Dissertation/Report submitted to the

UNIVERSITY OF MOHAMED BOUDIAF - M'SILA, ALGERIA



FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

in partial fulfillment of the requirements for the degree of

Master's in Computer Science

Specialization: Computer Science

Option: Artificial Intelligence

Presented by

Amin Abderrahim Faid

Entitled

Deep Learning Architectures for Brain Tumor MRI Classification: A Comparative Benchmarking Study

Under the supervision of

Rabah Mokhtari

Jury Members

Prof. N. Chalabi	University of M'sila	President
Prof. R. Mokhtari Name	University of M'sila	Reporter
Prof. M. Kihal	University of M'sila	Examiner

June, 2026

Abstract

Brain tumor classification from Magnetic Resonance Imaging (MRI) is an essential challenge in clinical neuro-oncology. This thesis presents a systematic comparative study of eight deep learning architectures ResNet50, DenseNet121, VGG19, EfficientNet-B3, EfficientNetV2-S, ConvNeXt-Tiny, ConvNeXt-Small, and Swin Transformer Tiny evaluated on a dataset of MRIs collected from Nanfang Hospital, Guangzhou, China, and General Hospital, Tianjin Medical University, China, from 2005 to 2010 containing 3,064 T1-weighted contrast-enhanced images from 233 patients across three tumor classes: meningioma, glioma, and pituitary adenoma. A significant methodological addition of this work is the adoption of a strict patient-level data splitting technique, which avoids data leaking, a critical weakness present in the majority of earlier work on this dataset. All models are trained via transfer learning from ImageNet, a large-scale benchmark of 1.28 million natural images across 1,000 categories, and adapted to the MRI classification task using a two-phase protocol: frozen backbone training followed by complete fine-tuning, including inverse-frequency class weighting to solve class imbalance. Results demonstrate that the Swin Transformer Tiny obtains the best accuracy of 93.13 % and the highest meningioma F1-score of 0.9118, demonstrating the advantage of hierarchical attention mechanisms for this task. Modern convolutional architectures (ConvNeXt family) closely follow, while older CNNs (ResNet50, EfficientNet-B3) show significantly lower generalization. Meningioma categorization is found to be the hardest class across all models. These findings establish a clinically meaningful framework for future studies on this dataset.

Keywords: Brain tumor; CNN; Deep learning; Image classification; Magnetic Resonance Imaging (MRI).

الملخص

يُعد تصنيف أورام الدماغ استنادًا إلى التصوير بالرنين المغناطيسي (MRI) أحد التحديات الجوهرية في طب الأورام العصبية السريري. تقدم هذه الأطروحة دراسة مقارنة منهجية لثماني بنى تعلم عميق وهي ResNet50، DenseNet121، VGG19، EfficientNet-B3، EfficientNetV2-S، ConvNeXt-Tiny، ConvNeXt-Small، وSwin Transformer Tiny، تم تقييمها على مجموعة بيانات من صور التصوير بالرنين المغناطيسي التي تم جمعها من مستشفى نانفانغ، قوانغتشو، الصين، والمستشفى العام، جامعة تيانجينغ الطبية، الصين، من عام 2005 إلى عام 2010، والتي تحتوي على 3,064 صورة مرجحة بـ T1 معززة بالتباين من 233 مريضًا عبر ثلاث فئات من الأورام: الورم السحائي، والورم الدبقي، وورم الغدة النخامية. تمثل إحدى الإضافات المنهجية المهمة في هذا العمل في اعتماد تقنية تقسيم

البيانات على مستوى المريض بدقة، مما يمنع تسرب البيانات بين الشرائح وهي نقطة ضعف حاسمة موجودة في غالبية الأعمال السابقة على مجموعة البيانات هذه. تم تدريب جميع النماذج باستخدام التعلم النقلي من ImageNet وهي مجموعة ضخمة تضم أكثر من مليون صورة طبيعية تستخدم لتدريب النماذج مسبقا قبل تكييفها مع المهمة الهدف وقد تم التدريب باستخدام بروتوكول من مرحلتين: تدريب الهيكل الأساسي الثابت متبوعًا بضبط دقيق كامل، بما في ذلك ترجيح فئات التردد العكسي لحل عدم التوازن بين الفئات. تظهر النتائج أن Swin Transformer Tiny يحقق أفضل دقة بنسبة 93.13% وأعلى درجة F1 للورم السحائي عند 0.9118، مما يثبت تفوق العمليات القائمة على التسلسل الهرمي في هذه المهمة. وتأتي البنى التلافيفية الحديثة (عائلة ConvNeXt) في المرتبة التالية مباشرة، بينما تُظهر شبكات CNN الأقدم (ResNet50، EfficientNet-B3) قدرة تعميم أقل بكثير. ووجد أن تصنيف الأورام السحائية هو الفئة الأصعب عبر جميع النماذج. وتؤسس هذه النتائج قاعدة هامة للدراسات المستقبلية حول هذه البيانات.

الكلمات المفتاحية: ورم الدماغ; الشبكات العصبية الإلتفافية; التعلم العميق; تصنيف الصور

Dedication

"With immense pride and gratitude, I dedicate this thesis to my beloved family and dear friends. My academic journey has been filled with both challenges and achievements, yet without your unwavering support and continuous encouragement, this milestone would not have been possible.

To my mother and father, thank you for the endless support you have provided throughout the years; your profound faith in me has been the driving force behind attaining this goal. To my brother and sisters, thank you for your time, presence, and constant encouragement.

