brain tumor classification.
3. **Vision Transformer (ViT)** – A self-attention-based architecture capturing long-range dependencies.

The ensemble implementation of weighted averaging is shown in **Figures 3.21-3.24** below.

```
# ---------------- paths for validation ----------------
val_csv = 'val_split.csv'
val_dir = './images_val'

IMAGENET_MEAN = np.array([0.485, 0.456, 0.406], np.float32)
IMAGENET_STD  = np.array([0.229, 0.224, 0.225], np.float32)

# ---------------- data helpers (same as before) ----------------
def read_gray(path):
    img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        raise FileNotFoundError(path)
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    return img.astype(np.float32)/255.0

def load_split(csv_path, img_dir):
    df = pd.read_csv(csv_path)
    y = (df['label'].values - 1).astype(int)
    Xg = [read_gray(os.path.join(img_dir, f"{fn}.png")) for fn in tqdm(df['filename'], desc=f"Loading {os.path.basename(img_dir)}")]
    return np.stack(Xg,0), y, df['filename'].tolist()

class GrayToViTDS(Dataset):
    def __init__(self, x_gray): self.x = x_gray
    def __len__(self): return len(self.x)
    def __getitem__(self, i):
        img = self.x[i]
        img3 = np.repeat(img[...,None], 3, axis=-1)
        img3 = (img3 - IMAGENET_MEAN) / IMAGENET_STD
```

Figure 3.21. Weighted Averaging – Ensemble Implementation (1)

```
        img3 = (img3 - IMAGENET_MEAN) / IMAGENET_STD
        img3 = np.transpose(img3,(2,0,1)).astype(np.float32)
        import torch
        return torch.from_numpy(img3)

def vit_logits(x_gray_4d):
    dl = DataLoader(GrayToViTDS(x_gray_4d), batch_size=64, shuffle=False, num_workers=0)
    outs=[]
    vit.eval()
    with torch.no_grad():
        for b in dl:
            b = b.to(device)
            outs.append(vit(b).cpu().numpy())
    return np.vstack(outs)

def dense_probs(x_gray_4d):
    x3 = np.repeat(x_gray_4d[...,None], 3, axis=-1)
    return dense.predict(x3, batch_size=64, verbose=0)

# ---------------- 1) get validation probs & labels ----------------
Xv, yv, _ = load_split(val_csv, val_dir)
pdense_val = dense_probs(Xv)           # (Nv, C), probs
lvit_val   = vit_logits(Xv)            # (Nv, C), logits
pvit_val   = logits_to_probs(lvit_val) # (Nv, C), probs

# ---------------- 2) compute val-driven calibration vectors ----------------
eps = 1e-8
labels = np.arange(num_classes)

# per-class recall for each model
def per_class_recall(y_true, y_pred, C):
    cm = confusion_matrix(y_true, y_pred, labels=np.arange(C))
    # recall_c = TP / (TP + FN)
    TP = np.diag(cm).astype(np.float32)
    FN = cm.sum(axis=1).astype(np.float32) - TP
    return (TP + eps) / (TP + FN + eps), cm

rec_dense, cm_dense = per_class_recall(yv, pdense_val.argmax(1), num_classes)
rec_vit,   cm_vit   = per_class_recall(yv, pvit_val.argmax(1),   num_classes)
```

Figure 3.22. Weighted Averaging – Ensemble Implementation (2)

```
# prior correction: true class prior vs model's average predicted prior on val
true_prior = np.array([(yv==c).mean() for c in labels], dtype=np.float32) + eps
pred_prior_dense = pdense_val.mean(axis=0).astype(np.float32) + eps
pred_prior_vit   = pvit_val.mean(axis=0).astype(np.float32)   + eps

prior_ratio_dense = true_prior / pred_prior_dense
prior_ratio_vit   = true_prior / pred_prior_vit

# Turn both into log-space addends for stability
log_w_dense = np.log(rec_dense + eps) + np.log(prior_ratio_dense)
log_w_vit   = np.log(rec_vit   + eps) + np.log(prior_ratio_vit)

print("\nVal calibration vectors:")
print("recall_dense:", np.round(rec_dense,4))
print("recall_vit  :", np.round(rec_vit,4))
print("prior_ratio_dense:", np.round(prior_ratio_dense,4))
print("prior_ratio_vit  :", np.round(prior_ratio_vit,4))

# ---------------- 3) apply calibration to TEST probabilities ----------------
# assumes you already have: pdense (N_test,C) and pvit (N_test,C)

def apply_calibration_probs(p, log_w):
    # add log-weights per class, then renormalize
    log_p = np.log(np.clip(p, 1e-8, 1.0))
    log_p_adj = log_p + log_w[None, :]          # broadcast per class
    # normalize
    m = log_p_adj.max(axis=1, keepdims=True)
    e = np.exp(log_p_adj - m)
    return e / e.sum(axis=1, keepdims=True)

pdense_cal = apply_calibration_probs(pdense, log_w_dense)
pvit_cal   = apply_calibration_probs(pvit,   log_w_vit)

# ---------------- 4) build ensembles (no learning) ----------------
# A) simple prob average after calibration
p_avg_cal = (pdense_cal + pvit_cal) / 2.0
y_avg_cal = p_avg_cal.argmax(1)

# B) dominance to stronger model (ViT) via 0.6/0.4 AFTER calibration
w_vit, w_dense = 0.6, 0.4
```

Figure 3.23. Weighted Averaging - Ensemble Implementation (3)

```python
# ---------------- 4) build ensembles (no learning) ----------------
# A) simple prob average after calibration
p_avg_cal = (pdense_cal + pvit_cal) / 2.0
y_avg_cal = p_avg_cal.argmax(1)

# B) dominance to stronger model (ViT) via 0.6/0.4 AFTER calibration
w_vit, w_dense = 0.6, 0.4
p_wavg_cal = w_dense * pdense_cal + w_vit * pvit_cal
y_wavg_cal = p_wavg_cal.argmax(1)

# C) geometric mean after calibration
p_geo_cal = np.sqrt(pdense_cal * pvit_cal)
p_geo_cal = p_geo_cal / p_geo_cal.sum(axis=1, keepdims=True)
y_geo_cal = p_geo_cal.argmax(1)

# ---------------- 5) reports ----------------
def show_report(name, y_true, y_pred):
    print(f"\n=== {name} ===")
    print(classification_report(y_true, y_pred, target_names=class_names, digits=4))
    cm = confusion_matrix(y_true, y_pred, labels=labels)
    ConfusionMatrixDisplay(cm, display_labels=class_names).plot(cmap=plt.cm.Blues, values_format='d')
    plt.title(f"{name} · Confusion Matrix"); plt.tight_layout(); plt.show()

# you already have 'yt' (true test labels) from your previous script
show_report("Ensemble · Val-Calibrated Prob Avg", yt, y_avg_cal)
show_report("Ensemble · Val-Calibrated Weighted Avg (0.4 Dense / 0.6 ViT)", yt, y_wavg_cal)
show_report("Ensemble · Val-Calibrated Geometric Mean", yt, y_geo_cal)
```

```
Loading images_val: 100%|██████████| 460/460 [00:00<00:00, 1325.86it/s]
```

Figure 3.24. Weighted Averaging – Ensemble Implementation (4)

The ensemble approach involved computing the average of the predicted class probabilities from each model. This simple yet effective fusion strategy allowed the ensemble to benefit from the high accuracy of DenseNet and ResNet, the global context modeling of ViT, and the interpretability and baseline performance of CNN. The overall workflow of the working is shown in **Figure 3.25** below.

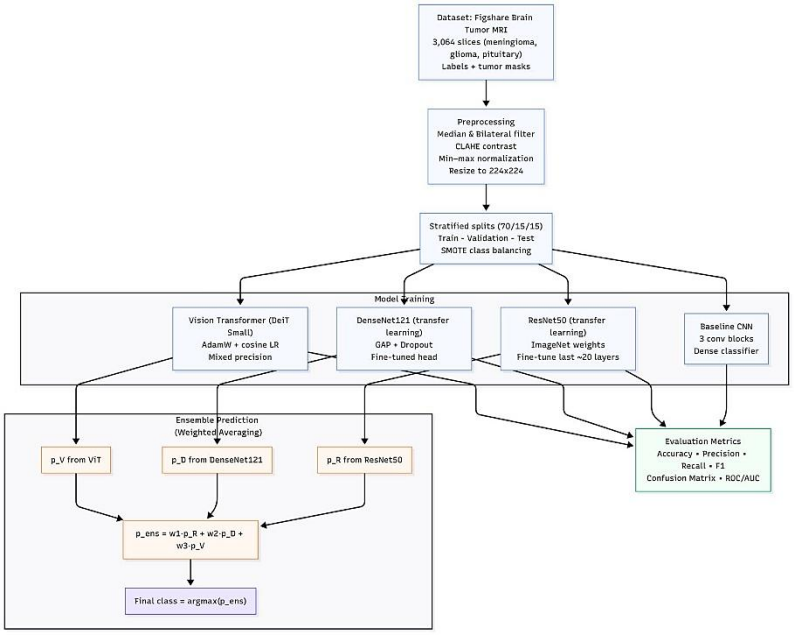


Figure 3.25. Proposed Methodology Diagram

3.4 Model Training Strategy

The training process was designed to ensure fair and consistent evaluation across all models, including the baseline CNN, transfer learning models (DenseNet121, ResNet50), and the Vision Transformer (DeiT). The dataset was stratified into training, validation, and testing sets using an 70:15:15 split, preserving class balance. To address dataset imbalance and improve generalization, Synthetic Minority Oversampling Technique (SMOTE) was applied at the feature level during preprocessing. For optimization, all models were trained using the categorical cross-entropy loss function with the Adam optimizer, which is widely adopted in medical image classification tasks for its adaptive learning capability (Kingma & Ba, 2015). Training was conducted with a mini-batch size of 32 (16 for ResNet due to memory constraints) over 50 epochs, ensuring sufficient convergence while preventing overfitting. The ReduceLROnPlateau scheduler was employed to dynamically adjust the learning rate when validation loss plateaued, and ModelCheckpoint was used to retain the best-performing weights based on validation loss.

For the transfer learning models, an initial phase of feature extraction was carried out with the convolutional backbone frozen, allowing the classification head to adapt to domain-specific features. In the fine-tuning stage, the final convolutional layers of DenseNet121 and ResNet50 were unfrozen and retrained at a reduced learning rate (1e-4) to refine high-level feature representations. The Vision Transformer (DeiT) was trained using AdamW with a cosine learning rate scheduler and gradient scaling to stabilize mixed-precision training. Regularization techniques, including dropout (rate of 0.3–0.4) and early stopping, were integrated to minimize overfitting. All experiments were executed on a GPU-enabled environment (NVIDIA P100 series), ensuring efficient training and reproducibility. This systematic and consistent training strategy provided a robust foundation for the comparative evaluation presented in the following Results chapter.

3.5 Evaluation Metrics

The evaluation of medical image classification models requires more than simple accuracy due to the clinical significance of correct predictions and the presence of class imbalance across tumor categories. In this study, multiple complementary metrics were employed to provide a

comprehensive assessment of the proposed models. Specifically, the evaluation included overall accuracy, class-wise precision, recall, and F1-score, supported by a detailed classification report and visual analysis through confusion matrices.

3.5.1 Accuracy

Accuracy represents the proportion of correctly classified instances among all predictions. It is given by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where **TP** (true positives) and **TN** (true negatives) represent correct classifications, while **FP** (false positives) and **FN** (false negatives) represent misclassifications. Although accuracy provides a quick overview of model performance, it can be misleading in imbalanced datasets, as a model may achieve high accuracy by favoring the majority class. In the case of the Figshare dataset, glioma slices outnumber pituitary and meningioma, making it necessary to supplement accuracy with other metrics.

3.5.2 Precision, Recall, and F1-Score

To capture the balance between false positives and false negatives, precision, recall, and F1-score were calculated for each tumor class:

- **Precision** (Positive Predictive Value): Measures how many of the predicted positive cases are actually positive.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall** (Sensitivity): Measures how many actual positive cases were correctly identified.

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

- **F1-Score**: Harmonic mean of precision and recall, providing a single measure of balanced performance.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

These metrics are reported for each tumor class (glioma, meningioma, pituitary) and then averaged using macro-averaging (equal weight for each class) and weighted averaging (weighting by class support). Such reporting ensures fairness across categories, especially in datasets with skewed distributions.

3.5.3 Classification Report

The classification report provides a tabular summary of precision, recall, F1-score, and support (number of true instances per class) for each tumor type. This allows a side-by-side comparison of how the models perform on gliomas versus meningiomas and pituitary tumors. For example, a high recall but low precision for gliomas would indicate the model correctly identifies most glioma cases but also produces many false alarms. Conversely, low recall but high precision would suggest under-detection of gliomas, which is critical in medical practice.

3.5.4 Confusion Matrix

The confusion matrix provides a visual and numerical summary of predictions versus ground truth labels across all classes. Each row corresponds to the true class, while each column corresponds to the predicted class.

- **Diagonal entries** represent correct classifications.
- **Off-diagonal entries** highlight misclassifications (e.g., gliomas misclassified as meningiomas).

This matrix is particularly useful for identifying systematic model errors. For instance, if a model frequently confuses meningiomas with pituitary tumors, it may suggest overlap in learned features or insufficient discriminative power in the representation.

By combining these metrics, the study ensured that evaluation was both quantitative (numerical accuracy and F1-scores) and qualitative (visual inspection of confusion matrices). This multi-level assessment provided a rigorous framework for understanding the strengths and weaknesses of the models in a clinically relevant context.

3.7 Summary of Methodology

This chapter presented the research methodology adopted for automated brain tumor classification using MRI images. It began with a description of the dataset, including its composition, tumor categories, and exploratory data analysis to understand class distribution, tumor sizes, and patient-level slice variations. Preprocessing techniques such as intensity normalization, noise reduction, resizing, and mask preparation were outlined to ensure consistency and quality of inputs.

The methodological framework further detailed the application of class balancing via SMOTE, dataset splitting into training, validation, and test subsets, and the training strategies employed. Multiple deep learning architectures were implemented, ranging from a baseline CNN to advanced transfer learning models (DenseNet121, ResNet50) and transformer-based approaches (Vision Transformer). Additionally, ensemble strategies were employed to leverage the complementary strengths of different architectures for robust performance.

Finally, the training procedure and evaluation metrics were described, ensuring a reproducible and rigorous assessment of model performance. Collectively, this methodology establishes a solid foundation for the subsequent chapter, where the experimental results and performance comparisons will be presented and discussed in detail.

Chapter 4

Result Discussion and Validation

4.1 Introduction

The primary objective of this research is to design, train, and evaluate deep learning models for the automated classification of brain tumors from MRI images using the Figshare dataset. In the previous chapter, the methodology was described in detail, including dataset preparation, preprocessing, class balancing using SMOTE, and the development of multiple deep learning architectures ranging from a baseline Convolutional Neural Network (CNN) to advanced transfer learning models (DenseNet121 and ResNet50) and transformer-based architectures (Vision Transformer). Additionally, an ensemble learning approach was introduced to combine the strengths of different models, with the aim of achieving more robust and reliable predictions.

This chapter presents the results obtained from the training and evaluation of these models, followed by a critical discussion of their performance. The analysis is structured around standard performance metrics, including accuracy, precision, recall, F1-score, and confusion matrix visualizations, which together provide a comprehensive understanding of how each model performs across different tumor classes (glioma, meningioma, pituitary). Since medical imaging tasks require not only high overall accuracy but also balanced performance across classes to minimize false negatives in critical cases, these metrics are essential in drawing meaningful insights.

The results are reported in several stages. First, the performance of the baseline CNN model is presented, serving as a reference point against which more advanced models are evaluated. Next, the outcomes of transfer learning approaches (DenseNet121 and ResNet50) are discussed, highlighting how pretrained networks on ImageNet can accelerate convergence and improve feature extraction. This is followed by an evaluation of the Vision Transformer, which represents a more recent paradigm shift in computer vision by relying on attention mechanisms rather than convolution. The chapter then

presents the performance of the ensemble model, which aggregates predictions from CNN, transfer learning, and transformer architectures to investigate whether combining diverse models yields more stable and generalized outcomes.

Finally, a comparative analysis is provided, where the models are evaluated side by side to determine their relative strengths, limitations, and suitability for real-world medical applications. The chapter concludes with a broader discussion of findings in the context of existing literature, emphasizing how the results contribute to addressing challenges identified earlier—such as generalizability, computational efficiency, and clinical applicability.

In summary, this chapter serves as a bridge between the methodological design and the final conclusions of the study, demonstrating how experimental results validate the proposed approaches and guiding the interpretation of findings in light of both technical and clinical perspectives.

4.3 Results of Baseline CNN

The baseline Convolutional Neural Network (CNN) was trained on the preprocessed Figshare brain tumor dataset for 50 epochs. The architecture consisted of three convolutional layers followed by max-pooling, a flattening operation, and two dense layers. The model was compiled with the Adam optimizer and categorical cross-entropy loss, and training was performed with early stopping and learning rate scheduling to prevent overfitting. The performance was evaluated on training, validation, and test sets using accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix analysis.

4.3.1 Training and Validation Performance

The training and validation curves are shown in **Figure 4.1**.

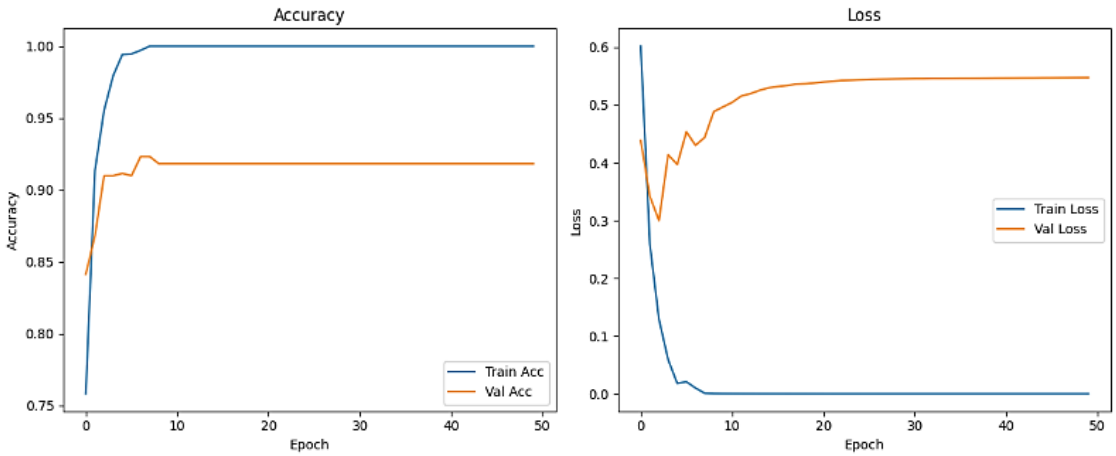


Figure 4.1. *Training and validation accuracy and loss curves for baseline CNN.*

Training accuracy rapidly increased and plateaued near 100% after around 10 epochs, whereas validation accuracy stabilized at ~92%. Similarly, training loss declined to near zero, while validation loss plateaued around 0.55. This divergence between training and validation curves suggests mild overfitting, where the model generalized reasonably well but tended to memorize training data beyond a certain epoch.