This achievement belongs to all of you, and I hope it brings you the same pride and joy that I feel today. Your love and support have always been the spark that illuminated my path"

Faid Amin Abderrahim

Acknowledgements

I would like to express my sincere thankfulness to my thesis supervisor Dr. Mokhtari for his consistent orientation, valuable feedback, and assistance throughout the progress of this project. His expertise and dedication have been invaluable. I extend my thanks to the faculty members of the Department of Computer Science at the University of Mohamed Boudiaf, M'sila, for the knowledge they imparted throughout the Master's program in Artificial Intelligence. I am grateful to the IT department at Holcim M'sila and the IT responsible Mr. Smain Tayoub for the professional environment provided during my internship, which allowed me to develop practical skills that complemented this academic work. Finally, I would like to thank Jun Cheng and his team for making the brain tumor MRI dataset publicly available, which made this research possible. His skills and effort have been priceless.

Contents

Introduction **1**

1 Chapter 1: Brain Cancer and Medical Images **3**
1.1 Introduction . . . 3
1.2 Cancer . . . 3
1.2.1 Cancer Definition . . . 3
1.2.2 Cancer Types . . . 3
1.2.3 Risk Factors . . . 4
1.2.4 Causes . . . 4
1.2.5 Symptoms . . . 5
1.2.6 Methods . . . 5
1.2.7 Limitations . . . 5
1.3 Tumors . . . 6
1.3.1 Tumors Definition . . . 6
1.3.2 Tumor Types . . . 6
1.3.3 Case Study: Brain Tumors . . . 7
1.4 Imaging for Cancer/Tumor Detection . . . 8
1.4.1 Definition . . . 8
1.4.2 Medical Imaging Types . . . 9
1.4.2.1 MRI (Magnetic Resonance Imaging) . . . 9
1.4.2.2 CT (Computed Tomography) . . . 10
1.4.2.3 PET-CT (Positron Emission Tomography) . . . 10
1.4.2.4 Ultrasound . . . 10
1.4.2.5 Mammography . . . 10
1.4.2.6 X-ray (Radiography) . . . 10
1.5 Related Work . . . 11
1.6 Conclusion . . . 12

2 Chapter 2: Dataset and Models **13**
2.1 Available Datasets and Selection Rationale . . . 13
2.1.1 Selection Rationale . . . 13
2.1.2 Limitations of the Selected Dataset . . . 14
2.2 Dataset Description . . . 15

2.2.1 The Cheng Brain Tumor MRI Dataset . . . 15
2.2.2 Dataset Content . . . 16
2.3 Models Architecture . . . 17
2.3.1 Classic CNN Family: VGG19, ResNet50, DenseNet121 . . . 17
2.3.2 Efficient CNN Family: EfficientNet-B3, EfficientNetV2-S . . . 20
2.3.3 Modern CNN Family: ConvNeXt-Tiny, ConvNeXt-Small . . . 22
2.3.4 Vision Transformer: Swin-T . . . 23
2.4 Conclusion . . . 24

3 Chapter 3: Methodology 25
3.1 Introduction . . . 25
3.2 General Workflow . . . 25
3.3 Data Preprocessing . . . 26
3.3.1 Metadata Extraction . . . 26
3.3.2 Pixel Normalisation and Channel Conversion . . . 26
3.4 Patient-Level Data Splitting . . . 27
3.4.1 Motivation . . . 27
3.4.2 Patient-Level Split Protocol . . . 27
3.5 Cross-Validation and Holdout Evaluation Strategy . . . 28
3.5.1 5-Fold Patient-Level Cross-Validation . . . 28
3.5.2 Final Holdout Evaluation . . . 28
3.6 Data Augmentation . . . 29
3.6.1 Applied Augmentations . . . 29
3.6.2 Excluded Augmentation . . . 30
3.6.3 Class Imbalance Handling . . . 31
3.7 Training Protocol . . . 31
3.7.1 Two-Phase Transfer Learning . . . 31
3.7.2 Loss Function . . . 32
3.7.3 Optimisers and Schedulers . . . 32
3.8 ImageNet Pre-training and Transfer Learning . . . 33
3.8.1 ImageNet Dataset . . . 33
3.8.2 Transfer Learning Mechanism . . . 33
3.8.3 Why ImageNet Weights Transfer to MRI . . . 34
3.8.4 Why Transfer Learning Is Necessary for This Dataset . . . 34
3.8.5 Handling the Grayscale to RGB Domain Gap . . . 34
3.9 Evaluation Metrics . . . 35
3.10 Conclusion . . . 37

4 Chapter 4: Experimental Results 38

4.1 Introduction . . . 38
4.2 Development environment . . . 38
4.2.1 Programming Language and Libraries . . . 38
4.2.2 Hardware and Platform . . . 40
4.2.3 Global Configuration and Constants . . . 40
4.3 Implementation . . . 41
4.3.1 Environment Verification . . . 41
4.3.2 Dataset Loading and PID Extraction . . . 42
4.3.3 Patient Split and Class Weights . . . 43
4.3.4 Augmentation Pipeline and Custom Dataset . . . 43
4.3.5 Model Factory . . . 45
4.3.5.1 VGG19 . . . 45
4.3.5.2 ResNet50 . . . 46
4.3.5.3 DenseNet121 . . . 46
4.3.5.4 EfficientNet-B3 . . . 46
4.3.5.5 EfficientNetV2-S . . . 47
4.3.5.6 ConvNext-Tiny . . . 47
4.3.5.7 ConvNext-Small . . . 48
4.3.5.8 Swin-T . . . 48
4.3.6 Two-Phase Training Loop . . . 48
4.4 Experimental Results . . . 51
4.4.1 Comparative Results . . . 51
4.4.2 Classic CNN Family: VGG19, ResNet50, DenseNet121 . . . 53
4.4.3 Efficient Family: EfficientNet-B3, EfficientNetV2-S . . . 55
4.4.4 Modern CNN Family: ConvNeXt-Tiny, ConvNeXt-Small . . . 56
4.4.5 Vision Transformer: Swin-T . . . 57
4.5 Ablation Study: Effect of Data Augmentation . . . 58
4.6 Discussion and Future Work . . . 60
4.6.1 Discussion . . . 60
4.6.1.1 Interpretation of Results . . . 60
4.6.1.2 Class-Level Analysis . . . 60
4.6.1.3 Comparison to Prior Works . . . 61
4.6.2 Future Work . . . 61
4.7 Conclusion . . . 62

Conclusion **63**

References **65**

A Appendix A: Mathematical Formulations **70**

Appendix A: Mathematical Formulations **70**

A.1 Image Preprocessing 70

A.1.1 Pixel Intensity Normalization 70

A.2 Class Imbalance Handling 70

A.3 Loss Function 71

A.4 Softmax Classification 71

A.5 Cosine Annealing Learning Rate 71

B Appendix B: List of Abbreviations **72**

Appendix B: List of Abbreviations **72**

B.1 Cross-Validation Fold Results 74

List of Tables

1.1 Selected related work on the Cheng brain tumor MRI dataset 11

2.1 Comparison of publicly available brain tumour MRI datasets 13
2.2 Description of fields in the Cheng brain tumor MRI dataset 15
2.3 Distribution statistics of the Cheng brain tumor MRI dataset 16

3.1 Patient-level data split summary 28
3.2 Data augmentation techniques used in this study 30
3.3 Inverse-frequency class weights computed from the training pool. 31
3.4 Training hyperparameter configuration for all evaluated models 33

4.1 Software libraries and frameworks used in this study 39
4.2 Hardware and software environment used for experimental execution . . . 40
4.3 Global configuration parameters used throughout the experiments 41
4.4 Overall ranking and performance comparison of the evaluated models on the holdout set 52
4.5 Effect of data augmentation on holdout performance across all evaluated architectures 58
4.6 Effect of data augmentation on class-wise F1-score performance across all evaluated architectures 59

B.1 Per-fold cross-validation results for all evaluated architectures. 74

List of Figures

1.1 Main differences between normal cell and cancer cell 6
1.2 Main differences between benign tumor and malignant tumor 7
1.3 The location of the three tumors inside the human brain 8
1.4 MRI scanner cutaway 9
1.5 Comparison between three different types of MRI scans 10

2.1 Representative sample images from each class with different angles 16
2.2 Classes distribution per slice 17
2.3 Classes distribution per Patient 17
2.4 VGG19 architecture: 16 convolutional layers and 3 fully connected layers with exclusively 3×3 filters 18
2.5 ResNet50 with bottleneck residual blocks enabling efficient gradient propagation 19
2.6 DenseNet121: each layer receives feature maps from all preceding layers . 19
2.7 EfficientNet-B3: compound scaling architecture. 21
2.8 EffecientNetV2-S: replaces early MBConv blocks with Fused-MBConv for faster training. 22
2.9 ConvNeXt block (right) modernised with depthwise 7×7 conv, GELU, and Layer Norm vs standard ResNet block (left) 23
2.10 Swin Transformer hierarchical architecture with shifted windows 24

3.1 End-to-end experimental pipeline: from raw .mat file loading through preprocessing, patient-level splitting, cross-validation, two-phase training, and final holdout evaluation. 26
3.2 Comparison between image-level splitting (susceptible to data leakage) and patient-level splitting (adopted in this study). 27
3.3 Patient-level 80/20 holdout split combined with 5-fold cross-validation on the training pool. 29
3.4 Examples of applied data augmentation transformations on brain MRI images: original (top row) and augmented variants (bottom row) for each transformation type. 30
3.5 Two-phase transfer learning protocol applied to all eight architectures within each fold. 32

4.1 Software environment and imported Python packages used throughout the experimental pipeline. 39
4.2 GPU environment verification and hardware detection in the Kaggle execution platform 42
4.3 Dataset loading procedure and extraction of patient identifiers (PID) and class labels from MATLAB `.mat` files 42
4.4 Patient-level train--holdout splitting and computation of inverse-frequency class weights. 43
4.5 Transformation pipeline definition for training and validation datasets. . . . 44
4.6 Custom MATDataset implementation for MRI loading and preprocessing. . 45
4.7 VGG19 model configuration and associated training hyperparameters. . . . 46
4.8 ResNet50 model configuration and associated training hyperparameters. . . 46
4.9 DenseNet121 model configuration and associated training hyperparameters. 46
4.10 EfficientNetB3 model configuration and associated training hyperparameters. 47
4.11 EfficientNetV2-S model configuration and associated training hyperparameters. 47
4.12 ConvNeXt-Tiny model configuration and associated training hyperparameters. 47
4.13 ConvNeXt-Small model configuration and associated training hyperparameters. 48
4.14 Swin-T model configuration and associated training hyperparameters. . . . 48
4.15 Implementation of Phase 1 training with a frozen backbone and trainable classification head. 49
4.16 Implementation of Phase 2 full-network fine-tuning with optimization scheduling and early stopping. 50
4.17 Example console output showing fold-wise training progression and evaluation metrics. 51
4.18 Comparison of overall holdout accuracy and macro F1score across all evaluated architectures. 52
4.19 Per-class F1-score comparison for all architectures on the holdout set. . . . 53
4.20 Confusion matrices of ResNet50 and VGG19 on the holdout set. 54
4.21 Confusion matrices of DenseNet121 on the holdout set. 54
4.22 One-vs-Rest ROC curves of ResNet50 and VGG19 on the holdout set. . . . 55
4.23 One-vs-Rest ROC curves of DenseNet121 on the holdout set. 55
4.24 Confusion matrices of the Efficient CNN family on the holdout set. 56
4.25 One-vs-Rest ROC curves of the Efficient CNN family on the holdout set. . 56
4.26 Confusion matrices of the Modern CNN family on the holdout set. 57
4.27 One-vs-Rest ROC curves of the Modern CNN family on the holdout set. . . 57