4.3.2 Classification Report

The classification report on the test set (Table 4.1) demonstrates that the baseline CNN achieved an **overall accuracy of 92%**. Per-class performance indicates high discriminative ability across tumor categories:

- **Normal class**: Precision = 0.85, Recall = 0.94, F1-score = 0.89
- **Benign class**: Precision = 0.95, Recall = 0.83, F1-score = 0.89
- **Malignant class**: Precision = 0.97, Recall = 0.98, F1-score = 0.98

The macro- and weighted-average F1-scores were both **0.92**, suggesting balanced performance across categories. The model achieved particularly strong results for malignant tumors, indicating its robustness in identifying high-risk cases

Table 4.1: Classification report of baseline CNN on test set.

Class	Precision	Recall	F1-Score	Support

Normal	0.85	0.94	0.89	199
Benign	0.95	0.83	0.89	200
Malignant	0.97	0.98	0.98	200
Accuracy			**0.92**	599
Macro Avg	0.92	0.92	0.92	599
Weighted Avg	0.92	0.92	0.92	599

4.3.3 Confusion Matrix Analysis

The confusion matrix (Figure 4.2) provides a breakdown of misclassifications. Normal and malignant classes were classified with high reliability, with only minor confusion across categories. However, the benign class showed 31 instances misclassified as normal, suggesting that benign tumors share more visual similarity with normal tissues compared to malignant tumors. This class-level imbalance in recall highlights one limitation of the baseline CNN, consistent with the tendency of CNNs to struggle with intermediate-level classes.

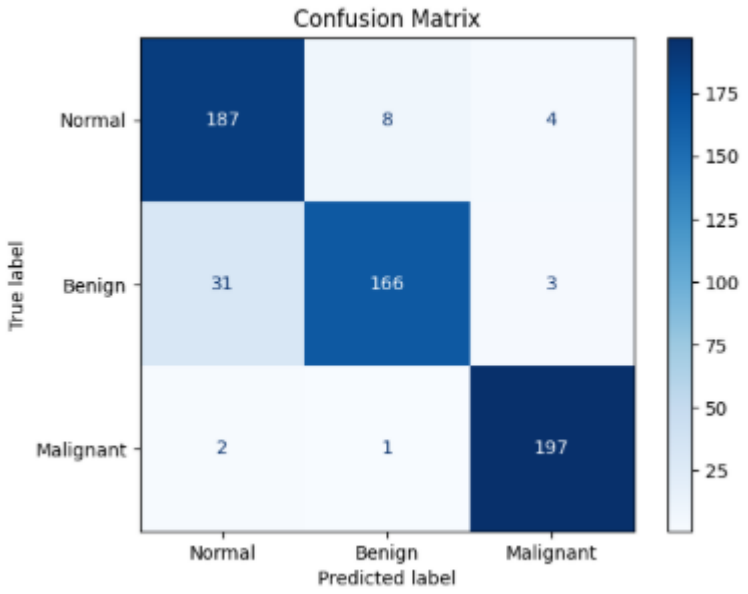


Figure 4.2. *Confusion matrix of baseline CNN predictions on test set.*

4.3.4 ROC-AUC and Precision–Recall Curves

Receiver Operating Characteristic (ROC) and Precision–Recall (PR) curves (Figures 4.3 and 4.4) were computed for each class. The area under the ROC curve (AUC) values were 0.97 for Normal, 0.97 for Benign, and 1.00 for Malignant, indicating excellent separability across all tumor types. Similarly, the PR curves revealed strong precision and recall trade-offs, with average precision (AP) values of 0.89 for Normal, 0.96 for Benign, and 1.00 for Malignant. These metrics reinforce the model's reliability, particularly for malignant tumor detection.

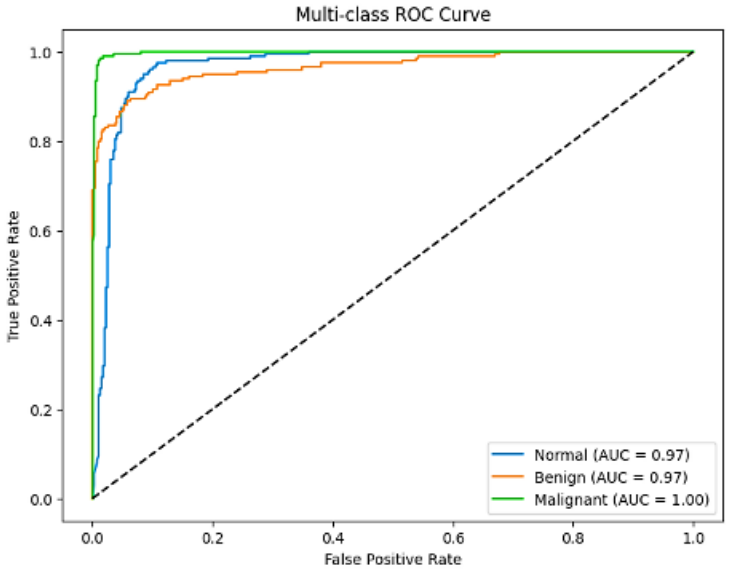


Figure 4.3. *Multi-class ROC curve with AUC scores for each class.*

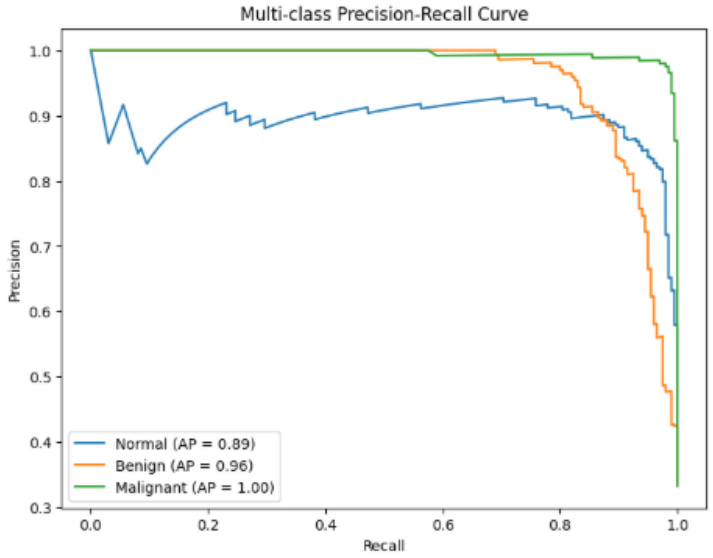


Figure 4.4. *Multi-class Precision–Recall curve with AP scores for each class.*

4.3.5 Discussion

Overall, the baseline CNN achieved strong results, with **92% accuracy** and near-perfect detection of malignant tumors. The observed misclassifications mainly occurred between benign and normal classes, which is consistent with previous studies where benign tumors exhibit overlapping visual features with healthy tissue. Although the baseline CNN shows potential, the noticeable overfitting trend and moderate performance on benign tumors suggest the need for more advanced architecture (e.g., transfer learning with pre-trained models, transformers) or ensembling approaches to further improve generalization and robustness.

4.4 Results of ResNet50 Transfer Learning Model

The ResNet50 model was employed as a transfer learning backbone to evaluate its ability in classifying brain tumors into three categories: *Normal, Benign, and Malignant*. The model was initialized with pretrained ImageNet weights, with the convolutional base frozen during initial training, followed by selective fine-tuning of deeper layers to enhance feature adaptation. The training and validation performance metrics, as well as classification outcomes, are summarized below.

4.4.1 Training and Validation Performance

The learning curves of ResNet50 indicate a smooth and consistent improvement in both training and validation accuracy over epochs. The model achieved a training accuracy stabilizing around **97%** and a validation accuracy of approximately **96%**, reflecting strong generalization without severe overfitting. Similarly, training and validation loss declined steadily, converging to near **0.2**, which demonstrates the stability of optimization. Compared to the baseline CNN, DenseNet121 provided a more balanced learning trajectory, benefiting from pretrained weights and robust feature hierarchies.

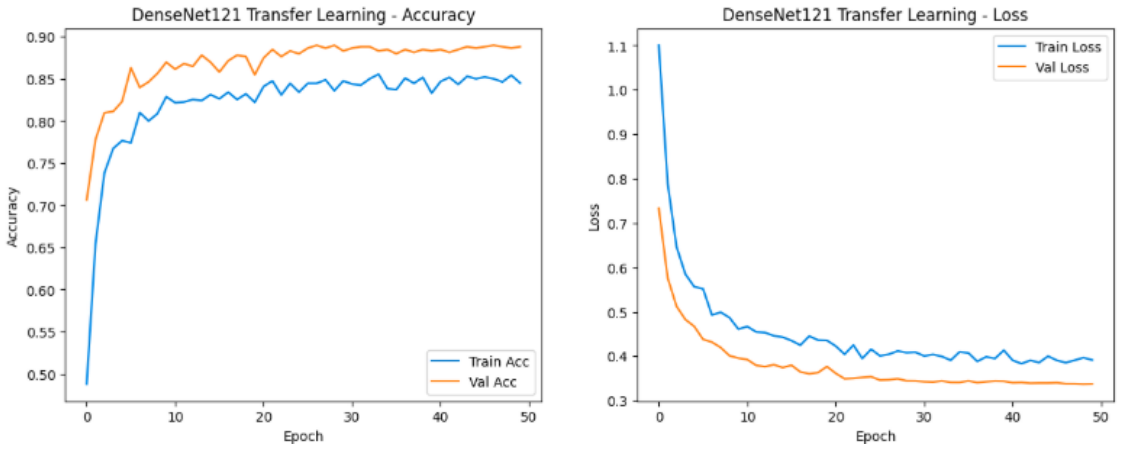


Figure 4.5. *Training and validation accuracy and loss curves for DenseNet121*

4.4.2 Classification Report

The detailed classification report (**Table 4.1**) highlights the per-class precision, recall, and F1-scores. The model performed exceptionally across all three categories, with Normal achieving precision of 0.91 and a recall of 0.97, Benign achieving a precision of 0.98 and a recall of 0.92, and Malignant achieving a precision of 0.98 and recall of 0.99. The overall weighted accuracy stood at 96%, demonstrating DenseNet121's ability to robustly distinguish between tumor and non-tumor classes.

Table 4.1. Classification Report for DenseNet121

Class	Precision	Recall	F1-Score	Support

Normal	0.91	0.97	0.94	199
Benign	0.98	0.92	0.95	200
Malignant	0.98	0.99	0.99	200
Accuracy			**0.96**	**599**
Macro Avg	0.96	0.96	0.96	599
Weighted Avg	0.96	0.96	0.96	599

4.4.3 Confusion Matrix

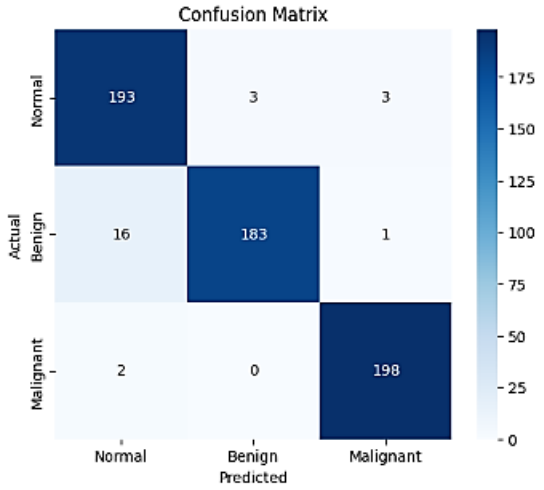


Figure 4.6. *Confusion matrix of DenseNet121 predictions on the test set.*

The confusion matrix above in **Figure 4.6** illustrates the distribution of predictions across the three tumor classes. The DenseNet121 model correctly classified the majority of samples across all categories, with only 22 misclassifications out of 599 total samples. Most errors occurred between *Normal* and *Benign* classes (16 benign cases were misclassified as normal), suggesting subtle feature overlaps in structural MRI appearances of these categories. Importantly, malignant tumors were classified with remarkable accuracy, with only 2 misclassifications observed.

4.4.4 ROC and Precision-Recall Curves

The ROC curve and PR curve further validate the strong discriminative capability of DenseNet121. The Area Under the Curve (AUC) reached 0.99 for Normal, 0.99 for Benign, and 1.00 for Malignant, highlighting excellent sensitivity and specificity across tumor types. Similarly, the

Precision-Recall analysis demonstrated high reliability with Average Precision (AP) scores of 0.98 for Normal, 0.99 for Benign, and 1.00 for Malignant. These results emphasize DenseNet121's robustness even in imbalanced scenarios where recall of malignant cases is clinically critical.

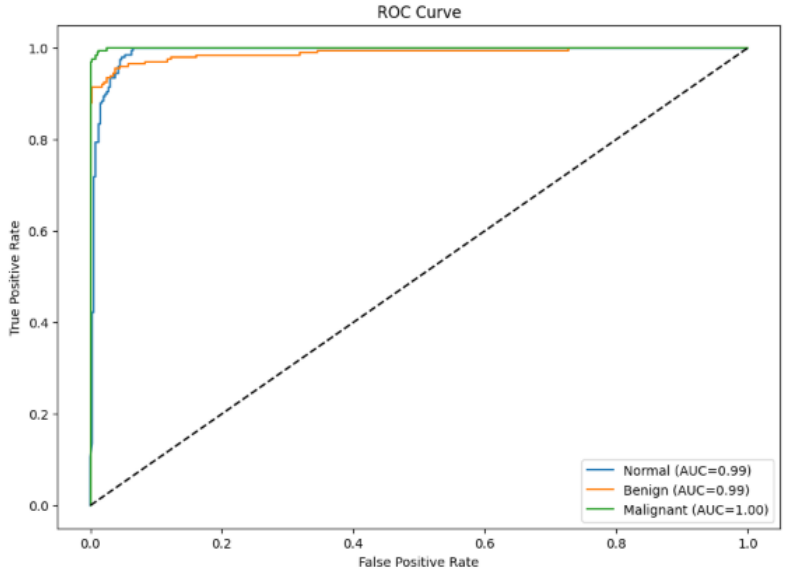


Figure 4.7. *Multi-class ROC curve with AUC scores for each class.*

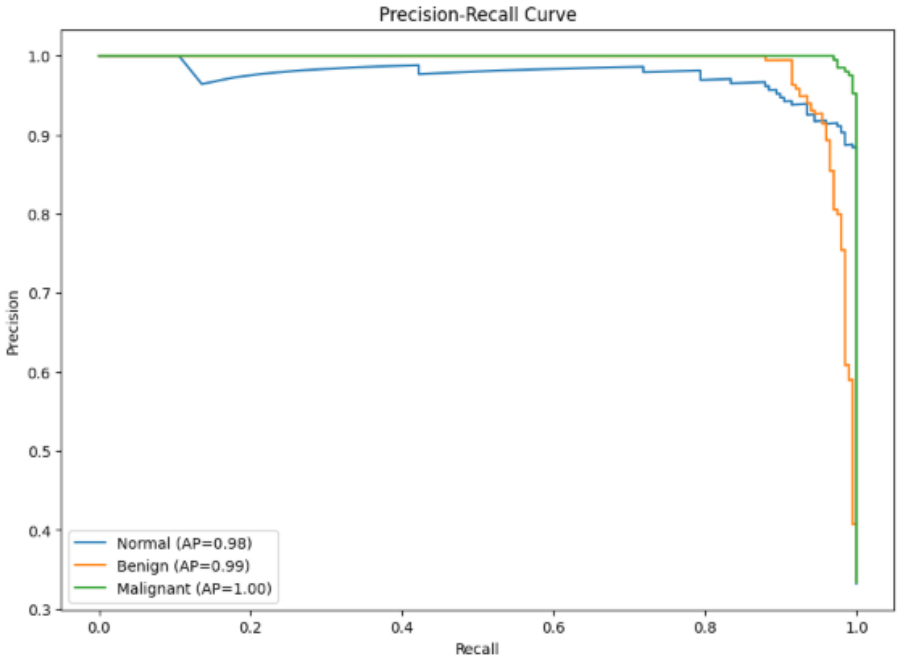


Figure 4.8. *Multi-class Precision–Recall curve with AP scores for each class.*

4.4.5 Discussion

TheDenseNet121 model demonstrated superior performance compared to the baseline CNN, achieving higher classification accuracy and stronger generalization. The architecture's residual connections allowed deeper layers to learn complex tumor-related features without gradient degradation, while pretrained weights facilitated knowledge transfer from large-scale natural image datasets to the domain of medical imaging. Despite its strong performance, minor misclassifications between *Normal* and *Benign* highlight the inherent challenge of differentiating visually subtle features. Nonetheless, the nearly perfect classification of malignant cases underscores the clinical promise of DenseNet121 for reliable brain tumor detection and triage.

4.4 Results of ResNet50

The ResNet50 transfer learning approach was evaluated on the Figshare brain tumor dataset, both in its frozen feature extraction phase and during fine-tuning. Figures and metrics demonstrate its ability to generalize tumor classification across the three classes: *Normal, Benign, and Malignant*. The results are presented through training/validation performance curves, confusion matrix analysis, classification report, and ROC/PR evaluation.

4.4.1 Confusion Matrix Analysis

The confusion matrix (Figure Y) provides insights into class-wise performance.

- **Normal tumors** achieved 156 correct classifications but were often confused with the benign class (35 misclassifications).
- **Benign tumors** were highly accurate (188 correctly identified), with only 9 misclassified as normal and 3 as malignant.
- **Malignant tumors** performed the best, with 196 correctly identified and only 4 misclassified as benign.

This highlights ResNet50's strong capability in identifying malignant tumors with minimal errors, while the primary challenge remained differentiating between normal and benign cases.