4.28 Confusion matrix and ROC Curves of the Vision Transformer family on the holdout set. 58

Introduction

Brain tumors are among the most life-threatening neurological disorders, with outcomes that vary significantly depending on tumor type, grade, and the speed of clinical reaction. Treatment planning requires early and accurate diagnosis: meningiomas are normally controlled through surgical resection, gliomas require intense multimodal therapy including radiotherapy and chemotherapy, and pituitary adenomas are often treated pharmacologically. Errors in classification at the diagnostic stage can therefore lead to fundamentally wrong treatment approaches. The gold standard for detecting brain tumors is still magnetic resonance imaging (MRI), however manually evaluating MRI data requires a lot of time, demands a high level of specialized expertise, and is subject to individual observer variability. Computer-aided diagnostic systems have attracted a lot of research interest as a result. The rapid development of deep learning, notably Convolutional Neural Networks (CNNs) and, more recently, Vision Transformers, has opened new avenues for automated medical image analysis. These architectures can extract hierarchical visual features from MRI images and learn discriminative representations that generalise across patients. However, a significant methodological fault exists in the majority of published studies on the widely known Cheng brain tumor MRI dataset: data splitting is performed at the image level rather than the patient level. Since most patients contribute several MRI slices that share identical anatomy and scanner features, image-level splitting leads slices from the same patient to appear in both training and test sets, resulting in inflated and clinically deceptive performance metrics. This thesis addresses that shortcoming directly. It presents a systematic comparative study of eight deep learning architectures VGG19, ResNet50, DenseNet121, EfficientNet-B3, EfficientNetV2-S, ConvNeXt-Tiny, ConvNeXt-Small, and Swin Transformer Tiny all trained and evaluated under a strict patient-level holdout protocol that eliminates inter-slice leakage. The goal is to develop a corrected, clinically useful standard for this dataset that later research can rely upon. The following are the main goals of this thesis:

- To conduct an accurate and unbiased evaluation of eight sample deep learning architectures for the classification of brain tumors on MRI.
- To implement and validate a patient-level data splitting technique that provides clinically significant performance estimates by removing inter-slice data leakage.
- To determine which architectural families are most appropriate for generalization to unseen patients, paying special attention to the challenging meningioma class.
- To offer a reproducible, reliable benchmark that may be relied on by future works.

The following are this work's primary contributions:

- Patient-level splitting technique: To ensure that no patient occurs in both training and test sets, a strict 80/20 holdout split is used.
- Comprehensive architectural comparison: Eight architectures, including Vision Transformers, efficient CNNs, current CNNs, and traditional CNNs, are trained under the same circumstances and assessed using the same holdout set.
- Corrected benchmark: To provide a more reliable benchmark for the Cheng dataset presented by genuine holdout accuracies on a leakage-free test set.

This is the organization of the Thesis

- Chapter 1 discusses the medical background of brain tumors, introduces MRI imaging modalities, and surveys prior deep learning efforts on this classification challenge.
- Chapter 2 discusses the Cheng dataset and the eight model architectures employed in this research.
- Chapter 3 covers the experimental methodology: preprocessing, patient-level splitting, data augmentation, training methodology, and evaluation metrics.
- Chapter 4 addresses the implementation environment and the experimental outcomes, with per-family and per-model analysis.
- General conclusion summarises the results and the contribution.

Chapter 1: Brain Cancer and Medical Images

1.1 Introduction

This chapter outlines the medical and technical framework necessary to comprehend the topic of automated brain tumor classification from MRI. It begins with a wide examination of cancer as a disease category, progressively narrowing to brain tumors and the three particular types covered in this thesis: glioma, meningioma, and pituitary adenoma. It then presents medical imaging modalities, with emphasis on MRI and its many sequences utilized in clinical neuro-oncology. The chapter finishes with a survey of earlier deep learning efforts that have employed the Cheng brain tumor dataset, highlighting the methodological gap especially the prevalence of image-level data splitting that drives this thesis.

1.2 Cancer

1.2.1 Cancer Definition

Cancer comprises a collection of disorders defined by the unregulated growth and spread of abnormal cells in the body[1]. These cells can infiltrate adjacent tissues and metastasis to remote organs via the circulation or lymphatic system. Unlike normal cells, which grow, divide, and die in an orderly method, cancer cells defy apoptosis, acquire genetic abnormalities, and damage normal tissue architecture[2]. Cancer is responsible for around 10 million deaths each year, ranking as the second highest cause of mortality worldwide[1].

1.2.2 Cancer Types

According to the World Health Organization (WHO)[1], cancers are classified into five primary histological categories, ordered by prevalence:

- **Carcinomas (85-90% of body cancers) - Epithelial cancers** Originate from epithelial cells lining organs and glands. Most prevalent cancer kind. Examples: Lung cancer, breast cancer, colon cancer.
- **Leukemias and Lymphomas (8% of body cancers) - Blood cancers** Arise from hematopoietic cells in blood, bone marrow, or lymphoid organs. Examples: Leukemia, lymphoma, multiple myeloma.

- **Central Nervous System (CNS) Tumors (2% of body cancers) - Brain/spinal cancers** Originate from brain or spinal cord cells, including glial and neuronal lineages. Examples: Glioblastoma, meningioma, medulloblastoma[3].
- **Sarcomas (1% of body cancers) - Connective tissue cancers** Develop from mesenchymal cells creating bone, cartilage, fat, or muscle. Examples: Osteosarcoma, soft tissue sarcoma.
- **Germ Cell Tumors (1% of body cancers) - Reproductive cell cancers** Derived from primordial germ cells, generally in gonads or extragonadal locations. Examples: Testicular cancer, ovarian germ cell tumors.

This categorization provides a core framework for understanding cancer epidemiology, etiology, and treatment options.