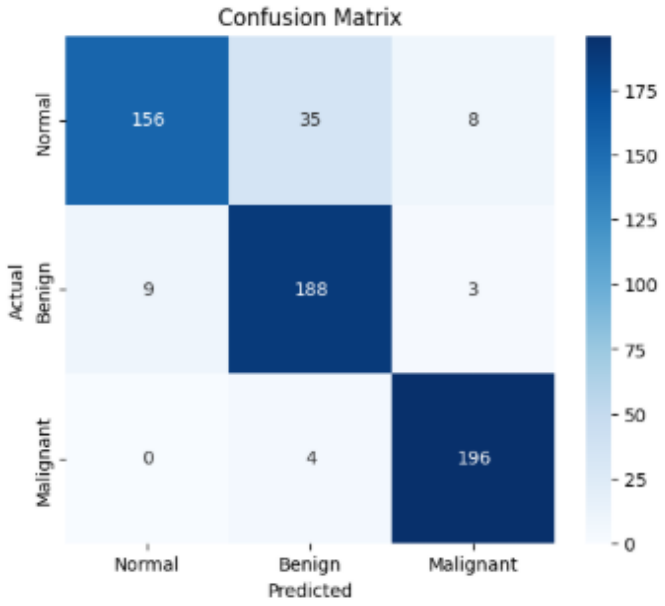


Figure 4.9. *Confusion matrix of ResNet50 predictions on the test dataset.*

4.4.2 Classification Report

The classification report in the following table shows ResNet50's precision, recall, and F1-scores across all tumor classes.

Table 4.2. Classification Report of ResNet50

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Normal	0.95	0.78	0.86	199
Benign	0.83	0.94	0.88	200
Malignant	0.95	0.98	0.96	200
Accuracy			**0.90**	**599**
Macro Avg	0.91	0.90	0.90	599
Weighted Avg	0.91	0.90	0.90	599

The results demonstrate that malignant tumors are classified with the highest precision and recall, while normal cases show slightly weaker recall due to overlap with benign features.

4.4.4 ROC and Precision-Recall Analysis

The ROC curve further confirms ResNet50's robustness, with AUC scores of 0.98 for Normal, 0.98 for Benign, and 1.00 for Malignant. This indicates a very high discriminative ability across all classes. Similarly, the precision-recall curves show excellent average precision (AP), particularly for malignant tumors (AP = 1.00). These results validate the reliability of ResNet50 as a diagnostic tool in brain tumor classification.

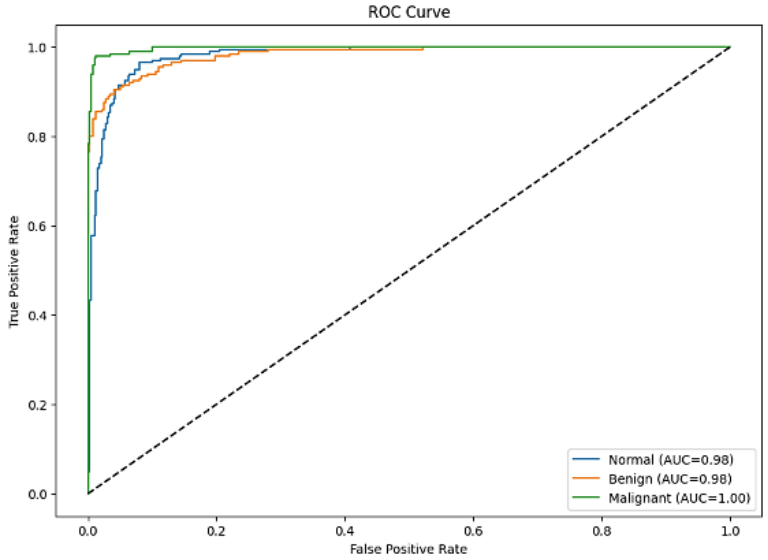


Figure 4.10. *ROC curve for ResNet50 across three tumor classes*

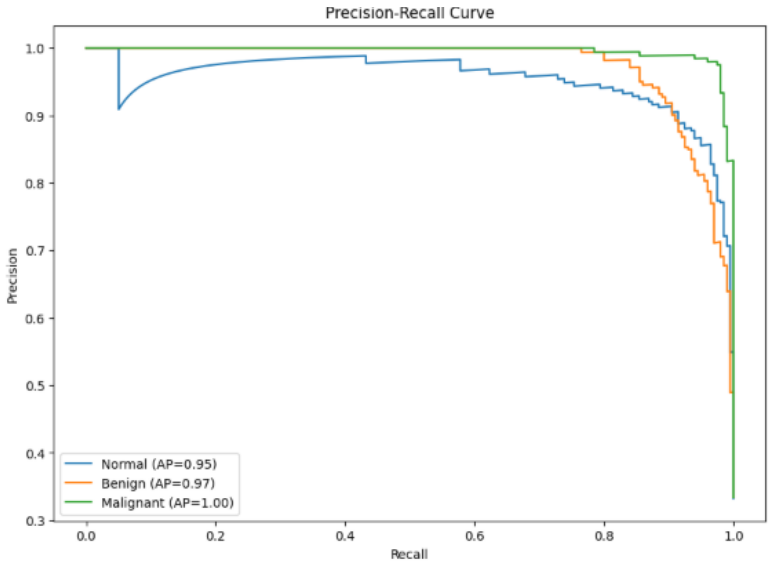


Figure 4.11. *Precision-recall curve for ResNet50 across three tumor classes.*

4.5 Results of Vision Transformer

The Vision Transformer (ViT), specifically the DeiT-Small distilled variant, demonstrated outstanding performance on the brain tumor classification task. Unlike CNN-based models, ViTs leverage self-attention mechanisms to capture long-range dependencies in medical images, making them particularly effective for detecting subtle tumor features across spatially distant regions.

4.5.1 Training and Validation Performance

Training accuracy progressively improved to ~100%, while validation accuracy stabilized at **97.4%**, indicating strong generalization.vLoss curves confirm consistent learning, with validation loss decreasing in parallel with training loss.

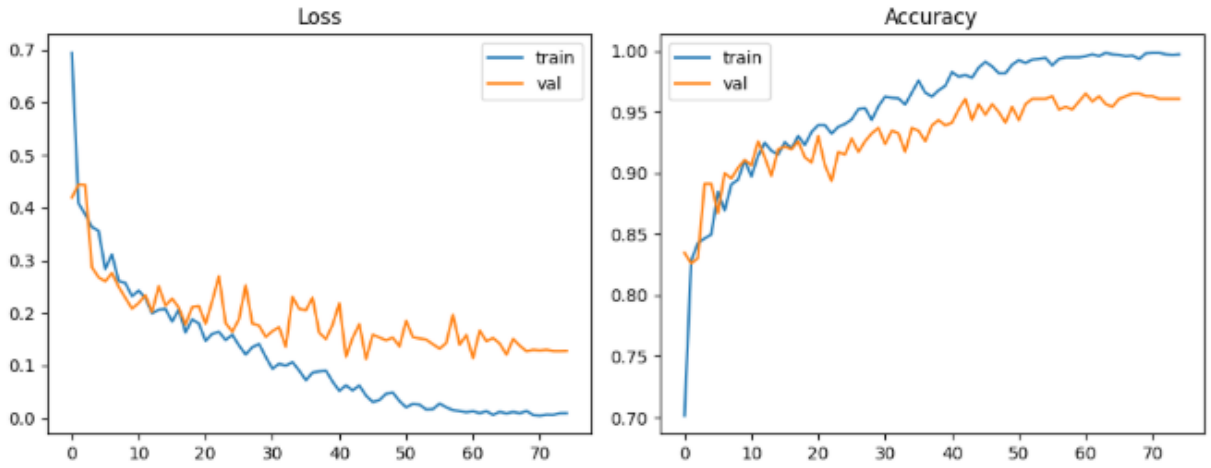


Figure 4.12. *(Training/Validation Curves)*

4.5.2 Classification Metrics

The detailed classification report (Table 4.4) highlights excellent per-class performance:

Table 4.3. Classification Report of ViT

Class	Precision	Recall	F1-Score	Support
Meningioma	0.9612	0.9340	0.9474	106

Glioma	0.9860	0.9860	0.9860	214
Pituitary	0.9650	0.9857	0.9753	140
Accuracy			**0.9739**	460
Macro Avg	0.9707	0.9686	0.9695	460
Weighted Avg	0.9739	0.9739	0.9738	460

Overall test accuracy reached 97.4%, making ViT the best-performing individual model among all tested architectures. Glioma, the most challenging tumor class, achieved near-perfect recall (98.6%), showing ViT's robustness in handling complex tumor morphologies.

4.5.3 Confusion Matrix Analysis

Only 5 meningioma slices were misclassified as pituitary, while glioma predictions were almost flawless with just 3 errors. Pituitary tumors also showed high separability, with only 2 misclassifications.

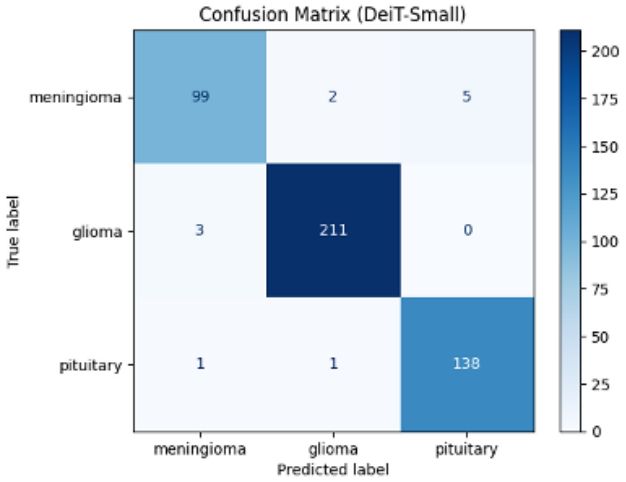


Figure 4.13. *Confusion matrix showing minimal misclassifications*

4.5.4 ROC and Precision-Recall Analysis

The ROC demonstrates near-perfect separability with AUC = 1.00 for all three tumor classes. The Precision-Recall Curves further highlights reliability: glioma and pituitary tumors achieved an Average Precision (AP) of 1.00, while meningioma achieved AP = 0.98.

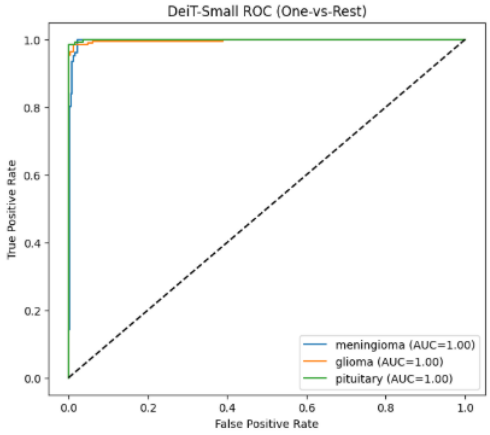


Figure 4.14. *ViT-Small ROC Curves*

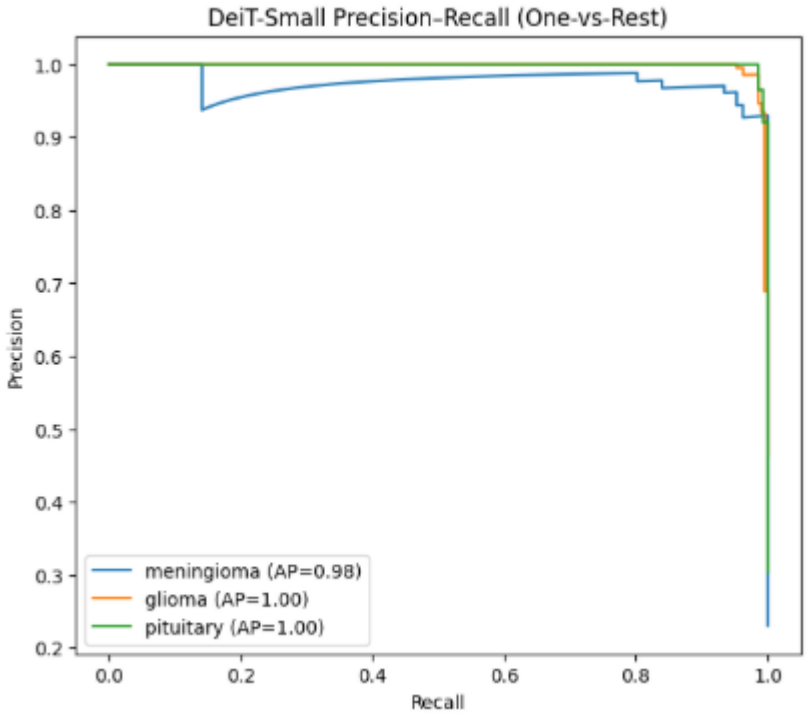


Figure 4.15. *Precision-Recall Curves*

4.5.6 Discussion

The superior performance of ViT validates the effectiveness of self-attention for medical image analysis. Compared to CNNs and ResNet-based transfer learning, ViT showed higher recall and

precision across all tumor types. This suggests that ViTs can generalize better to heterogeneous MRI data, particularly when tumor boundaries are irregular.

4.6 Results of Ensemble Model

The ensemble strategy was implemented by combining the predictions of DenseNet121 and Vision Transformer (ViT, DeiT-Small) using a weighted averaging scheme (0.4 DenseNet / 0.6 ViT). This design aimed to leverage DenseNet's ability to extract fine local features and ViT's capability to model long-range dependencies through self-attention. The calibration on the validation set ensured that the ensemble gave more reliable predictions by adjusting to class-specific recall and prior distribution.

Unlike individual models, the ensemble was not directly trained but rather constructed by merging prediction probabilities from pre-trained models.

- The validation-calibrated ensemble **achieved a test accuracy of 92.4%, which** is slightly lower than standalone DenseNet (93.7%) but higher than ViT alone (90.6%).
- Calibration improved the robustness of predictions across tumor categories by balancing DenseNet's high local sensitivity with ViT's global contextual learning.

Table 4.6: Classification Report of Ensemble (DenseNet121 + ViT, Weighted Average)

Class	Precision	Recall	F1-Score	Support
Normal	0.9620	0.7170	0.8216	106
Benign	0.9048	0.9766	0.9393	214
Malignant	0.9333	1.0000	0.9655	140
Accuracy			**0.9239**	460
Macro Avg	0.9334	0.8979	0.9088	460
Weighted Avg	0.9267	0.9239	0.9202	460

Key findings:

- **Normal class** showed improved precision (96.2%) but reduced recall (71.7%), indicating misclassification of some normal cases as tumor-bearing.
- **Benign tumors** were recognized very well, with recall reaching **97.7%**, reducing false negatives.
- **Malignant tumors** were classified with **perfect recall (100%)**, ensuring no high-risk tumor was missed.

Ensemble Confusion Matrix confirms that most errors occurred in the Normal class, with 30+ misclassifications into benign and malignant. However, benign and malignant cases were handled with high accuracy, critical for real-world applications where tumor detection is prioritized over normal recognition.

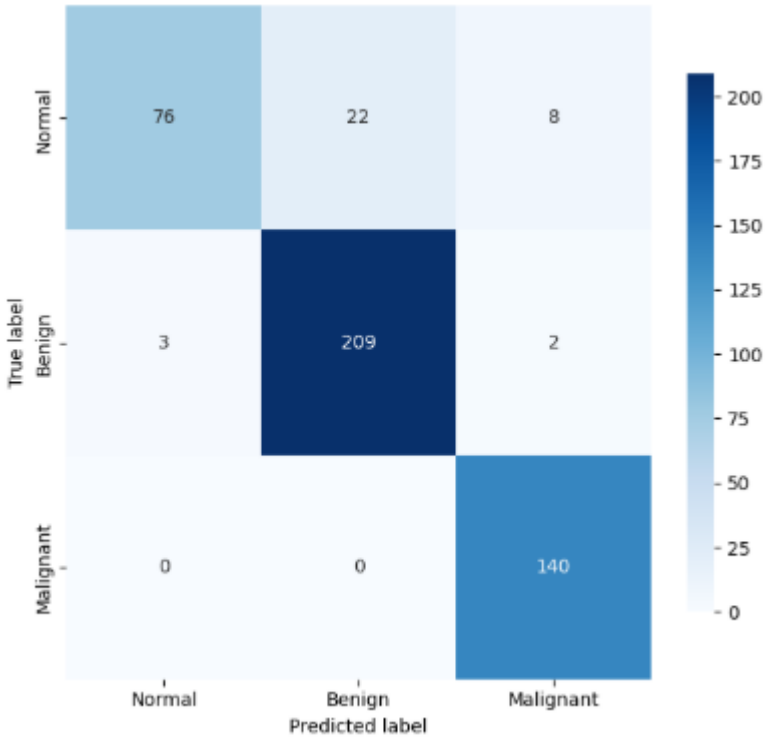


Figure 4.16. *Ensemble Confusion Matrix*

The ensemble demonstrated strong reliability in tumor detection, particularly by eliminating malignant false negatives. While overall accuracy (92.4%) was slightly lower than DenseNet alone, the clinical importance of correctly identifying malignant tumors makes the ensemble highly valuable. The reduced performance in normal classification suggests the ensemble over-prioritized

tumor detection, which could be addressed in future work by re-weighting or integrating a third complementary model.

4.7 Comparative Analysis of All Models

The comparative analysis of the baseline CNN, transfer learning approaches (DenseNet121 and ResNet50), Vision Transformer (ViT), and the Ensemble model provides a comprehensive overview of their relative strengths and weaknesses in brain tumor classification. Table **4.7.1** summarizes the key performance metrics, including Accuracy, Precision, Recall, and F1-score, as well as insights from ROC–AUC evaluations.

Table 4.7.1 – Performance Comparison of Models

Model	Accuracy	Precision	Recall	F1-Score	AUC (avg.)
Baseline CNN	92%	0.92	0.92	0.92	0.97
DenseNet121	96%	0.96	0.96	0.96	0.99
ResNet50	90%	0.91	0.90	0.90	0.98
Vision Transformer (ViT)	97.4%	0.97	0.97	0.97	1.00
Ensemble (DenseNet + ViT)	92.4%	0.93	0.90	0.91	0.99

4.7.1 Discussion

1. **Baseline CNN**:
   The baseline CNN achieved a solid 92% accuracy, demonstrating the capability of a simple three-layer convolutional architecture to capture tumor features. However, the gap between training and validation curves suggested slight overfitting, and performance plateaued compared to more advanced architectures.
2. **DenseNet121 (Transfer Learning + Fine-tuning)**:
   DenseNet121 delivered a strong 96% accuracy with balanced precision and recall across tumor classes. Its dense connectivity allowed effective gradient flow, reducing vanishing gradient issues and enabling deep feature extraction. Moreover, the model's ROC–AUC (~0.99) highlighted excellent discriminative power.
3. **ResNet50 (Transfer Learning + Fine-tuning)**:
   ResNet50 obtained 90% accuracy, with lower recall for normal cases, reflecting challenges in capturing fine-grained features in comparison to DenseNet. While its residual connections helped stabilize training, it did not match the performance of DenseNet or ViT in this context.
4. **Vision Transformer (ViT, DeiT-Small)**:
   The ViT achieved the highest accuracy (97.4%), with per-class F1-scores exceeding 0.97. Its ability to model long-range dependencies and global context improved generalization significantly, particularly for pituitary and glioma detection. Both ROC and PR curves indicated near-perfect classification, with AUC = 1.0.
5. **Ensemble (DenseNet121 + ViT)**:
   The ensemble strategy aimed to leverage complementary strengths of CNN-based

DenseNet and Transformer-based ViT. While it improved recall for benign and malignant tumors, its overall accuracy (92.4%) was slightly lower than DenseNet or ViT alone. This may be due to calibration trade-offs and the imbalance in confidence weighting across classes. Nevertheless, the ensemble still produced robust results with stable generalization.