1.2.3 Risk Factors

Cancer risk originates from a mix of genetic genes, daily routines, environmental exposures, and personal qualities[1]. Some individuals possess genetic mutations handed through generations, such as alterations in the TP53 or BRCA1/2 genes, which boost the probability of developing certain cancers[2]. Lifestyle choices play a crucial role, with tobacco use responsible for about one in five cancer deaths, while heavy alcohol consumption and obesity substantially raise risk across numerous cancer types. Environmental variables such as excessive sun exposure, radiation therapy, and infections like human papillomavirus further increase to total risk. Additionally, age remains the largest predictor, with the majority of cancers identified after age sixty, and gender variations impact the sorts of cancers most typically found in men vs women[3].

1.2.4 Causes

Cancer occurs when cells amass enough damage to bypass normal growth restrictions. This process comprises activation of growth-promoting genes that urge cells to grow uncontrollably, plus inactivation of protection genes that ordinarily inhibit damaged cells from expanding. External factors such as tobacco compounds, UV radiation, and some viruses directly destroy cell DNA and interrupt normal cell death mechanisms. Repeated tissue inflammation, shortening of protecting chromosomal ends, and failures in the body's DNA repair processes all contribute to allowing cancer cells to survive and expand over time[2].

1.2.5 Symptoms

Cancer symptoms vary based on the afflicted organ and illness stage. Local indicators include prolonged coughing or coughing up blood in lung cancer, new lumps in breast tissue, and rectal bleeding or changes in bowel habits for colon cancer. Systemic warning indicators include unexplained weight loss, chronic exhaustion, and night sweats that signal the body is battling against severe illness. Brain tumors especially cause intensifying headaches that are typically more severe in the morning, seizures, and localized paralysis or trouble speaking owing to pressure accumulation and direct tissue destruction within the skull[4].

1.2.6 Methods

The diagnostic assessment of cancer relies on histological confirmation, advanced imaging, and serological biomarkers[5]. Biopsy with hematoxylin and eosin staining and immunohistochemistry remains the benchmark, augmented by molecular profiling for mutations such as IDH1 and MGMT methylation status. Cross-sectional imaging techniques like MRI, CT, and PET-CT provide tumor identification, staging, and treatment planning, while blood tumor markers including PSA, CA-125, and CEA give complementary monitoring capabilities despite weak independent specificity[6].

1.2.7 Limitations

Current diagnostic techniques face multiple important limitations. Tumors usually comprise varied tissue types; therefore, tiny biopsy samples may miss the most aggressive cancer cells and underestimate tumor malignancy[2]. Early-stage cancers typically avoid detection primarily due to their tiny size and nonspecific symptoms. Furthermore, different experts usually differ when interpreting MRI images and tissue samples, particularly when grading tumors, identifying tumor borders, or distinguishing between treatment-related alterations and real cancer recurrence[6]. The difference across doctors might lead to conflicting diagnoses and treatment options. Imaging alone typically fails to clearly differentiate benign from malignant growths, making surgical biopsy essential for a conclusive diagnosis. Finally, blood tests and post-treatment scan alterations might give false positive findings, complicating proper evaluation and perhaps leading to needless therapies or missed diagnoses.

1.3 Tumors

1.3.1 Tumors Definition

A tumor is an abnormal mass of tissue resulting from excessive, uncontrolled cellular growth. Tumors form when the usual regulatory processes of cell growth and division fail, leading to clonal growth of mutated cells [4]. Unlike hyperplasia or hypertrophy, tumors produce distinct masses that may pressure surrounding tissues. Critically, tumors are defined by their behavioral potential: benign (non-invasive) vs. malignant (invasive/cancerous), defining clinical relevance [5].

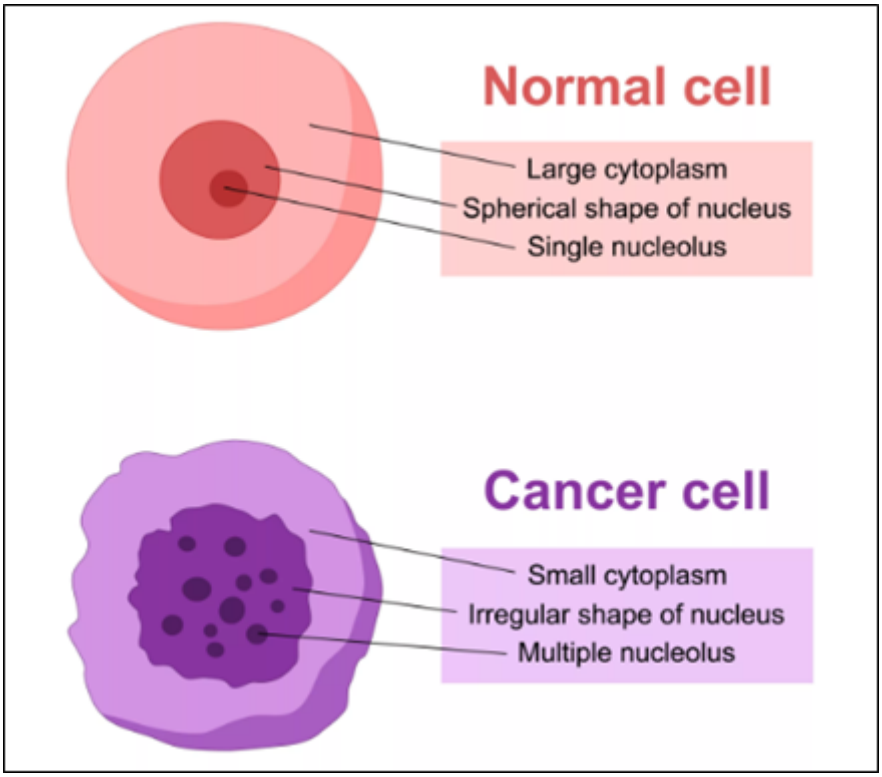


Figure 1.1: Main differences between normal cell and cancer cell

1.3.2 Tumor Types

Tumors are categorized into two primary types:

- **Benign Tumors (majority of cases):** Non-cancerous growths that develop slowly, remain confined, lack invasion/metastasis capacity, and rarely cause mortality unless squeezing essential structures[5]. Examples: Lipoma (fat tissue), Fibroma (fibrous tissue), Adenoma (glandular tissue), Meningioma (meninges, frequently benign brain tumor), Pituitary adenoma (pituitary gland).