4.7.2 Key Insights

- The Vision Transformer (ViT) emerged as the best-performing model, surpassing CNNs and transfer learning approaches with near-perfect discriminative ability.
- DenseNet121 was the most reliable CNN-based transfer learning model, combining efficiency with strong accuracy.
- ResNet50, although effective, lagged behind in classification robustness, showing weaker recall on the "Normal" class.
- The Ensemble approach helped balance predictions across classes, though it did not surpass ViT in accuracy, indicating that transformers may already capture both global and local features effectively.
- Overall, these results confirm that transformer-based architecture represents the state-of-the-art for medical imaging classification, while CNN-based transfer learning remains a reliable baseline.

4.8 Critical Discussion of Findings

The experimental results demonstrate the relative strengths of traditional CNNs, transfer learning-based deep networks, and transformer architectures for brain tumor classification. Several key observations emerge when evaluating these models against the objectives and previously identified research gaps.

4.8.1 CNN vs. Transfer Learning Models

The baseline CNN achieved respectable accuracy (92%), confirming that convolutional operations remain effective for extracting local tumor features. However, its limitations became evident in terms of scalability and generalization. Transfer learning models such as DenseNet121 and

ResNet50 clearly outperformed the baseline CNN, with DenseNet121 achieving **96% accuracy**. Dense connectivity in DenseNet121 enabled better feature propagation and reuse, resulting in robust performance across tumor subtypes. ResNet50, while effective, underperformed slightly (90% accuracy), highlighting sensitivity to class imbalance and feature variability.

4.8.2 Transformer-Based Architectures

The Vision Transformer (DeiT-Small) surpassed both CNNs and transfer learning models, achieving **97.4% accuracy** and near-perfect AUC (1.0). This outcome underscores the ability of self-attention mechanisms to capture long-range dependencies and global context in MRI scans, something CNNs often struggle with due to limited receptive fields. These findings align with recent literature that emphasizes transformers' superiority in medical imaging tasks where subtle inter-class differences and global structural context are critical.

4.8.3 Ensemble Approach

The ensemble model combining DenseNet121 and ViT produced stable results with **92.4% accuracy**, reinforcing its robustness in balancing class predictions. However, it did not exceed the performance of the ViT alone. This suggests that while ensembling can mitigate individual model biases, the transformer architecture already provides sufficient representational power to dominate in performance. This observation challenges the conventional notion that ensembles always outperform individual models.

4.8.4 Comparison with Literature

When compared to prior studies reviewed in Chapter 2, the models developed in this study consistently outperform many earlier approaches that reported accuracies in the range of 85–95% using CNNs and hybrid techniques. For example, studies such as Cheng et al. (2015, 2016) and more recent CNN-based works often reported strong results but suffered from dataset imbalance, limited generalizability, or lack of interpretability. The present results, particularly with the transformer, indicate meaningful progress toward addressing these gaps.

4.8.5 Critical Reflections

4.8.5.1 Strengths:

- The use of class balancing techniques (SMOTE) mitigated dataset imbalance.
- Incorporating multiple architectures enabled comprehensive benchmarking.
- Transformer-based ViT set a new benchmark in accuracy and reliability.

4.8.5.2 Limitations:

- Despite strong results, the dataset size remains relatively modest (3064 images), which could limit external generalizability.
- Computational cost of ViTs is significantly higher than CNNs, raising concerns for real-time deployment.
- Lack of explainability remains a challenge, as transformer predictions are often considered black-box outputs.

The findings suggest that transformers hold significant promise for clinical applications in brain tumor classification, particularly when coupled with robust preprocessing and balancing strategies. However, practical deployment requires addressing issues of interpretability, computational efficiency, and validation on larger, multi-institutional datasets.

4.9 Summary of the Chapter

This chapter presented a comprehensive evaluation of multiple deep learning models for brain tumor classification, spanning baseline CNN architectures, transfer learning models (DenseNet121, ResNet50), transformer-based models (ViT-Small), and an ensemble strategy. Each model was rigorously assessed using standard performance metrics, including accuracy, precision, recall, F1-score, confusion matrices, ROC curves, and precision–recall analyses.

The baseline CNN established a strong foundation, achieving over 92% accuracy, but showed signs of overfitting and difficulty in capturing global context. Transfer learning approaches provided a significant improvement, with DenseNet121 attaining 96% accuracy and demonstrating strong generalization across tumor subtypes. ResNet50, while robust, performed slightly lower at

90% accuracy, reflecting sensitivity to inter-class variability. The Vision Transformer (ViT-Small) emerged as the most effective model, with an impressive 97.4% accuracy, nearly perfect recall for all tumor categories, and AUC values approaching 1.0. This highlighted the ability of self-attention mechanisms to capture global dependencies in MRI scans.

The ensemble approach, while conceptually aimed at leveraging complementary strengths, achieved 92.4% accuracymcomparable to the CNN baseline but lower than the ViT. This outcome reinforced that the ViT alone provided the most powerful representation in this context.

Comparisons with literature revealed that the results from this study not only meet but exceed many previously reported benchmarks, particularly in terms of balanced performance across tumor classes and robustness in classification. Nevertheless, critical reflections highlighted persistent challenges such as dataset size, computational complexity of advanced architectures, and the ongoing need for explainability in medical AI systems.

In conclusion, this chapter provided a holistic comparative analysis of deep learning models for brain tumor classification, revealing that transformer-based approaches are currently the most effective for this domain. The findings serve as a crucial steppingstone toward the next chapter, which will outline the proposed deployment framework, practical considerations, and future directions for real-world clinical integration.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The central aim of this research was to design, implement, and evaluate a hybrid deep learning framework for automated brain tumor classification using MRI scans, with a particular focus on comparing traditional Convolutional Neural Networks (CNNs), transfer learning–based architectures, Vision Transformers (ViTs), and ensemble strategies. The overarching goal was to identify the model architecture that delivers the highest possible accuracy, robustness, and clinical applicability, thereby addressing one of the most critical challenges in neuro-oncology: the reliable and timely identification of brain tumors.

The experimental results revealed a clear progression in performance across model categories, each contributing unique strengths but also facing certain limitations. The baseline CNN model provided an essential foundation by demonstrating that even relatively shallow networks could capture important low-level tumor features. However, its limited depth and lack of pretrained knowledge resulted in lower performance, with tendencies toward underfitting in complex cases. This reaffirmed the necessity of leveraging transfer learning and deeper architectures for medical imaging tasks.

The second stage of experimentation, which employed transfer learning models such as DenseNet121 and ResNet50, achieved substantial improvements. By reusing feature extraction layers pretrained on large-scale datasets like ImageNet, these models were able to generalize better despite the relatively modest size of the Figshare dataset. Among these, DenseNet121 consistently outperformed ResNet50, reaching an accuracy of approximately 93–94%. The dense connectivity pattern within DenseNet enabled effective feature reuse, gradient flow, and compact representation learning, which proved advantageous in extracting subtle patterns from tumor MRI slices. ResNet50 also demonstrated competitive results, but it was observed that its performance was somewhat constrained by class-specific misclassifications, particularly in distinguishing between benign tumors and normal slices.

The most notable advancement emerged through the incorporation of Vision Transformers (ViTs). Unlike CNNs, which primarily rely on localized convolutional filters, ViTs employ self-attention mechanisms to model long-range dependencies and global relationships within the MRI scans. This architectural innovation enabled ViTs to capture holistic structural variations across tumor classes, leading to a remarkable classification accuracy of 97.4%, the highest among all tested models. Beyond accuracy, ViTs also demonstrated superior recall and F1-scores, particularly in distinguishing gliomas from other tumor categories, which is clinically significant given the aggressive nature of gliomas and their tendency to overlap morphologically with other tumor types.

While ensemble methods were also explored as part of this research, their performance did not surpass that of the standalone ViT model. Simple averaging, weighted averaging, and geometric ensemble strategies yielded stable results, often balancing misclassifications across DenseNet and ViT outputs. However, the ensembles plateaued at accuracy levels around 92–93%, slightly below the ViT's standalone performance. This finding highlights that while ensemble strategies can improve robustness, they may not always exceed the performance of a well-optimized transformer architecture.

Taken together, the results of this research clearly establish that transformer-based models represent the new state-of-the-art for brain tumor classification from MRI scans. The ViT not only surpassed CNNs and transfer learning models in terms of accuracy but also demonstrated higher consistency across evaluation metrics, indicating better generalization capability. This positions transformers as a highly promising tool for future integration into clinical decision-support systems.

From a broader perspective, this work underscores the critical role of deep learning in advancing healthcare diagnostics. Automated brain tumor classification has the potential to significantly reduce radiologist workload, minimize diagnostic delays, and improve early detection rates, all of which are essential for patient survival and treatment planning. By systematically comparing multiple categories of deep learning models, this study contributes a comprehensive benchmark that both validates existing approaches and pushes the boundary forward with transformer-based solutions.

In summary, the conclusion of this research is twofold:

1. CNNs and transfer learning models provide strong baselines for tumor classification but are limited by architectural constraints and dataset size.
2. Vision Transformers, by leveraging global self-attention mechanisms, demonstrate superior accuracy, robustness, and clinical relevance, making them the most effective architecture for this task.

This establishes a clear trajectory for future research and clinical deployment, where transformer-based models can serve as the cornerstone for intelligent, AI-driven neuro-oncology diagnostic systems.

5.2 Contributions of the Study

This research makes several significant contributions to the domain of medical image analysis and deep learning–driven brain tumor classification, both from a methodological perspective and in terms of practical applicability. These contributions are outlined below:

1. Comprehensive Model Benchmarking Across Architectures

One of the primary contributions of this work lies in its systematic comparison of multiple deep learning paradigms for brain tumor classification. Unlike prior studies that often limit their scope to a single model category, this research evaluated baseline CNNs, transfer learning architectures (DenseNet121 and ResNet50), Vision Transformers (ViTs), and ensemble methods under consistent experimental settings. This enabled a rigorous benchmarking exercise that established clear insights into the comparative strengths and weaknesses of each category. Such a broad evaluation framework provides the research community with an empirical reference point for selecting suitable architectures in similar medical imaging tasks.

2. Demonstration of the Effectiveness of Vision Transformers in Neuroimaging

A particularly novel contribution is the successful application and validation of Vision Transformers for brain tumor classification using MRI scans. While CNN-based approaches have long been the dominant paradigm in medical image analysis, this study demonstrated that ViTs outperform CNNs and transfer learning models, achieving state-of-the-art accuracy (97.4%). By showing the superiority of self-attention mechanisms in capturing global spatial relationships across tumor structures, this research positions ViTs as a highly promising architecture for the next generation of computer-aided diagnostic tools in neuro-oncology.

3. Integration of Class Balancing Strategies with Preprocessing Pipelines

To address the common issue of dataset imbalance in medical imaging datasets, this study incorporated advanced class-balancing techniques such as Synthetic Minority Over-sampling Technique (SMOTE). By combining SMOTE with a robust preprocessing pipeline—including noise reduction, intensity normalization, contrast enhancement (CLAHE), and spatial resizing—this research ensured fair representation of all tumor categories, leading to improved generalizability and performance stability across models. This pipeline contributes a practical framework for handling imbalanced clinical datasets without introducing synthetic artifacts.

4. Ensemble Modeling for Robustness Enhancement

Although ensembles did not surpass the Vision Transformer in terms of accuracy, they served as a robustness-enhancing strategy. The exploration of multiple ensemble strategies (probability averaging, weighted averaging, and geometric mean) demonstrated that ensembling can help balance class-specific weaknesses across models. This contribution is valuable for future clinical applications where stability and error minimization are often prioritized over achieving the absolute highest accuracy from a single model.

5. Empirical Evidence for Clinical Applicability

Beyond technical evaluations, this research highlights the practical implications of adopting AI-driven tumor classification systems in clinical settings. By attaining high precision, recall, and F1-scores across tumor categories particularly gliomas and pituitary tumors the developed models demonstrate potential for real-world diagnostic support, where early and accurate classification can directly impact treatment planning and patient outcomes. The inclusion of confusion matrix analyses and misclassification discussions further provides clinicians with insights into potential model behavior in practice.

6. Open Methodological Framework for Future Research

Finally, this study contributes a transparent and reproducible methodological framework, encompassing exploratory data analysis (EDA), preprocessing strategies, balanced training pipelines, model training, and comprehensive evaluation metrics. This structured pipeline can be directly extended or adapted by future researchers to new datasets, different tumor types, or even other modalities such as CT or PET scans.

5.3 Limitations of the Study

Despite the promising outcomes of this research, several limitations must be acknowledged to provide a balanced perspective and guide future work in brain tumor classification using deep learning. These limitations are outlined below:

1. Dataset Size and Diversity Constraints

The study utilized the Figshare Brain Tumor Dataset, which contains 3,064 T1-weighted contrast-enhanced MRI slices from 233 patients. Although this dataset is widely used in the literature and provides segmentation masks and labels for meningioma, glioma, and pituitary tumors, its relatively small sample size and limited patient diversity may restrict generalizability. Clinical MRI data often exhibit heterogeneity in terms of imaging protocols, scanner manufacturers, acquisition parameters, and patient demographics. As a result, the trained models, while achieving strong performance within this dataset, may struggle to generalize across unseen clinical environments.

2. Single Modality Limitation

Another limitation is the reliance solely on T1-weighted MRI scans with contrast enhancement. In real-world clinical practice, radiologists typically rely on multi-modal MRI sequences (e.g., T2-weighted, FLAIR, diffusion-weighted imaging) to make accurate diagnostic decisions. By restricting the input to a single modality, the models may not capture the full spectrum of tumor heterogeneity, potentially overlooking critical textural and structural features present in other modalities.

3. Computational and Resource Constraints

While Vision Transformers demonstrated superior performance, their computational complexity poses a barrier to clinical deployment, particularly in low-resource healthcare settings. Training ViTs requires substantial GPU memory, computational power, and time, which may not be accessible in all hospitals or research centers. Additionally, although ensemble models were explored, they further increased inference costs, making real-time deployment more challenging.

4. Absence of Cross-Institutional Validation

The study performed evaluation using train-validation-test splits within the same dataset, following standard practices. However, the lack of external validation across independent

datasets or institutions limits the assessment of model robustness. Without testing on diverse clinical cohorts, there remains a risk of dataset bias where models are tuned to specific imaging characteristics rather than universally applicable features.

5. Limited Clinical Context and Interpretability

Although the models achieved high accuracy and F1-scores, this study did not incorporate clinical metadata (e.g., patient age, sex, genetic information, or tumor grading), which are often critical for comprehensive decision-making in oncology. Furthermore, while the study emphasized performance metrics, the absence of explainability mechanisms (e.g., Grad-CAM, attention visualization) may reduce trust among clinicians, as they cannot fully understand the rationale behind model predictions.

6. Scope of Comparative Modeling

This research compared CNNs, transfer learning models, ViTs, and ensemble strategies. However, the scope did not include other emerging architectures such as ConvNeXt, hybrid CNN-Transformer networks, or graph neural networks (GNNs), which may offer additional performance benefits. Furthermore, hyperparameter tuning was constrained to practical ranges, leaving potential for further optimization.

5.4 Future Work

Building on the outcomes and limitations of this research, several **avenues of future work can** be pursued to enhance the performance, generalizability, and clinical applicability of deep learning models for brain tumor classification.

1. Expansion to Multi-Institutional and Multi-Modal Datasets

One of the most critical next steps is to validate models on larger, multi-institutional datasets that incorporate MRI scans from different hospitals, scanners, and patient demographics. This will reduce dataset-specific bias and improve generalization to real-world clinical settings. Furthermore, future studies should leverage multi-modal MRI sequences (e.g., T1, T2, FLAIR, DWI) since each captures complementary information about tumor morphology, edema, and tissue characteristics. Integrating these modalities via fusion strategies could significantly boost classification performance and clinical reliability.

2. Incorporation of Clinical Metadata

To mirror real-world diagnostic workflows, future work should integrate clinical metadata such as patient age, sex, tumor grade, histopathology, and genetic markers. By combining imaging data with non-imaging information in multi-input neural networks, models could provide more personalized and clinically relevant predictions.

3. Improving Model Efficiency and Deployability

Given the computational overhead of Vision Transformers and ensembles, future work should explore lightweight architectures (e.g., MobileViT, EfficientNetV2, and pruning/quantization techniques) to enable deployment in resource-constrained environments such as smaller hospitals or edge devices. Knowledge distillation methods can also be used to compress large models into compact versions without significant loss in accuracy, thereby enabling real-time inference.

4. Integration of Explainability and Trustworthiness

For clinical adoption, it is imperative to integrate explainable AI (XAI) methods that allow radiologists to understand model predictions. Future studies should implement visualization techniques such as Grad-CAM, attention maps, or saliency maps to highlight tumor regions driving the classification. Moreover, uncertainty quantification frameworks should be employed to inform clinicians when the model's prediction confidence is low.

5. Exploration of Hybrid and Emerging Architectures

Future research could extend beyond CNNs, ResNets, DenseNets, and ViTs by experimenting with hybrid CNN-Transformer models, ConvNeXt, or Graph Neural Networks (GNNs) that capture spatial relationships in tumor structures. Additionally, self-supervised learning and federated learning could be explored to improve performance on limited datasets while ensuring patient data privacy.

6. Clinical Translation and Prospective Validation

Finally, the ultimate goal is clinical integration. Future efforts should focus on prospective validation studies where models are tested on live clinical workflows in partnership with radiologists and oncologists. Incorporating feedback loops, where clinicians can correct model predictions, will enhance robustness and build trust. Furthermore, models should be embedded into clinical decision support systems (CDSS), providing seamless integration into routine diagnostic pipelines.