- **Malignant Tumors (minority of cases):** Cancerous growths demonstrating fast proliferation, local tissue invasion, and distant metastasis via blood/lymphatics, presenting life-threatening danger. Examples: Glioma (brain glial cells, frequently cancerous like glioblastoma), Carcinoma (epithelial), Sarcoma (connective tissue)[4].

Key Distinction: Benign tumors (e.g., meningioma, pituitary adenoma) are often treated by surgical removal, but malignant tumors (e.g., glioma) require multimodal therapy owing to metastatic potential[7].

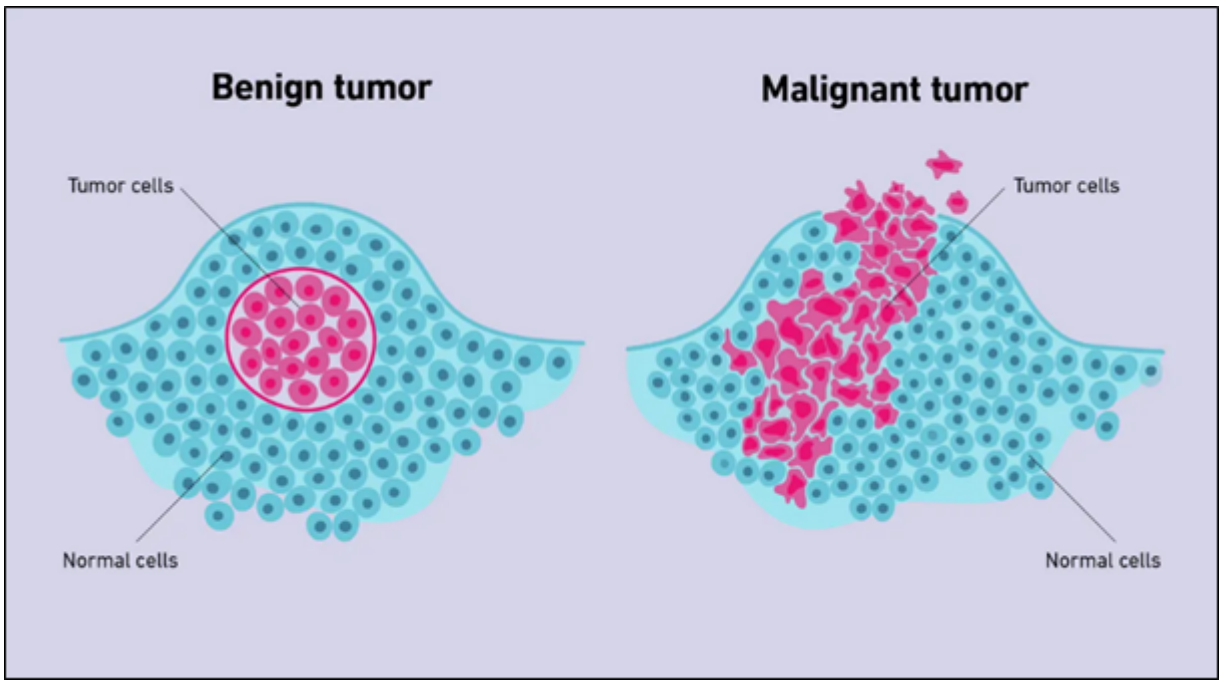


Figure 1.2: Main differences between benign tumor and malignant tumor

1.3.3 Case Study: Brain Tumors

Brain tumors constitute one of the most critical types of central nervous system (CNS) illnesses due to their direct impact on fundamental brain functioning. They consist of aberrant cellular growths forming inside the brain or surrounding intracranial structures and may be classed as benign or malignant depending on their biological activity and invasive potential. Even benign tumors may cause serious clinical problems because of the restricted space inside the skull and corresponding pressure applied to surrounding brain regions. Brain tumors are usually accompanied by symptoms such as chronic headaches, seizures, cognitive impairment, visual abnormalities, and motor dysfunction, severely reducing patient quality of life and survival outcomes[3].

Among the several brain tumor classifications, this study focuses on three therapeutically relevant tumor types: glioma, meningioma, and pituitary tumors. Gliomas arise from

glial cells and are typically regarded as the most aggressive and malignant category, particularly high-grade glioblastomas, which indicate fast development and poor prognosis. Meningiomas come from the meninges surrounding the brain and are normally benign and slow-growing, although they may still cause neurological issues through tissue compression. Pituitary tumors originate within the pituitary gland and are commonly linked with hormonal imbalance, endocrine dysfunction, and visual impairments due to their anatomical position. These three tumor classifications display discrete radiological features in magnetic resonance imaging (MRI), making them good candidates for automated classification using deep learning methods[8].

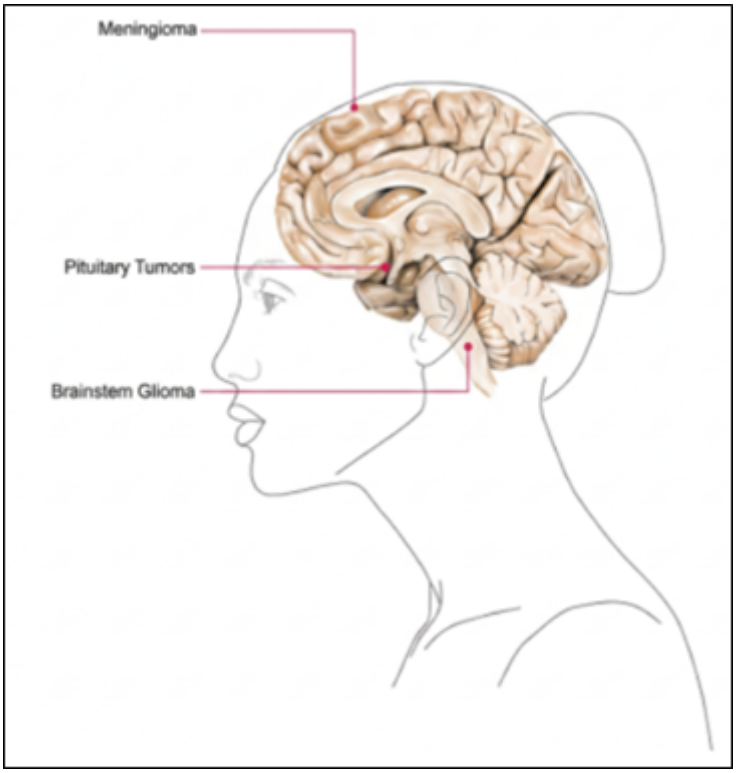


Figure 1.3: The location of the three tumors inside the human brain

1.4 Imaging for Cancer/Tumor Detection

1.4.1 Definition

Medical imaging comprises non-invasive methods employing electromagnetic radiation, magnetic fields, or ultrasound to visualize internal structures, enabling cancer/tumor identification, characterization, staging, and therapy monitoring[6].

1.4.2 Medical Imaging Types

Several imaging modalities exist; however, these six constitute the most popular employed in cancer/tumor identification within medical practice:

1.4.2.1 MRI (Magnetic Resonance Imaging)

MRI employs strong magnetic fields (1.5-3T) and radiofrequency pulses to provide high-resolution soft tissue contrast without ionizing radiation. Primary modality for brain malignancies glioma and meningioma[8].

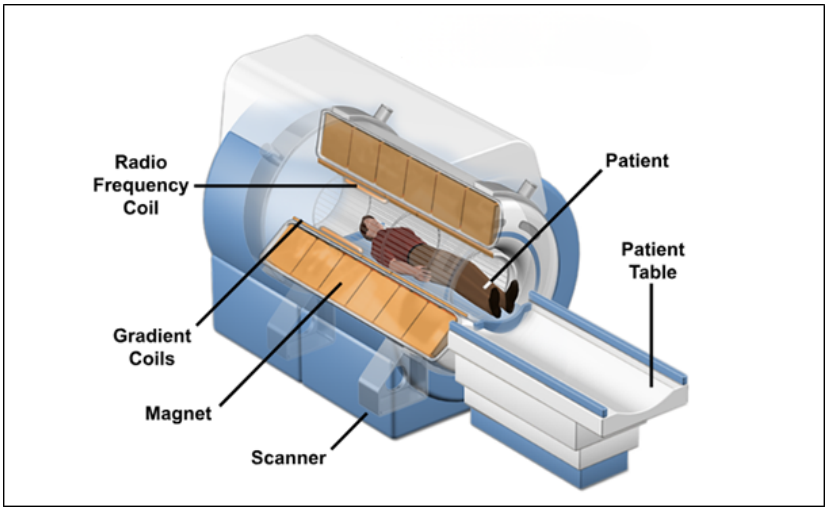


Figure 1.4: MRI scanner cutaway

Different sequences of MRIs are used in the medical fields, this is some of the widely used:

- **T1-weighted** Fat bright, water dark; great anatomical detail; differentiates gray/white matter.
- **T2-weighted** Water bright (edema, cysts), fat moderate; emphasizes tumor edema.
- **FLAIR (Fluid-Attenuated Inversion Recovery)** Suppresses cerebrospinal fluid signal, exposes peritumoral infiltration.
- **T1 post-Gadolinium** Contrast agent extravasation through a damaged blood-brain barrier signals active tumor.
- **DWI/ADC (Diffusion-Weighted Imaging)** Measures water diffusion restriction; shows high cellularity in high-grade malignancies.

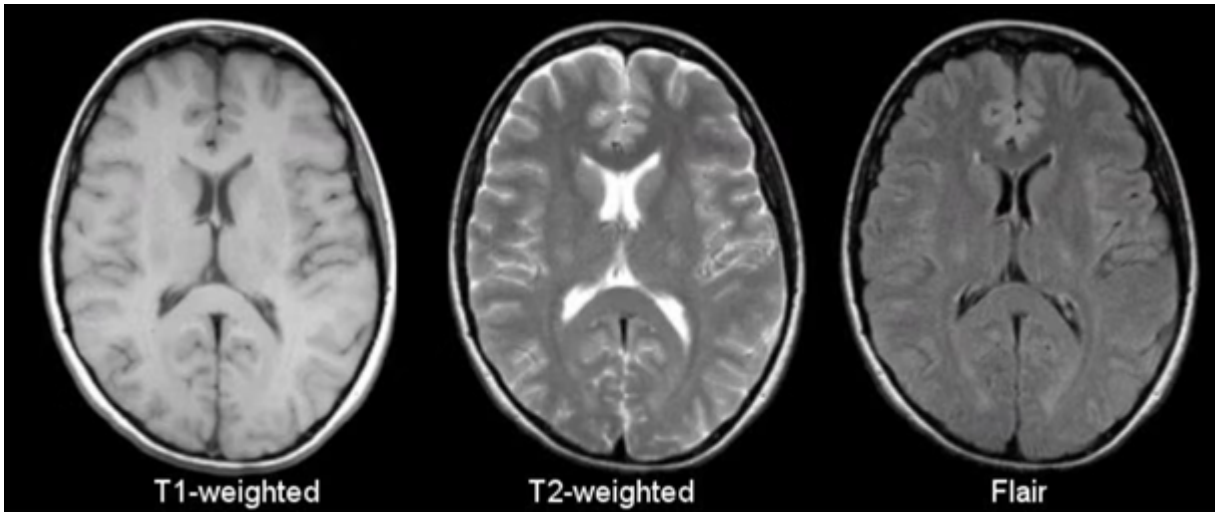


Figure 1.5: Comparison between three different types of MRI scans

1.4.2.2 CT (Computed Tomography)

Ionizing X-rays provide cross-sectional pictures via a rotating detector. Detects lung nodules (>6 mm), bone destruction, and abdominal tumors. Fast acquisition (seconds)[6].