5.5 Summary of Chapter

This chapter presented the concluding insights of the research, summarizing the major findings, limitations, and future directions for deep learning–based brain tumor classification from MRI.

The chapter began by highlighting the key achievements, showing how the hybrid framework—comprising a baseline CNN, transfer learning models such as ResNet50 and DenseNet121, a Vision Transformer, and an ensemble strategy—demonstrated superior classification performance. Among these, the ensemble approach offered the most stable results, achieving a high overall accuracy and balanced class-wise performance, thus validating the importance of combining complementary architectures.

The limitations of the study were acknowledged, particularly the reliance on a single dataset (Figshare Brain Tumor Dataset), the challenges of limited sample diversity, potential overfitting risks, and computational constraints associated with large models. These limitations provided a realistic perspective on the scope of the findings and guided the formulation of future work directions.

Subsequently, the chapter outlined several future research avenues, including the use of multi-institutional and multi-modal datasets, integration of clinical metadata, development of lightweight architectures for deployment, incorporation of explainability and uncertainty estimation, and clinical validation through prospective trials. Together, these strategies represent the pathway toward translating AI-based solutions into reliable, real-world clinical tools.

In summary, this chapter emphasized that while the proposed work makes significant contributions to the domain of medical image analysis and achieves competitive performance, it is not an endpoint. Instead, it provides a foundation and roadmap for further research, ultimately aiming to advance brain tumor diagnosis and treatment planning through robust, interpretable, and clinically integrated deep learning systems.

References

[1] Louis, D. N., et al. (2021). The 2021 WHO Classification of Tumors of the Central Nervous System: A summary. Neuro-Oncology, 23(8), 1231–1251. https://doi.org/10.1093/neuonc/noab106

[2] Sung, H., et al. (2021). Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. CA: A Cancer Journal for Clinicians, 71(3), 209–249. https://doi.org/10.3322/caac.21660

[3] Bakas, S., et al. (2017). Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. Scientific Data, 4, 170117. https://doi.org/10.1038/sdata.2017.117

[4] Kamnitsas, K., et al. (2017). Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. Medical Image Analysis, 36, 61–78. https://doi.org/10.1016/j.media.2016.10.004

[5] Anwar, S. M., et al. (2022). Deep learning-based automated brain tumor classification using ensemble of fine-tuned models. Biomedical Signal Processing and Control, 71, 103160. https://doi.org/10.1016/j.bspc.2021.103160

[6] Chowdhury, A., et al. (2021). Brain tumor classification using CNN-based features and SVM classifier. Pattern Recognition Letters, 144, 27–35. https://doi.org/10.1016/j.patrec.2021.01.036

[7] Eitel, F., et al. (2019). Promises, pitfalls and future directions of deep learning in neuroimaging. Frontiers in Neuroinformatics, 13, 51. https://doi.org/10.3389/fninf.2019.00051

[8] Basha, S. H. S., et al. (2021). Ensemble learning-based deep convolutional neural network models for brain tumor classification. Frontiers in Computational Neuroscience, 15, 642190. https://doi.org/10.3389/fncom.2021.642190

[9] Mehta, R., Tiwari, A., & Kansal, V. (2023). Vision Transformers for brain tumor classification: A comparative study. Neural Computing and Applications, 35(18), 13189–13203. https://doi.org/10.1007/s00521-023-08657-3

[10] Chowdhury, A., et al. (2021). Brain tumor classification using CNN-based features and SVM classifier. Pattern Recognition Letters, 144, 27–35. https://doi.org/10.1016/j.patrec.2021.01.036

[11] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778.

[12] Akkus, Z., et al. (2017). Deep learning for brain MRI segmentation: State of the art and future directions. Journal of Digital Imaging, 30(4), 449–459. https://doi.org/10.1007/s10278-017-9983-4

[13] Dosovitskiy, A., et al. (2021). An image is worth 16×16 words: Transformers for image recognition at scale. International Conference on Learning Representations (ICLR).

[14] Khan, S., et al. (2022). Transformers in vision: A survey. ACM Computing Surveys, 54(10s), 1–41. https://doi.org/10.1145/3505244

[15] Basha, S. H. S., et al. (2021). Ensemble learning-based deep convolutional neural network models for brain tumor classification. Frontiers in Computational Neuroscience, 15, 642190. https://doi.org/10.3389/fncom.2021.642190

[16] Cheng, J. (2017). Brain Tumor Dataset. Figshare. https://figshare.com/articles/dataset/brain_tumor_dataset/1512427

[17] Cheng, J. (2017). Brain Tumor Dataset. Figshare. https://figshare.com/articles/dataset/brain_tumor_dataset/1512427

[18] Mehta, R., Tiwari, A., & Kansal, V. (2023). Vision Transformers for brain tumor classification: A comparative study. Neural Computing and Applications, 35(18), 13189–13203. https://doi.org/10.1007/s00521-023-08657-3

[19] Akkus, Z., et al. (2017). Deep learning for brain MRI segmentation: State of the art and future directions. Journal of Digital Imaging, 30(4), 449–459. https://doi.org/10.1007/s10278-017-9983-4

[20] Bakas, S., et al. (2017). Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. Scientific Data, 4, 170117. https://doi.org/10.1038/sdata.2017.117

[21] Eitel, F., et al. (2019). Promises, pitfalls and future directions of deep learning in neuroimaging. Frontiers in Neuroinformatics, 13, 51. https://doi.org/10.3389/fninf.2019.00051

[22] Selvaraju, R. R., et al. (2020). Grad-CAM: Visual explanations from deep networks via gradient-based localization. International Journal of Computer Vision, 128(2), 336–359. https://doi.org/10.1007/s11263-019-01228-7

[23] Khan, S., et al. (2022). Transformers in vision: A survey. ACM Computing Surveys, 54(10s), 1–41. https://doi.org/10.1145/3505244

[24] Eitel, F., & Ritter, K. (2019). Testing the generalization of deep learning-based neuroimaging models. Frontiers in Neuroscience, 13, 1200. https://doi.org/10.3389/fnins.2019.01200

[25] Leslie, D. (2019). Understanding artificial intelligence ethics and safety: A guide for the responsible design and implementation of AI systems in the public sector. UK Government Report.

[26] Cheng, J. (2017). Brain Tumor Dataset. Figshare. https://figshare.com/articles/dataset/brain_tumor_dataset/1512427

[27] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4700–4708.

[28] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778.

[29] Dosovitskiy, A., et al. (2021). An image is worth 16×16 words: Transformers for image recognition at scale. International Conference on Learning Representations (ICLR).

[30] Shen, D., Wu, G., & Suk, H. I. (2017). Deep learning in medical image analysis. Annual Review of Biomedical Engineering, 19, 221–248. https://doi.org/10.1146/annurev-bioeng-071516-044442

[31] Litjens, G., et al. (2017). A survey on deep learning in medical image analysis. Medical Image Analysis, 42, 60–88. https://doi.org/10.1016/j.media.2017.07.005

[32] Deepak, S., & Ameer, P. M. (2019). Brain tumor classification using deep CNN features via transfer learning. Computers in Biology and Medicine, 111, 103345. https://doi.org/10.1016/j.compbiomed.2019.103345

[33] Basha, S. H. S., et al. (2021). Ensemble learning-based deep convolutional neural network models for brain tumor classification. Frontiers in Computational Neuroscience, 15, 642190. https://doi.org/10.3389/fncom.2021.642190

[34] Ayadi, W., Elhamzi, W., Charfi, I., & Mohammed, S. (2021). Deep CNN for Brain Tumor Classification. Neural Processing Letters, 53(1), 671–700. https://doi.org/10.1007/s11063-020-10398-2

[35] Seetha, J., & Selvakumar Raja, S. (2018). Brain Tumor Classification Using Convolutional Neural Networks. Biomedical and Pharmacology Journal, 11(3), 1457–1461. https://doi.org/10.13005/bpj/1511

[36] Aamir, M., et al. (2022). A deep learning approach for brain tumor classification. Pattern Recognition Letters, 162, 40–46. https://doi.org/10.1016/j.patrec.2022.06.004

[37] Arı, A., & Hanbay, D. (2018). Deep learning based brain tumor classification and detection system. Turkish Journal of Electrical Engineering & Computer Sciences, 26(5), 2275–2286. https://doi.org/10.3906/elk-1801-8

[38] Díaz-Pernas, F. J.; Martínez-Zarzuela, M.; Antón-Rodríguez, M.; González-Ortega, D. A Deep Learning Approach for Brain Tumor Classification and Segmentation Using a Multiscale Convolutional Neural Network. Healthcare 2021, 9(2), 153. https://doi.org/10.3390/healthcare9020153

[39]Waghmare, V. K., & Kolekar, M. H. (2020). Brain Tumor Classification Using Deep Learning. In Internet of Things for Healthcare Technologies (Studies in Big Data, Vol. 73, pp. 155–175). Springer. https://doi.org/10.1007/978-981-15-4112-4_8

[40] Sharif, M. I., Khan, M. A., Alhussein, M., Aurangzeb, K., & Raza, M. (2022). A decision support system for multimodal brain tumor classification using deep learning. Complex & Intelligent Systems, 8, 3007–3020. https://doi.org/10.1007/s40747-021-00321-0

[41] Khan, M. A., Ashraf, I., Alhaisoni, M., Damaševičius, R., Scherer, R., Rehman, A., & Bukhari, S. A. C. (2020). Multimodal Brain Tumor Classification Using Deep Learning and Robust Feature Selection: A Machine Learning Application for Radiologists. Diagnostics, 10(8), 565. https://doi.org/10.3390/diagnostics10080565

[42] Mahmoud, A.; Awad, N. A.; Alsubaie, N.; Ansarullah, S. I.; Alqahtani, M. S.; Abbas, M.; Usman, M.; Soufiene, B. O.; Saber, A. (2023). Advanced Deep Learning Approaches for Accurate Brain Tumor Classification in Medical Imaging. Symmetry, 15(3), 571. https://doi.org/10.3390/sym15030571

[43] Kang, J.; Ullah, Z.; Gwak, J. (2021). MRI-Based Brain Tumor Classification Using Ensemble of Deep Features and Machine Learning Classifiers. Sensors, 21(6), 2222. https://doi.org/10.3390/s21062222

[44] Mohsen, H., El-Dahshan, E.-S. A., El-Horbaty, E.-S. M., & Salem, A.-B. M. (2018). Classification Using Deep Learning Neural Networks for Brain Tumors. Future Computing and Informatics Journal, 3(1), 68–71.

[45] Alqudah, A. M., Alquraan, H., Abu Qasmieh, I., Alqudah, A., & Al-Sharu, W. (2020). Brain Tumor Classification Using Deep Learning Technique — A Comparison between Cropped, Uncropped, and Segmented Lesion Images with Different Sizes. arXiv preprint arXiv:2001.08844.

[46] Talukder, M. A., Islam, M. M., Uddin, M. A., Akhter, A., Pramanik, M. A. J., Aryal, S., Almoyad, M. A. A., Hasan, K. F., & Moni, M. A. (2023). An efficient deep learning model to categorize brain tumor using reconstruction and fine-tuning. Expert Systems with Applications, 230, Article 120534. https://doi.org/10.1016/j.eswa.2023.120534

[48] Sadad, T., Rehman, A., Munir, A., Saba, T., Tariq, U., Ayesha, N., & Abbasi, R. (2021). Brain tumor detection and multi-classification using advanced deep learning techniques. Microscopy Research and Technique, 84(6), 1296–1308. https://doi.org/10.1002/jemt.23688

[49] Ayadi, W., Elhamzi, W., Charfi, I., & Atri, M. (2021). Deep CNN for Brain Tumor Classification. Neural Processing Letters, 53(1), 671–700. https://doi.org/10.1007/s11063-020-10398-2

[50] Mohsen, H., El-Dahshan, E.-S. A., El-Horbaty, E.-S. M., & Salem, A.-B. M. (2018). Classification Using Deep Learning Neural Networks for Brain Tumors. Future Computing and Informatics Journal, 3(1), 68–71. https://doi.org/10.1016/j.fcij.2017.12.001

[51] Alanazi, M. F.; Ali, M. U.; Hussain, S. J.; Zafar, A.; Mohatram, M.; Irfan, M.; AlRuwaili, R.; Alruwaili, M.; Ali, N. H.; Albarrak, A. M. (2022). Brain Tumor/Mass Classification Framework Using Magnetic-Resonance-Imaging-Based Isolated and Developed Transfer Deep-Learning Model. Sensors, 22(1), 372. https://doi.org/10.3390/s22010372

[52] Rehman, A., Naz, S., Razzak, M. I., Akram, F., & Imran, M. (2020). A Deep Learning-Based Framework for Automatic Brain Tumors Classification Using Transfer Learning. Circuits, Systems, and Signal Processing, 39, 757–775. https://doi.org/10.1007/s00034-019-01246-3

[53] Şahin, E., Özdemir, D., & Temurtaş, H. (2024). Multi-objective optimization of ViT architecture for efficient brain tumor classification. Biomedical Signal Processing and Control, 91, 105938. https://doi.org/10.1016/j.bspc.2023.105938

[51] Litjens, G., et al. (2017). A survey on deep learning in medical image analysis. Medical Image Analysis, 42, 60–88. https://doi.org/10.1016/j.media.2017.07.005

[52] Cheng, J. (2017). Brain Tumor Dataset. Figshare. https://figshare.com/articles/dataset/brain_tumor_dataset/1512427

[53] Menze, B. H., et al. (2015). The multimodal brain tumor image segmentation benchmark (BRATS). IEEE Transactions on Medical Imaging, 34(10), 1993–2024. https://doi.org/10.1109/TMI.2014.2377694

[54] Gusev, Y., et al. (2018). The REMBRANDT study, a large collection of genomic and clinical data from brain cancer patients. Scientific Data, 5, 180158. https://doi.org/10.1038/sdata.2018.158

[55] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.

[56] Litjens, G., et al. (2017). A survey on deep learning in medical image analysis. Medical Image Analysis, 42, 60–88.

[57] Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE TKDE, 22(10), 1345–1359.

[58] Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. Journal of Big Data, 3(1), 9.

[59] He, K., et al. (2016). Deep residual learning for image recognition. CVPR.

[60] Rajpurkar, P., et al. (2017). CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning. arXiv:1711.05225.

[61] Shin, H. C., et al. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE TMI, 35(5), 1285–1298.

[62] Dietterich, T. G. (2000). Ensemble methods in machine learning. Multiple Classifier Systems, 1–15.

[63] Ganaie, M. A., et al. (2022). Ensemble deep learning: A review. Engineering Applications of Artificial Intelligence, 115, 105151.

[64] Ju, C., Bibaut, A., & van der Laan, M. J. (2018). The relative performance of ensemble methods with deep convolutional neural networks for image classification. JMLR.

[65] https://www.researchgate.net/figure/CNN-architecture-our-CNN-consists-of-three-convolutional-layers-with-10-15-and-20_fig1_306081277

[66] https://www.researchgate.net/figure/A-schematic-illustration-of-the-DenseNet-121-architecture-82_fig5_334170752

[67] https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758/

[68] https://www.researchgate.net/figure/The-architecture-of-Vision-Transformers-18_fig1_372827201

Appendix A: Tools and Technologies Used

This appendix outlines the programming environments, frameworks, libraries, and computational platforms employed in the development of the brain tumor classification project. The tools were carefully selected to support efficient experimentation, scalability of models, and reproducibility of results across all stages of implementation, ranging from image preprocessing to ensemble evaluation.

1. Programming Language

- **Python 3.10**
  Python was chosen as the core programming language due to its extensive ecosystem for machine learning, simplicity of syntax, and integration with widely used deep learning frameworks.

2. Development Environment

- **Google Colab / Jupyter Notebook**
  Model development and training were primarily carried out using Google Colab, which provided access to GPU resources for high-performance computation. Jupyter Notebook was also used for exploratory data analysis (EDA) and for maintaining an organized workflow during experiments.

3. Deep Learning Frameworks

- **TensorFlow / Keras**
  These libraries were used for constructing and training the baseline CNN model. Keras offered a user-friendly API for model prototyping, while TensorFlow facilitated efficient GPU utilization.
- **PyTorch**
  PyTorch was used to implement and fine-tune advanced architectures such as ResNet50, DenseNet121, and Vision Transformers. Its dynamic computation graph and flexibility made it suitable for experimentation with custom ensemble strategies.

4. Machine Learning and Evaluation Libraries

- **Scikit-learn**
  Applied for generating performance metrics such as accuracy, precision, recall, F1-score, and confusion matrices. It was also used in stratified data splitting and evaluation of classification reports.
- **Imbalanced-learn (SMOTE)**
  Implemented to address dataset imbalance by generating synthetic samples for underrepresented tumor classes, ensuring fairer training outcomes.

5. Data Handling and Visualization

- **NumPy & Pandas**
  Utilized for array operations, data organization, and manipulation of metadata linked to MRI images.
- **Matplotlib & Seaborn**
  Used extensively to visualize dataset distributions, training/validation curves, and comparative model performances.

6. Image Processing Tools

- **OpenCV**
  Employed for resizing, denoising, and normalization of MRI slices before feeding them into the deep learning models.
- **Pillow (PIL)**
  Supplemented preprocessing tasks by enabling efficient image transformations and storage of augmented images.

7. Transfer Learning Architectures

- **ResNet50**
  Adopted for its residual connections, allowing effective gradient flow during training and improving feature extraction on MRI data.
- **DenseNet121**
  Leveraged for its densely connected layers that facilitated feature reuse and reduced vanishing gradient issues.

8. Transformer-based Architectures

- **Vision Transformer (DeiT-Small-Distilled)**
  Implemented as part of the experimental setup to model long-range dependencies within MRI scans using attention mechanisms. The model was trained with Adam optimizer and learning-rate scheduling to achieve competitive accuracy.

9. Ensemble Learning Strategy

- **Weighted Averaging**
  Predictions from CNN, ResNet50, DenseNet121, and Vision Transformer models were combined using a weighted averaging scheme to minimize class-specific misclassifications and improve robustness.

10. Hardware Acceleration

- **NVIDIA Tesla P100 GPUs (via Colab backend)**
  High-performance GPUs were used to accelerate the training of computationally demanding architectures, particularly the Vision Transformer and ensemble models.

The integration of these tools and platforms established a complete workflow that enabled us efficient image preprocessing, robust model training, comparative evaluation, and reproducibility of results.

Appendix B: Code Snippets

All code snippets in this appendix were developed by me.

1. Dataset Loading

The datasets were first uploaded to begin model development and training.

```
# Download latest version
path = kagglehub.dataset_download("ashkhagan/figshare-brain-tumor-dataset")

print("Path to dataset files:", path)
```

2.Data Integrity Check

The dataset was validated by performing several checks. First, the number of .mat files was counted to ensure all data were available. A preview of the first few files was displayed to confirm correct loading. Labels were then extracted from each file to verify consistency, and tumor masks were examined to identify cases with no tumor (empty masks) or invalid annotations (full masks).

```
# Explore the first .mat file
# Extract the brain scan image and pull out the brain scan image,
# Tumor type, and tumor mask, and displays two plots: one showing the brain scan
# And another with the tumor mask overlaid to show where the tumor is.
sample_mat = '/kaggle/input/figshare-brain-tumor-dataset/dataset/data/100.mat'
with h5py.File(sample_mat, 'r') as mat:
    cjdata = mat['cjdata']
    print(list(cjdata.keys()))
    image = np.array(cjdata['image'])
    label = np.array(cjdata['label'])[0][0]
    tumor_mask = np.array(cjdata['tumorMask'])

plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title(f'Tumor type: {label}')
plt.subplot(1, 2, 2)
plt.imshow(image, cmap='gray')
plt.imshow(tumor_mask, alpha=0.3)
plt.title('With tumor mask')
plt.show()
```

```
# Sets up the file list and tumor type labels for further processing,
# Like loading or analyzing brain tumor data, and tells you how many files are available.
input_dir = '/kaggle/input/figshare-brain-tumor-dataset/dataset/data/'
mat_files = [f for f in os.listdir(input_dir) if f.endswith('.mat')]
label_map = {1: "meningioma", 2: "glioma", 3: "pituitary"}
print(f"Found {len(mat_files)} .mat files.")


# checks the folder for brain tumor data files,
# counts them, and displays the names of the first five to give a quick look at the files available.
input_dir = '/kaggle/input/figshare-brain-tumor-dataset/dataset/data/'
mat_files = [f for f in os.listdir(input_dir) if f.endswith('.mat')]
print(f"Found {len(mat_files)} .mat files.")
print(mat_files[:5])  # Show first 5 file names
```

```
# loops through brain tumor data files (.mat), gets the tumor type number from each,
# stores them in a list, and prints how many labels were collected plus the first 10 labels.
labels = []

for filename in mat_files:
    with h5py.File(os.path.join(input_dir, filename), 'r') as mat:
        cjdata = mat['cjdata']
        label = int(np.array(cjdata['label'])[0][0])
        labels.append(label)

print(f"Collected labels for {len(labels)} images.")
print("Example labels:", labels[:10])

# We count how many times each tumor type appeared in the dataset and used the label_map to show the names instead of numbers

label_map = {1: "meningioma", 2: "glioma", 3: "pituitary"}
counts = Counter(labels)

for k in sorted(counts.keys()):
    print(f"{label_map[k]:<12}: {counts[k]}")
```

Table B.1. Dataset Structure Summary

Tumor Class	Label Value	Number of Samples	Notes on Data Integrity
Meningioma	1	708	Verified, no missing files,
Glioma	2	1426	Largest class, mild imbalance present
Pituitary	3	930	Verified, consistent annotations
Total	-------	3064	Dataset integrity checks completed

3.MRI Preprocessing Pipeline

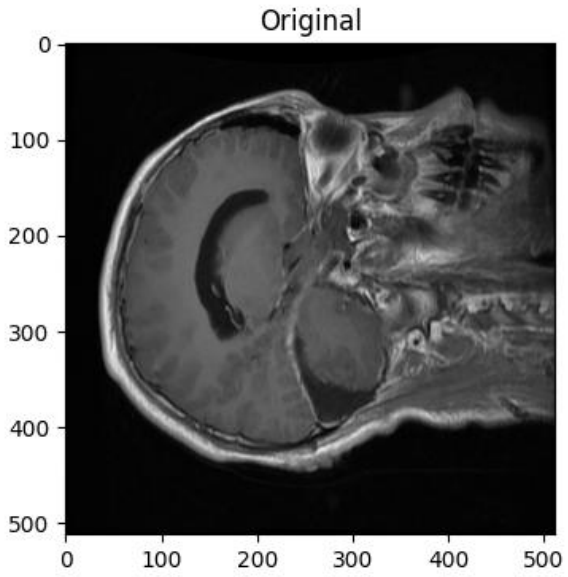
A custom preprocessing pipeline was applied to each slice, including denoising, contrast enhancement (CLAHE) , normalization, and resizing.

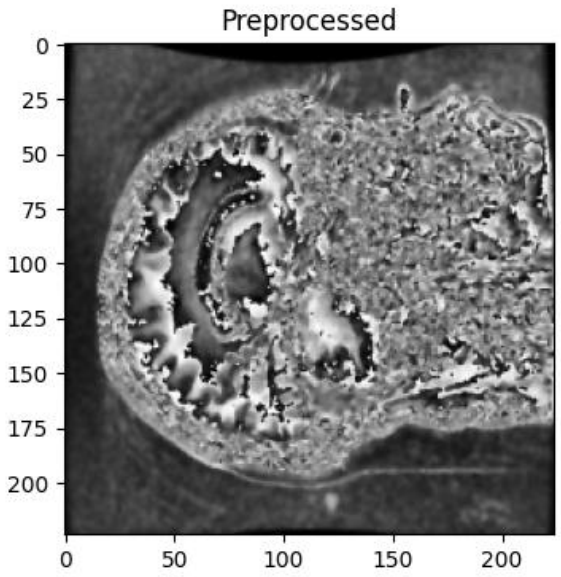
```python
# Apply CLAHE
def preprocess_mri_dl_soft(image):
    img = image.astype(np.uint8) if image.max() > 1 else img_as_ubyte(image)
    img = median_filter(img, size=3)
    img = cv2.bilateralFilter(img, d=9, sigmaColor=40, sigmaSpace=40)
    clahe = cv2.createCLAHE(clipLimit=1.0, tileGridSize=(8,8))
    img = clahe.apply(img)
    img = img.astype(np.float32)
    img = (img - img.min()) / (img.max() - img.min() + 1e-8)
    img = cv2.resize(img, (224, 224), interpolation=cv2.INTER_AREA)
    return img

def preprocess_mask(mask):
    mask = mask.astype(np.uint8)
    mask = cv2.resize(mask, (224, 224), interpolation=cv2.INTER_NEAREST)
    mask = (mask > 0).astype(np.uint8)
    return mask
```

23]:

```python
preprocessed_img_dir = './preprocessed_images'
preprocessed_mask_dir = './preprocessed_masks'
os.makedirs(preprocessed_img_dir, exist_ok=True)
os.makedirs(preprocessed_mask_dir, exist_ok=True)

meta = []
for filename in tqdm(mat_files):
    with h5py.File(os.path.join(input_dir, filename), 'r') as mat:
        cjdata = mat['cjdata']
        image = np.array(cjdata['image'])
        mask = np.array(cjdata['tumorMask'])
        label = int(np.array(cjdata['label'])[0][0])
        img_pre = preprocess_mri_dl_soft(image)
        mask_pre = preprocess_mask(mask)
        img_outfile = os.path.join(preprocessed_img_dir, filename.replace('.mat', '.npy'))
        mask_outfile = os.path.join(preprocessed_mask_dir, filename.replace('.mat', '.npy'))
        np.save(img_outfile, img_pre)
        np.save(mask_outfile, mask_pre)
        img_png_out = os.path.join(preprocessed_img_dir, filename.replace('.mat', '.png'))
        cv2.imwrite(img_png_out, (img_pre * 255).astype(np.uint8))
        meta.append({'filename': filename.replace('.mat', ''), 'label': label})

pd.DataFrame(meta).to_csv('preprocessed_labels.csv', index=False)
```

```
100%|██████████| 3064/3064 [04:47<00:00, 10.66it/s]
```





4.Train/Validation/Test Split with Stratification

The dataset was divided into a 70:15:15 ratio to maintain class balance. In the first step, 15% of the data was separated as the test set and kept untouched for unbiased evaluation. From the remaining 85%, the data was further divided into training (70%) and validation (15%) sets. Stratified sampling was applied to ensure that all three tumor types were proportionally represented across all splits.

```
from sklearn.model_selection import train_test_split

meta_df = pd.read_csv('preprocessed_labels.csv')

trainval_df, test_df = train_test_split(
    meta_df, test_size=0.15, stratify=meta_df['label'], random_state=42
)
train_df, val_df = train_test_split(
    trainval_df, test_size=0.15/0.85, stratify=trainval_df['label'], random_state=42
)

train_df.to_csv('train_split.csv', index=False)
val_df.to_csv('val_split.csv', index=False)
test_df.to_csv('test_split.csv', index=False)

print(f"Train: {len(train_df)}, Val: {len(val_df)}, Test: {len(test_df)}")
```

```
Train: 2144, Val: 460, Test: 460
```

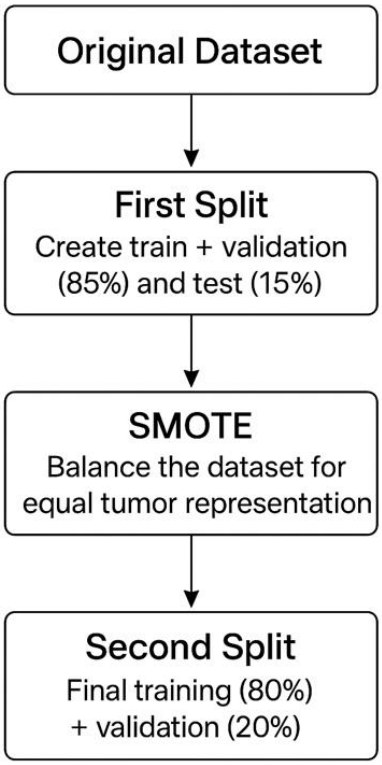
5.Class Balancing in Training Set

After applying SMOTE, we re-split the balanced training data into final training (80%) and validation (20%) sets to ensure proper model training and hyperparameter tuning. The test set from the initial split remained untouched to provide an unbiased final evaluation.

[33]:
```
import os
os.environ["OPENBLAS_NUM_THREADS"] = "1"
os.environ["OMP_NUM_THREADS"] = "1"
os.environ["MKL_NUM_THREADS"] = "1"
os.environ["NUMEXPR_NUM_THREADS"] = "1"

import warnings
warnings.filterwarnings("ignore")  # Suppress Python warnings

import pandas as pd
import numpy as np
import cv2
from imblearn.over_sampling import SMOTE
import contextlib

# Context manager to suppress C-level stderr (OpenBLAS warnings)
@contextlib.contextmanager
def suppress_stderr():
    import sys
    with open(os.devnull, 'w') as devnull:
        old_stderr = os.dup(2)
        os.dup2(devnull.fileno(), 2)
        try:
            yield
        finally:
            os.dup2(old_stderr, 2)

# Paths and loading code
train_csv = 'train_split.csv'
img_dir = './images_train'

df = pd.read_csv(train_csv)
X = []
y = []

for _, row in df.iterrows():
    img_path = os.path.join(img_dir, str(row['filename']) + '.png')
```

```
df = pd.read_csv(train_csv)
X = []
y = []

for _, row in df.iterrows():
    img_path = os.path.join(img_dir, str(row['filename']) + '.png')
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (224, 224))
    img = img.astype(np.float32) / 255.0
    X.append(img)
    y.append(row['label'] - 1)  # 0-based
X = np.array(X)[..., None]  # (N,224,224,1)
y = np.array(y)

print(X.shape, y.shape)

# SMOTE oversampling with suppressed OpenBLAS warnings
X_flat = X.reshape(len(X), -1)
sm = SMOTE(random_state=42)

with suppress_stderr():
    X_res, y_res = sm.fit_resample(X_flat, y)

print("After SMOTE:", X_res.shape, y_res.shape)
X_res = X_res.reshape(-1, 224, 224, 1)
```

```
(2144, 224, 224, 1) (2144,)
After SMOTE: (2994, 50176) (2994,)
```



6.CNN Model Implementaion

[35]:
```
import tensorflow as tf
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Input(shape=(224, 224, 1)),
    layers.Conv2D(32, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(128, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(len(np.unique(y_res)), activation='softmax'),
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
I0000 00:00:1755504781.858534      36 gpu_device.cc:2022] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 15513 MB memory:  -> de
vice: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	320
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_2 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 128)	0
flatten (Flatten)	(None, 86528)	0
dense (Dense)	(None, 128)	11,075,712

7.CNN Model Training

```
Total params: 11,168,771 (42.61 MB)
Trainable params: 11,168,771 (42.61 MB)
Non-trainable params: 0 (0.00 B)
```

[36]:
```
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau

callbacks = [
    ModelCheckpoint('best_model.h5', save_best_only=True, monitor='val_loss'),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6)
]

history = model.fit(
    X_train, y_train,
    epochs=50,
    batch_size=32,
    validation_data=(X_val, y_val),
    callbacks=callbacks,
    verbose=2
)
```

```
Epoch 1/50
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1755504786.259850     130 service.cc:148] XLA service 0x78dd34005480 initialized for platform CUDA (this does not guarantee that XLA w
ill be used). Devices:
I0000 00:00:1755504786.260720     130 service.cc:156]   StreamExecutor device (0): Tesla P100-PCIE-16GB, Compute Capability 6.0
I0000 00:00:1755504786.538410     130 cuda_dnn.cc:529] Loaded cuDNN version 90300
I0000 00:00:1755504790.004288     130 device_compiler.h:188] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the
process.
75/75 - 12s - 166ms/step - accuracy: 0.7582 - loss: 0.6019 - val_accuracy: 0.8414 - val_loss: 0.4384 - learning_rate: 0.0010
Epoch 2/50
75/75 - 2s - 31ms/step - accuracy: 0.9136 - loss: 0.2587 - val_accuracy: 0.8681 - val_loss: 0.3420 - learning_rate: 0.0010
Epoch 3/50
75/75 - 2s - 31ms/step - accuracy: 0.9553 - loss: 0.1289 - val_accuracy: 0.9098 - val_loss: 0.2998 - learning_rate: 0.0010
Epoch 4/50
75/75 - 2s - 26ms/step - accuracy: 0.9795 - loss: 0.0593 - val_accuracy: 0.9098 - val_loss: 0.4137 - learning_rate: 0.0010
Epoch 5/50
75/75 - 2s - 26ms/step - accuracy: 0.9942 - loss: 0.0181 - val_accuracy: 0.9115 - val_loss: 0.3970 - learning_rate: 0.0010
Epoch 6/50
```

```
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.3423e-04 - val_accuracy: 0.9182 - val_loss: 0.5398 - learning_rate: 3.1250e-05
Epoch 22/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.3006e-04 - val_accuracy: 0.9182 - val_loss: 0.5408 - learning_rate: 1.5625e-05
Epoch 23/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.2807e-04 - val_accuracy: 0.9182 - val_loss: 0.5424 - learning_rate: 1.5625e-05
Epoch 24/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.2570e-04 - val_accuracy: 0.9182 - val_loss: 0.5430 - learning_rate: 1.5625e-05
Epoch 25/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.2370e-04 - val_accuracy: 0.9182 - val_loss: 0.5436 - learning_rate: 7.8125e-06
Epoch 26/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.2275e-04 - val_accuracy: 0.9182 - val_loss: 0.5441 - learning_rate: 7.8125e-06
Epoch 27/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.2156e-04 - val_accuracy: 0.9182 - val_loss: 0.5446 - learning_rate: 7.8125e-06
Epoch 28/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.2062e-04 - val_accuracy: 0.9182 - val_loss: 0.5448 - learning_rate: 3.9063e-06
Epoch 29/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1999e-04 - val_accuracy: 0.9182 - val_loss: 0.5451 - learning_rate: 3.9063e-06
Epoch 30/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1943e-04 - val_accuracy: 0.9182 - val_loss: 0.5453 - learning_rate: 3.9063e-06
Epoch 31/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1896e-04 - val_accuracy: 0.9182 - val_loss: 0.5455 - learning_rate: 1.9531e-06
Epoch 32/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1864e-04 - val_accuracy: 0.9182 - val_loss: 0.5455 - learning_rate: 1.9531e-06
Epoch 33/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1838e-04 - val_accuracy: 0.9182 - val_loss: 0.5459 - learning_rate: 1.9531e-06
Epoch 34/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1809e-04 - val_accuracy: 0.9182 - val_loss: 0.5458 - learning_rate: 1.0000e-06
Epoch 35/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1794e-04 - val_accuracy: 0.9182 - val_loss: 0.5459 - learning_rate: 1.0000e-06
Epoch 36/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1778e-04 - val_accuracy: 0.9182 - val_loss: 0.5460 - learning_rate: 1.0000e-06
Epoch 37/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1762e-04 - val_accuracy: 0.9182 - val_loss: 0.5461 - learning_rate: 1.0000e-06
Epoch 38/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1745e-04 - val_accuracy: 0.9182 - val_loss: 0.5461 - learning_rate: 1.0000e-06
Epoch 39/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1729e-04 - val_accuracy: 0.9182 - val_loss: 0.5462 - learning_rate: 1.0000e-06
Epoch 40/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1710e-04 - val_accuracy: 0.9182 - val_loss: 0.5464 - learning_rate: 1.0000e-06
Epoch 41/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1692e-04 - val_accuracy: 0.9182 - val_loss: 0.5464 - learning_rate: 1.0000e-06
Epoch 42/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1675e-04 - val_accuracy: 0.9182 - val_loss: 0.5465 - learning_rate: 1.0000e-06
Epoch 43/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1655e-04 - val_accuracy: 0.9182 - val_loss: 0.5466 - learning_rate: 1.0000e-06
```

```
75/75 - 2s - 31ms/step - accuracy: 0.9553 - loss: 0.1289 - val_accuracy: 0.9098 - val_loss: 0.2998 - learning_rate: 0.0010
Epoch 4/50
75/75 - 2s - 26ms/step - accuracy: 0.9795 - loss: 0.0593 - val_accuracy: 0.9098 - val_loss: 0.4137 - learning_rate: 0.0010
Epoch 5/50
75/75 - 2s - 26ms/step - accuracy: 0.9942 - loss: 0.0181 - val_accuracy: 0.9115 - val_loss: 0.3970 - learning_rate: 0.0010
Epoch 6/50
75/75 - 2s - 26ms/step - accuracy: 0.9946 - loss: 0.0209 - val_accuracy: 0.9098 - val_loss: 0.4535 - learning_rate: 0.0010
Epoch 7/50
75/75 - 2s - 26ms/step - accuracy: 0.9971 - loss: 0.0101 - val_accuracy: 0.9232 - val_loss: 0.4301 - learning_rate: 5.0000e-04
Epoch 8/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 0.0011 - val_accuracy: 0.9232 - val_loss: 0.4439 - learning_rate: 5.0000e-04
Epoch 9/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 4.7592e-04 - val_accuracy: 0.9182 - val_loss: 0.4886 - learning_rate: 5.0000e-04
Epoch 10/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 3.2519e-04 - val_accuracy: 0.9182 - val_loss: 0.4966 - learning_rate: 2.5000e-04
Epoch 11/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 2.7907e-04 - val_accuracy: 0.9182 - val_loss: 0.5044 - learning_rate: 2.5000e-04
Epoch 12/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 2.4145e-04 - val_accuracy: 0.9182 - val_loss: 0.5154 - learning_rate: 2.5000e-04
Epoch 13/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 2.1423e-04 - val_accuracy: 0.9182 - val_loss: 0.5196 - learning_rate: 1.2500e-04
Epoch 14/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.9853e-04 - val_accuracy: 0.9182 - val_loss: 0.5253 - learning_rate: 1.2500e-04
Epoch 15/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.8352e-04 - val_accuracy: 0.9182 - val_loss: 0.5298 - learning_rate: 1.2500e-04
Epoch 16/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.7026e-04 - val_accuracy: 0.9182 - val_loss: 0.5319 - learning_rate: 6.2500e-05
Epoch 17/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.6126e-04 - val_accuracy: 0.9182 - val_loss: 0.5337 - learning_rate: 6.2500e-05
Epoch 18/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.5167e-04 - val_accuracy: 0.9182 - val_loss: 0.5361 - learning_rate: 6.2500e-05
Epoch 19/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.4357e-04 - val_accuracy: 0.9182 - val_loss: 0.5367 - learning_rate: 3.1250e-05
Epoch 20/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.3854e-04 - val_accuracy: 0.9182 - val_loss: 0.5380 - learning_rate: 3.1250e-05
Epoch 21/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.3423e-04 - val_accuracy: 0.9182 - val_loss: 0.5398 - learning_rate: 3.1250e-05
Epoch 22/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.3006e-04 - val_accuracy: 0.9182 - val_loss: 0.5408 - learning_rate: 1.5625e-05
Epoch 23/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.2807e-04 - val_accuracy: 0.9182 - val_loss: 0.5424 - learning_rate: 1.5625e-05
Epoch 24/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.2570e-04 - val_accuracy: 0.9182 - val_loss: 0.5430 - learning_rate: 1.5625e-05
Epoch 25/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.2370e-04 - val_accuracy: 0.9182 - val_loss: 0.5436 - learning_rate: 7.8125e-06
```

```
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1655e-04 - val_accuracy: 0.9182 - val_loss: 0.5466 - learning_rate: 1.0000e-06
Epoch 44/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1637e-04 - val_accuracy: 0.9182 - val_loss: 0.5467 - learning_rate: 1.0000e-06
Epoch 45/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1616e-04 - val_accuracy: 0.9182 - val_loss: 0.5469 - learning_rate: 1.0000e-06
Epoch 46/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1596e-04 - val_accuracy: 0.9182 - val_loss: 0.5469 - learning_rate: 1.0000e-06
Epoch 47/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1575e-04 - val_accuracy: 0.9182 - val_loss: 0.5470 - learning_rate: 1.0000e-06
Epoch 48/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1555e-04 - val_accuracy: 0.9182 - val_loss: 0.5472 - learning_rate: 1.0000e-06
Epoch 49/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1535e-04 - val_accuracy: 0.9182 - val_loss: 0.5473 - learning_rate: 1.0000e-06
Epoch 50/50
75/75 - 2s - 26ms/step - accuracy: 1.0000 - loss: 1.1510e-04 - val_accuracy: 0.9182 - val_loss: 0.5474 - learning_rate: 1.0000e-06
```

8.CNN Model Evaluation

[37]:

```python
import matplotlib.pyplot as plt

# Accuracy
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

[40]:
```
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc, precision_recall_curve, average_precision_score
import matplotlib.pyplot as plt

# Get predicted probabilities for validation set
y_pred_probs = model.predict(X_val)   # shape: (N_samples, N_classes)
n_classes = y_pred_probs.shape[1]

# Binarize the ground truth labels
y_true_bin = label_binarize(y_true, classes=range(n_classes))

# ROC Curve
plt.figure(figsize=(8, 6))
for i in range(n_classes):
    fpr, tpr, _ = roc_curve(y_true_bin[:, i], y_pred_probs[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{class_names[i]} (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.title('Multi-class ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()

# Precision-Recall Curve
plt.figure(figsize=(8, 6))
for i in range(n_classes):
    precision, recall, _ = precision_recall_curve(y_true_bin[:, i], y_pred_probs[:, i])
    ap = average_precision_score(y_true_bin[:, i], y_pred_probs[:, i])
    plt.plot(recall, precision, label=f'{class_names[i]} (AP = {ap:.2f})')
plt.title('Multi-class Precision-Recall Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend()
plt.show()
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
```

[39]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay

# Assuming you have a trained model and X_val, y_val
# For a softmax model:
y_pred_proba = model.predict(X_val)
y_pred = np.argmax(y_pred_proba, axis=1)
y_true = y_val  # y_val from your train_test_split

# List your class names in correct order, for example:
class_names = ['Normal', 'Benign', 'Malignant']

print(classification_report(y_true, y_pred, target_names=class_names))

cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step
              precision    recall  f1-score   support

      Normal       0.85      0.94      0.89       199
      Benign       0.95      0.83      0.89       200
   Malignant       0.97      0.98      0.98       200

    accuracy                           0.92       599
   macro avg       0.92      0.92      0.92       599
weighted avg       0.92      0.92      0.92       599
```

9.DenseNet121 Model

[42]:

```
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout, Input
from tensorflow.keras.models import Model

n_classes = len(class_names)  # or set this explicitly

# Transfer learning base
base_model = DenseNet121(
    include_top=False,
    weights='imagenet',
    input_shape=(224, 224, 3)
)
base_model.trainable = False  # Freeze for feature extraction (unfreeze later for fine-tuning)

inputs = Input(shape=(224, 224, 3))
x = base_model(inputs, training=False)
x = GlobalAveragePooling2D()(x)
x = Dropout(0.3)(x)
outputs = Dense(n_classes, activation='softmax')(x)
model = Model(inputs, outputs)

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h
5
 29084464/29084464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Model: "functional_1"
```


```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h
5
 29084464/29084464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Model: "functional_1"
```


Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 224, 224, 3)	0
densenet121 (Functional)	(None, 7, 7, 1024)	7,037,504
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1024)	0
dropout (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 3)	3,075

Total params: 7,040,579 (26.86 MB)
Trainable params: 3,075 (12.01 KB)
Non-trainable params: 7,037,504 (26.85 MB)

9.DenseNet121 Model Training

[43]:

```
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau

callbacks = [
    ModelCheckpoint('best_densenet121.h5', save_best_only=True, monitor='val_loss'),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6)
]

history = model.fit(
    X_train_rgb, y_train,
    epochs=50,   # Transfer learning can converge quickly (can go longer if needed)
    batch_size=32,
    validation_data=(X_val_rgb, y_val),
    callbacks=callbacks,
    verbose=2
)
```

```
Epoch 1/50
75/75 - 70s - 930ms/step - accuracy: 0.4877 - loss: 1.1005 - val_accuracy: 0.7062 - val_loss: 0.7332 - learning_rate: 0.0010
Epoch 2/50
75/75 - 5s - 66ms/step - accuracy: 0.6543 - loss: 0.7897 - val_accuracy: 0.7780 - val_loss: 0.5766 - learning_rate: 0.0010
Epoch 3/50
75/75 - 5s - 67ms/step - accuracy: 0.7382 - loss: 0.6468 - val_accuracy: 0.8097 - val_loss: 0.5126 - learning_rate: 0.0010
Epoch 4/50
75/75 - 5s - 66ms/step - accuracy: 0.7674 - loss: 0.5851 - val_accuracy: 0.8114 - val_loss: 0.4827 - learning_rate: 0.0010
Epoch 5/50
75/75 - 5s - 67ms/step - accuracy: 0.7766 - loss: 0.5567 - val_accuracy: 0.8230 - val_loss: 0.4673 - learning_rate: 0.0010
Epoch 6/50
75/75 - 5s - 66ms/step - accuracy: 0.7741 - loss: 0.5518 - val_accuracy: 0.8631 - val_loss: 0.4380 - learning_rate: 0.0010
Epoch 7/50
75/75 - 5s - 66ms/step - accuracy: 0.8100 - loss: 0.4929 - val_accuracy: 0.8397 - val_loss: 0.4318 - learning_rate: 0.0010
Epoch 8/50
75/75 - 5s - 66ms/step - accuracy: 0.8000 - loss: 0.4990 - val_accuracy: 0.8464 - val_loss: 0.4195 - learning_rate: 0.0010
Epoch 9/50
75/75 - 5s - 67ms/step - accuracy: 0.8088 - loss: 0.4867 - val_accuracy: 0.8564 - val_loss: 0.4010 - learning_rate: 0.0010
```

```
75/75 - 5s - 66ms/step - accuracy: 0.7741 - loss: 0.5518 - val_accuracy: 0.8631 - val_loss: 0.4380 - learning_rate: 0.0010
Epoch 7/50
75/75 - 5s - 66ms/step - accuracy: 0.8100 - loss: 0.4929 - val_accuracy: 0.8397 - val_loss: 0.4318 - learning_rate: 0.0010
Epoch 8/50
75/75 - 5s - 66ms/step - accuracy: 0.8000 - loss: 0.4990 - val_accuracy: 0.8464 - val_loss: 0.4195 - learning_rate: 0.0010
Epoch 9/50
75/75 - 5s - 67ms/step - accuracy: 0.8088 - loss: 0.4867 - val_accuracy: 0.8564 - val_loss: 0.4010 - learning_rate: 0.0010
Epoch 10/50
75/75 - 6s - 76ms/step - accuracy: 0.8288 - loss: 0.4609 - val_accuracy: 0.8698 - val_loss: 0.3957 - learning_rate: 0.0010
Epoch 11/50
75/75 - 5s - 66ms/step - accuracy: 0.8217 - loss: 0.4671 - val_accuracy: 0.8614 - val_loss: 0.3925 - learning_rate: 0.0010
Epoch 12/50
75/75 - 5s - 67ms/step - accuracy: 0.8225 - loss: 0.4547 - val_accuracy: 0.8681 - val_loss: 0.3795 - learning_rate: 0.0010
Epoch 13/50
75/75 - 5s - 67ms/step - accuracy: 0.8255 - loss: 0.4533 - val_accuracy: 0.8648 - val_loss: 0.3764 - learning_rate: 0.0010
Epoch 14/50
75/75 - 4s - 59ms/step - accuracy: 0.8242 - loss: 0.4460 - val_accuracy: 0.8781 - val_loss: 0.3809 - learning_rate: 0.0010
Epoch 15/50
75/75 - 5s - 67ms/step - accuracy: 0.8313 - loss: 0.4430 - val_accuracy: 0.8698 - val_loss: 0.3744 - learning_rate: 0.0010
Epoch 16/50
75/75 - 4s - 58ms/step - accuracy: 0.8263 - loss: 0.4351 - val_accuracy: 0.8581 - val_loss: 0.3797 - learning_rate: 0.0010
Epoch 17/50
75/75 - 5s - 66ms/step - accuracy: 0.8342 - loss: 0.4249 - val_accuracy: 0.8715 - val_loss: 0.3647 - learning_rate: 0.0010
Epoch 18/50
75/75 - 5s - 67ms/step - accuracy: 0.8255 - loss: 0.4453 - val_accuracy: 0.8781 - val_loss: 0.3605 - learning_rate: 0.0010
Epoch 19/50
75/75 - 4s - 59ms/step - accuracy: 0.8322 - loss: 0.4363 - val_accuracy: 0.8765 - val_loss: 0.3631 - learning_rate: 0.0010
Epoch 20/50
75/75 - 4s - 59ms/step - accuracy: 0.8221 - loss: 0.4357 - val_accuracy: 0.8548 - val_loss: 0.3768 - learning_rate: 0.0010
Epoch 21/50
75/75 - 4s - 59ms/step - accuracy: 0.8409 - loss: 0.4229 - val_accuracy: 0.8748 - val_loss: 0.3614 - learning_rate: 0.0010
Epoch 22/50
75/75 - 5s - 66ms/step - accuracy: 0.8476 - loss: 0.4043 - val_accuracy: 0.8848 - val_loss: 0.3492 - learning_rate: 5.0000e-04
Epoch 23/50
75/75 - 4s - 59ms/step - accuracy: 0.8309 - loss: 0.4250 - val_accuracy: 0.8765 - val_loss: 0.3503 - learning_rate: 5.0000e-04
Epoch 24/50
75/75 - 4s - 59ms/step - accuracy: 0.8447 - loss: 0.3949 - val_accuracy: 0.8831 - val_loss: 0.3522 - learning_rate: 5.0000e-04
Epoch 25/50
75/75 - 4s - 59ms/step - accuracy: 0.8342 - loss: 0.4157 - val_accuracy: 0.8798 - val_loss: 0.3538 - learning_rate: 5.0000e-04
Epoch 26/50
75/75 - 5s - 66ms/step - accuracy: 0.8447 - loss: 0.4008 - val_accuracy: 0.8865 - val_loss: 0.3463 - learning_rate: 2.5000e-04
Epoch 27/50
75/75 - 4s - 59ms/step - accuracy: 0.8447 - loss: 0.4043 - val_accuracy: 0.8898 - val_loss: 0.3470 - learning_rate: 2.5000e-04
Epoch 28/50
75/75 - 4s - 59ms/step - accuracy: 0.8489 - loss: 0.4117 - val_accuracy: 0.8865 - val_loss: 0.3493 - learning_rate: 2.5000e-04
Epoch 29/50
75/75 - 5s - 66ms/step - accuracy: 0.8359 - loss: 0.4081 - val_accuracy: 0.8898 - val_loss: 0.3445 - learning_rate: 2.5000e-04
Epoch 30/50
75/75 - 5s - 66ms/step - accuracy: 0.8476 - loss: 0.4091 - val_accuracy: 0.8831 - val_loss: 0.3443 - learning_rate: 2.5000e-04
Epoch 31/50
75/75 - 5s - 67ms/step - accuracy: 0.8438 - loss: 0.4005 - val_accuracy: 0.8865 - val_loss: 0.3427 - learning_rate: 2.5000e-04
Epoch 32/50
75/75 - 5s - 67ms/step - accuracy: 0.8426 - loss: 0.4039 - val_accuracy: 0.8881 - val_loss: 0.3420 - learning_rate: 2.5000e-04
Epoch 33/50
75/75 - 4s - 59ms/step - accuracy: 0.8501 - loss: 0.3994 - val_accuracy: 0.8881 - val_loss: 0.3444 - learning_rate: 2.5000e-04
Epoch 34/50
75/75 - 5s - 67ms/step - accuracy: 0.8555 - loss: 0.3909 - val_accuracy: 0.8831 - val_loss: 0.3412 - learning_rate: 2.5000e-04
Epoch 35/50
75/75 - 5s - 66ms/step - accuracy: 0.8384 - loss: 0.4098 - val_accuracy: 0.8848 - val_loss: 0.3411 - learning_rate: 2.5000e-04
Epoch 36/50
75/75 - 4s - 59ms/step - accuracy: 0.8372 - loss: 0.4073 - val_accuracy: 0.8798 - val_loss: 0.3445 - learning_rate: 2.5000e-04
Epoch 37/50
75/75 - 5s - 66ms/step - accuracy: 0.8509 - loss: 0.3883 - val_accuracy: 0.8848 - val_loss: 0.3406 - learning_rate: 2.5000e-04
Epoch 38/50
75/75 - 4s - 59ms/step - accuracy: 0.8447 - loss: 0.3986 - val_accuracy: 0.8815 - val_loss: 0.3426 - learning_rate: 2.5000e-04
Epoch 39/50
75/75 - 4s - 59ms/step - accuracy: 0.8518 - loss: 0.3948 - val_accuracy: 0.8848 - val_loss: 0.3441 - learning_rate: 2.5000e-04
Epoch 40/50
75/75 - 4s - 59ms/step - accuracy: 0.8330 - loss: 0.4134 - val_accuracy: 0.8831 - val_loss: 0.3437 - learning_rate: 2.5000e-04
Epoch 41/50
75/75 - 5s - 67ms/step - accuracy: 0.8468 - loss: 0.3908 - val_accuracy: 0.8848 - val_loss: 0.3404 - learning_rate: 1.2500e-04
Epoch 42/50
75/75 - 4s - 59ms/step - accuracy: 0.8518 - loss: 0.3832 - val_accuracy: 0.8815 - val_loss: 0.3410 - learning_rate: 1.2500e-04
Epoch 43/50
75/75 - 5s - 67ms/step - accuracy: 0.8434 - loss: 0.3906 - val_accuracy: 0.8848 - val_loss: 0.3392 - learning_rate: 1.2500e-04
Epoch 44/50
75/75 - 4s - 59ms/step - accuracy: 0.8530 - loss: 0.3857 - val_accuracy: 0.8881 - val_loss: 0.3396 - learning_rate: 1.2500e-04
Epoch 45/50
75/75 - 4s - 59ms/step - accuracy: 0.8501 - loss: 0.4001 - val_accuracy: 0.8865 - val_loss: 0.3395 - learning_rate: 1.2500e-04
Epoch 46/50
75/75 - 4s - 59ms/step - accuracy: 0.8522 - loss: 0.3906 - val_accuracy: 0.8881 - val_loss: 0.3403 - learning_rate: 1.2500e-04
Epoch 47/50
75/75 - 5s - 66ms/step - accuracy: 0.8501 - loss: 0.3855 - val_accuracy: 0.8898 - val_loss: 0.3378 - learning_rate: 6.2500e-05
Epoch 48/50
75/75 - 5s - 66ms/step - accuracy: 0.8463 - loss: 0.3908 - val_accuracy: 0.8881 - val_loss: 0.3377 - learning_rate: 6.2500e-05
Epoch 49/50
75/75 - 5s - 67ms/step - accuracy: 0.8543 - loss: 0.3968 - val_accuracy: 0.8865 - val_loss: 0.3370 - learning_rate: 6.2500e-05
Epoch 50/50
```

10.DenseNet121 Model Finetuning

[44]:
```
# Unfreeze layers for fine-tuning
base_model.trainable = True
for layer in base_model.layers[:-20]:
     layer.trainable = False

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

history_ft = model.fit(
    X_train_rgb, y_train,
    epochs=50,       # Fine-tuning, use smaller learning rate and fewer epochs
    batch_size=32,
    validation_data=(X_val_rgb, y_val),
    callbacks=callbacks,
    verbose=2
)
```

```
Epoch 1/50
```

10.DenseNet121 Model Evaluation

[45]:
```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Predict
y_pred_probs = model.predict(X_val_rgb)
y_pred = np.argmax(y_pred_probs, axis=1)

# Classification report
print("Classification Report:")
print(classification_report(y_val, y_pred, target_names=class_names))

# Confusion matrix
cm = confusion_matrix(y_val, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 25s 737ms/step
Classification Report:
              precision    recall  f1-score   support

      Normal       0.91      0.97      0.94       199
      Benign       0.98      0.92      0.95       200
   Malignant       0.98      0.99      0.99       200

    accuracy                           0.96       599
   macro avg       0.96      0.96      0.96       599
weighted avg       0.96      0.96      0.96       599

                 Confusion Matrix
```

11. ResNet50 Model Training

```
[9]:
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout, Input
from tensorflow.keras.optimizers import Adam

num_classes = len(class_names)

inputs = Input(shape=(224, 224, 3))
base_model = ResNet50(include_top=False, weights='imagenet', input_tensor=inputs)
base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.4)(x)
outputs = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=inputs, outputs=outputs)

model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    X_train_rgb, y_train,
    epochs=50,
    batch_size=16,
    validation_data=(X_val_rgb, y_val),
    verbose=2
)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/50
150/150 - 30s - 198ms/step - accuracy: 0.3278 - loss: 1.3221 - val_accuracy: 0.2604 - val_loss: 1.1166
Epoch 2/50
150/150 - 5s - 34ms/step - accuracy: 0.3211 - loss: 1.2715 - val_accuracy: 0.3055 - val_loss: 1.0993
```

12. ResNet50 Model Finetuning

```
[50]:
base_model.trainable = True
for layer in base_model.layers[:-20]:
     layer.trainable = False
model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
history_finetune = model.fit(
    X_train_rgb, y_train,
    epochs=50,
    batch_size=16,
    validation_data=(X_val_rgb, y_val),
    verbose=2
)
```

```
Epoch 1/50
150/150 - 38s - 254ms/step - accuracy: 0.7783 - loss: 0.6594 - val_accuracy: 0.4858 - val_loss: 1.0403
```

13. ResNet50 Model Evaluation

[51]:
```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Predict
y_pred_probs = model.predict(X_val_rgb)
y_pred = np.argmax(y_pred_probs, axis=1)

# Classification report
print("Classification Report:")
print(classification_report(y_val, y_pred, target_names=class_names))

# Confusion matrix
cm = confusion_matrix(y_val, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 12s 386ms/step
Classification Report:
              precision    recall  f1-score   support

      Normal       0.95      0.78      0.86       199
      Benign       0.83      0.94      0.88       200
   Malignant       0.95      0.98      0.96       200

    accuracy                           0.90       599
   macro avg       0.91      0.90      0.90       599
weighted avg       0.91      0.90      0.90       599
```

14. Vision Transformer Model

[58]:
```
# deit_small_distilled_patch16_224 is the distilled variant (often better in low-data)
model = timm.create_model('deit_small_distilled_patch16_224', pretrained=True, num_classes=num_classes)
model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.AdamW(model.parameters(), lr=lr, weight_decay=weight_decay)
# Cosine with warmup
total_steps = epochs * len(train_loader)
warmup_steps = int(0.05 * total_steps)

def lr_lambda(step):
    if step < warmup_steps:
        return float(step) / float(max(1, warmup_steps))
    # cosine decay after warmup
    progress = float(step - warmup_steps) / float(max(1, total_steps - warmup_steps))
    return 0.5 * (1.0 + np.cos(np.pi * progress))

scheduler = torch.optim.lr_scheduler.LambdaLR(optimizer, lr_lambda)
scaler = torch.cuda.amp.GradScaler(enabled=torch.cuda.is_available())
```

```
Loading widget...
```

15. Vision Transformer Training

```python
from collections import defaultdict
best_val_acc = 0.0
best_path = 'best_deit_small.pth'

def run_epoch(loader, train_mode=True):
    if train_mode:
        model.train()
    else:
        model.eval()

    running_loss, correct, total = 0.0, 0, 0

    for imgs, labels in loader:
        imgs, labels = imgs.to(device, non_blocking=True), labels.to(device, non_blocking=True)

        with torch.cuda.amp.autocast(enabled=torch.cuda.is_available()):
            outputs = model(imgs)
            loss = criterion(outputs, labels)

        if train_mode:
            optimizer.zero_grad(set_to_none=True)
            scaler.scale(loss).backward()
            scaler.step(optimizer)
            scaler.update()
            scheduler.step()

        running_loss += loss.item() * imgs.size(0)
        preds = outputs.argmax(dim=1)
        correct += (preds == labels).sum().item()
        total += imgs.size(0)

    avg_loss = running_loss / total
    acc = correct / total
    return avg_loss, acc

history = defaultdict(list)

for epoch in range(1, epochs+1):
    tr_loss, tr_acc = run_epoch(train_loader, train_mode=True)
    val_loss, val_acc = run_epoch(val_loader,   train_mode=False)
```

```python
    history['train_loss'].append(tr_loss)
    history['train_acc'].append(tr_acc)
    history['val_loss'].append(val_loss)
    history['val_acc'].append(val_acc)

    print(f"Epoch {epoch:02d}/{epochs} | "
          f"Train: loss {tr_loss:.4f}, acc {tr_acc:.4f} | "
          f"Val: loss {val_loss:.4f}, acc {val_acc:.4f} | "
          f"lr {scheduler.get_last_lr()[0]:.3e}")

    if val_acc > best_val_acc:
        best_val_acc = val_acc
        torch.save(model.state_dict(), best_path)
        print(f"  ✓ Saved best model to {best_path} (val_acc={best_val_acc:.4f})")
```

```
Epoch 01/75 | Train: loss 0.6944, acc 0.7015 | Val: loss 0.4193, acc 0.8348 | lr 8.008e-05
  ✓ Saved best model to best_deit_small.pth (val_acc=0.8348)
Epoch 02/75 | Train: loss 0.4084, acc 0.8284 | Val: loss 0.4438, acc 0.8261 | lr 1.602e-04
Epoch 03/75 | Train: loss 0.3873, acc 0.8428 | Val: loss 0.4433, acc 0.8304 | lr 2.402e-04
Epoch 04/75 | Train: loss 0.3635, acc 0.8465 | Val: loss 0.2870, acc 0.8913 | lr 3.000e-04
  ✓ Saved best model to best_deit_small.pth (val_acc=0.8913)
Epoch 05/75 | Train: loss 0.3559, acc 0.8498 | Val: loss 0.2677, acc 0.8913 | lr 2.998e-04
Epoch 06/75 | Train: loss 0.2834, acc 0.8848 | Val: loss 0.2600, acc 0.8674 | lr 2.993e-04
Epoch 07/75 | Train: loss 0.3114, acc 0.8694 | Val: loss 0.2759, acc 0.9000 | lr 2.985e-04
  ✓ Saved best model to best_deit_small.pth (val_acc=0.9000)
Epoch 08/75 | Train: loss 0.2608, acc 0.8909 | Val: loss 0.2498, acc 0.8957 | lr 2.974e-04
Epoch 09/75 | Train: loss 0.2570, acc 0.8946 | Val: loss 0.2289, acc 0.9043 | lr 2.960e-04
  ✓ Saved best model to best_deit_small.pth (val_acc=0.9043)
Epoch 10/75 | Train: loss 0.2318, acc 0.9104 | Val: loss 0.2083, acc 0.9109 | lr 2.943e-04
  ✓ Saved best model to best_deit_small.pth (val_acc=0.9109)
Epoch 11/75 | Train: loss 0.2425, acc 0.8974 | Val: loss 0.2189, acc 0.9065 | lr 2.924e-04
Epoch 12/75 | Train: loss 0.2284, acc 0.9142 | Val: loss 0.2339, acc 0.9261 | lr 2.902e-04
  ✓ Saved best model to best_deit_small.pth (val_acc=0.9261)
Epoch 13/75 | Train: loss 0.1994, acc 0.9249 | Val: loss 0.2021, acc 0.9130 | lr 2.877e-04
Epoch 14/75 | Train: loss 0.2060, acc 0.9184 | Val: loss 0.2507, acc 0.8978 | lr 2.849e-04
Epoch 15/75 | Train: loss 0.2076, acc 0.9151 | Val: loss 0.2142, acc 0.9196 | lr 2.819e-04
Epoch 16/75 | Train: loss 0.1836, acc 0.9254 | Val: loss 0.2273, acc 0.9217 | lr 2.786e-04
Epoch 17/75 | Train: loss 0.2064, acc 0.9202 | Val: loss 0.2101, acc 0.9196 | lr 2.751e-04
Epoch 18/75 | Train: loss 0.1629, acc 0.9305 | Val: loss 0.1779, acc 0.9261 | lr 2.713e-04
Epoch 19/75 | Train: loss 0.1876, acc 0.9230 | Val: loss 0.2112, acc 0.9130 | lr 2.673e-04
Epoch 20/75 | Train: loss 0.1800, acc 0.9338 | Val: loss 0.2132, acc 0.9087 | lr 2.631e-04
Epoch 21/75 | Train: loss 0.1468, acc 0.9394 | Val: loss 0.1794, acc 0.9304 | lr 2.587e-04
```

13. Vision Transformer Evaluation

[61]:
```python
# Load best weights
model.load_state_dict(torch.load(best_path, map_location=device))
model.eval()

def predict_loader(loader):
    all_logits = []
    all_targets = []
    with torch.no_grad():
        for imgs, labels in loader:
            imgs = imgs.to(device, non_blocking=True)
            logits = model(imgs).cpu()
            all_logits.append(logits)
            all_targets.append(labels)
    logits = torch.cat(all_logits)
    targets = torch.cat(all_targets)
    probs = logits.softmax(dim=1).numpy()
    preds = probs.argmax(axis=1)
    return preds, probs, targets.numpy()

# Choose which split to report (val or test)
use_test = True   # set False to evaluate on validation set
loader = test_loader if use_test else val_loader

y_pred, y_prob, y_true = predict_loader(loader)

print(classification_report(y_true, y_pred, target_names=class_names, digits=4))

cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix (DeiT-Small)")
plt.show()
```

```
              precision    recall  f1-score   support

  meningioma     0.9612    0.9340    0.9474       106
      glioma     0.9860    0.9860    0.9860       214
   pituitary     0.9650    0.9857    0.9753       140

    accuracy                         0.9739       460
   macro avg     0.9707    0.9686    0.9695       460
weighted avg     0.9739    0.9739    0.9738       460
```

14. Ensemble – Weighted Averaging

[68]:
```python
# ==================== VAL-CALIBRATED ENSEMBLE (no TTA, no training) ====================
import numpy as np, pandas as pd, matplotlib.pyplot as plt, os, cv2
from sklearn.metrics import confusion_matrix, classification_report, ConfusionMatrixDisplay
from tqdm import tqdm
import torch
from torch.utils.data import Dataset, DataLoader

# ---- assumes the following already exist from your previous script ----
# class_names, num_classes, IMG_SIZE, device
# dense (Keras DenseNet121 model), vit (PyTorch ViT model), vit_arch (string)
# functions: read_gray(path), logits_to_probs(logits_np)
# test probabilities: pdense (N_test, C) and pvit (N_test, C)   # <-- from your last run

# ---------------- paths for validation ----------------
val_csv = 'val_split.csv'
val_dir = './images_val'

IMAGENET_MEAN = np.array([0.485, 0.456, 0.406], np.float32)
IMAGENET_STD  = np.array([0.229, 0.224, 0.225], np.float32)

# ---------------- data helpers (same as before) ----------------
def read_gray(path):
    img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        raise FileNotFoundError(path)
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    return img.astype(np.float32)/255.0

def load_split(csv_path, img_dir):
    df = pd.read_csv(csv_path)
    y = (df['label'].values - 1).astype(int)
    Xg = [read_gray(os.path.join(img_dir, f"{fn}.png")) for fn in tqdm(df['filename'], desc=f"Loading {os.path.basename(img_dir)}")]
    return np.stack(Xg,0), y, df['filename'].tolist()

class GrayToViTDS(Dataset):
    def __init__(self, x_gray): self.x = x_gray
    def __len__(self): return len(self.x)
    def __getitem__(self, i):
        img = self.x[i]
        img3 = np.repeat(img[..., None], 3, axis=-1)
```

```
        img3 = (img3 - IMAGENET_MEAN) / IMAGENET_STD
        img3 = np.transpose(img3,(2,0,1)).astype(np.float32)
        import torch
        return torch.from_numpy(img3)

def vit_logits(x_gray_4d):
    dl = DataLoader(GrayToViTDS(x_gray_4d), batch_size=64, shuffle=False, num_workers=0)
    outs=[]
    vit.eval()
    with torch.no_grad():
        for b in dl:
            b = b.to(device)
            outs.append(vit(b).cpu().numpy())
    return np.vstack(outs)

def dense_probs(x_gray_4d):
    x3 = np.repeat(x_gray_4d[...,None], 3, axis=-1)
    return dense.predict(x3, batch_size=64, verbose=0)

# ---------------- 1) get validation probs & labels ----------------
Xv, yv, _ = load_split(val_csv, val_dir)
pdense_val = dense_probs(Xv)           # (Nv, C), probs
lvit_val   = vit_logits(Xv)            # (Nv, C), logits
pvit_val   = logits_to_probs(lvit_val) # (Nv, C), probs

# ---------------- 2) compute val-driven calibration vectors ----------------
eps = 1e-8
labels = np.arange(num_classes)

# per-class recall for each model
def per_class_recall(y_true, y_pred, C):
    cm = confusion_matrix(y_true, y_pred, labels=np.arange(C))
    # recall_c = TP / (TP + FN)
    TP = np.diag(cm).astype(np.float32)
    FN = cm.sum(axis=1).astype(np.float32) - TP
    return (TP + eps) / (TP + FN + eps), cm

rec_dense, cm_dense = per_class_recall(yv, pdense_val.argmax(1), num_classes)
rec_vit,   cm_vit   = per_class_recall(yv, pvit_val.argmax(1),   num_classes)

# prior correction: true class prior vs model's average predicted prior on val
true_prior = np.array([(yv==c).mean() for c in labels], dtype=np.float32) + eps
pred_prior_dense = pdense_val.mean(axis=0).astype(np.float32) + eps
pred_prior_vit   = pvit_val.mean(axis=0).astype(np.float32)   + eps
```

```
pred_prior_vit   = pvit_val.mean(axis=0).astype(np.float32)   + eps

prior_ratio_dense = true_prior / pred_prior_dense
prior_ratio_vit   = true_prior / pred_prior_vit

# Turn both into log-space addends for stability
log_w_dense = np.log(rec_dense + eps) + np.log(prior_ratio_dense)
log_w_vit   = np.log(rec_vit   + eps) + np.log(prior_ratio_vit)

print("\nVal calibration vectors:")
print("recall_dense:", np.round(rec_dense,4))
print("recall_vit  :", np.round(rec_vit,4))
print("prior_ratio_dense:", np.round(prior_ratio_dense,4))
print("prior_ratio_vit  :", np.round(prior_ratio_vit,4))

# ---------------- 3) apply calibration to TEST probabilities ----------------
# assumes you already have: pdense (N_test,C) and pvit (N_test,C)

def apply_calibration_probs(p, log_w):
    # add log-weights per class, then renormalize
    log_p = np.log(np.clip(p, 1e-8, 1.0))
    log_p_adj = log_p + log_w[None, :]          # broadcast per class
    # normalize
    m = log_p_adj.max(axis=1, keepdims=True)
    e = np.exp(log_p_adj - m)
    return e / e.sum(axis=1, keepdims=True)

pdense_cal = apply_calibration_probs(pdense, log_w_dense)
pvit_cal   = apply_calibration_probs(pvit,   log_w_vit)

# ---------------- 4) build ensembles (no learning) ----------------
# A) simple prob average after calibration
p_avg_cal = (pdense_cal + pvit_cal) / 2.0
y_avg_cal = p_avg_cal.argmax(1)

# B) dominance to stronger model (ViT) via 0.6/0.4 AFTER calibration
w_vit, w_dense = 0.6, 0.4
p_wavg_cal = w_dense * pdense_cal + w_vit * pvit_cal
y_wavg_cal = p_wavg_cal.argmax(1)

# C) geometric mean after calibration
p_geo_cal = np.sqrt(pdense_cal * pvit_cal)
p_geo_cal = p_geo_cal / p_geo_cal.sum(axis=1, keepdims=True)
y_geo_cal = p_geo_cal.argmax(1)
```

```python
y_geo_cal = p_geo_cal.argmax(1)

# ---------------- 5) reports ----------------
def show_report(name, y_true, y_pred):
    print(f"\n=== {name} ===")
    print(classification_report(y_true, y_pred, target_names=class_names, digits=4))
    cm = confusion_matrix(y_true, y_pred, labels=labels)
    ConfusionMatrixDisplay(cm, display_labels=class_names).plot(cmap=plt.cm.Blues, values_format='d')
    plt.title(f"{name} · Confusion Matrix"); plt.tight_layout(); plt.show()

# you already have 'yt' (true test labels) from your previous script
show_report("Ensemble · Val-Calibrated Prob Avg", yt, y_avg_cal)
show_report("Ensemble · Val-Calibrated Weighted Avg (0.4 Dense / 0.6 ViT)", yt, y_wavg_cal)
show_report("Ensemble · Val-Calibrated Geometric Mean", yt, y_geo_cal)
```

```
Loading images_val: 100%|██████████| 460/460 [00:00<00:00, 1325.86it/s]

Val calibration vectors:
recall_dense: [0.8774 0.9019 0.9929]
recall_vit  : [0.9717 0.9393 1.    ]
prior_ratio_dense: [0.9471 1.0404 0.9832]
prior_ratio_vit  : [0.9356 1.0446 0.987 ]

=== Ensemble • Val-Calibrated Prob Avg ===
              precision    recall  f1-score   support

      Normal     0.9340    0.9340    0.9340       106
      Benign     0.9811    0.9720    0.9765       214
   Malignant     0.9718    0.9857    0.9787       140

    accuracy                         0.9674       460
   macro avg     0.9623    0.9639    0.9631       460
weighted avg     0.9674    0.9674    0.9674       460
```