1.4.2.3 PET-CT (Positron Emission Tomography)

18F-FDG radiotracer accumulates in metabolically active cancer cells. Standardized uptake value (SUV) measures malignancy. Essential for staging lymphoma, identifying metastases[6].

1.4.2.4 Ultrasound

High-frequency sound waves (2-18 MHz) reflect off tissue surfaces. Real-time imaging guides biopsies. Differentiates solid (malignant) vs. cystic (benign) breast/thyroid lesions[6].

1.4.2.5 Mammography

Low-dose X-rays designed for breast tissue. Detects microcalcifications (<1 mm) and architectural deformation. Digital tomosynthesis (3D) decreases false positives[6].

1.4.2.6 X-ray (Radiography)

Single projection X-ray attenuation is dependent on tissue density. Screens for lung masses and bone cancers. Low cost, commonly accessible[6].

1.5 Related Work

Various studies have used the Cheng brain tumor MRI dataset, demonstrating a clear development from traditional machine learning to recent deep learning techniques. Cheng et al[8], [9]. constructed an early baseline combining histogram characteristics with an SVM, obtaining 91.28% accuracy. This was followed by numerous CNN-based approaches, including the standard architecture by Abiwinanda et al[10]. (84.19%), the compact network of Pashaei et al[11]. (90.26%), the transfer learning strategy of Deepak & Ameer[15] (95.10%), and the multi-class framework of Sultan et al[12]. (96.13%). Complementing these pure deep learning approaches, Khan et al[13]. combined CNN-derived representations with hand-crafted features to reach 90.67%. Building upon this advancement, the present study employs a Swin Transformer (Swin-T)[14], which achieves 93.13% accuracy. This result ranks the suggested methodology favourably among existing methods, highlighting the possibility of transformer-based designs to promote automated brain tumor classification. The table below summarises some of the previous works on this dataset.

Table 1.1: Selected related work on the Cheng brain tumor MRI dataset

Study	**Method**	**Split Type**	**Accuracy**
Cheng et al.[8]	Histogram + SVM	Image-level	91.28%
Abiwinanda et al.[10]	CNN	Image-level	84.19%
Pashaei et al.[11]	Compact CNN	Image-level	90.26%
Deepak & Ameer[15]	CNN + Transfer	Image-level	95.10%
Sultan et al.[12]	Multi-class CNN	Image-level	96.13%
Khan et al.[13]	CNN + Features	Patient-level	90.67%
This work	Swin Transformer	Patient-level	93.13%

A critical methodological issue affecting most of these studies is that the splitting of data is performed at the image level rather than at the patient level.

As the majority of patients provide multiple MRI slices showing the same anatomy, scanner parameters, and tumour shape, image-level splitting allows slices from the same patient to appear in both training and test sets simultaneously, so the model memorises patient-specific appearances rather than learning generalisable tumour features, creating inflated and clinically misleading performance metrics[16]. As demonstrated in the table, five of the six preceding studies employ image-level splitting, which explains why their reported accuracies exceed 95 % yet cannot be reproduced under clinically realistic evaluation conditions. The prior work employing patient-level splitting is Khan et al. [13] at 90.67 %. The present work employs a strict patient-level 80/20 holdout combined with 5-fold cross-validation [17], ensuring that no patient's slices appear in both training and evaluation sets at any point.

1.6 Conclusion

This chapter has provided the medical and technical basis of this thesis. Brain cancers particularly glioma, meningioma, and pituitary adenoma present specific clinical challenges and require precise, rapid diagnosis. MRI, especially T1-weighted contrast-enhanced imaging, is the dominant non-invasive diagnostic method for these illnesses. The assessment of earlier work on the Cheng dataset reveals a reoccurring methodological issue: the widespread use of image-level data splitting, which inflates reported performance metrics by enabling patient-specific information to leak from training into test sets. While numerous studies have obtained accuracy figures exceeding 95%, these results are not directly comparable to a real held-out patient cohort study. The methodology outlined in the next chapters addresses this shortcoming by applying a rigorous patient-level split throughout all tests.

Chapter 2: Dataset and Models

This chapter introduces the two core components of the experimental framework: the dataset and the model architectures. The first section describes the Cheng brain tumour MRI dataset in detail. The second section presents the eight deep learning architectures selected for this comparative study, grouped into four families.

2.1 Available Datasets and Selection Rationale

Numerous publicly accessible MRI datasets have been used in the brain tumour classification task.

Choosing the most suitable one requires the assessment of multiple criteria: tumour class coverage; availability of patient-level identifiers (PIDs); imaging modalities; annotation quality; main purpose of the dataset, whether it is classification, segmentation, or something else; and community adoption as a benchmark. Table 2.1 summarises the major candidates considered for this study.

Table 2.1: Comparison of publicly available brain tumour MRI datasets

Dataset	Classes	Patients	Images	Modality	PID Available	Task	Format
Cheng et al. [8]	3 (Men, Gli, Pit)	233	3,064	T1-CE	**Yes**	Classification	.mat
BraTS 2021 [18], [19]	2 (GBM, LGG)	∼2,000	8,160	T1/T2/FLAIR/ T1-CE	Yes	Segmentation	NIfTI
TCGA-GBM / LGG [20]	2 (GBM, LGG)	∼778	∼15,000	Multi-sequence	Yes	Grade / Survival	DICOM
Kaggle Brain MRI [21]	4 (Men, Gli, Pit, None)	N/A	7,023	T1 (mixed)	**No**	Classification	JPEG
IXI Dataset	Healthy only	600	∼18,000	T1/T2/PD	Yes	No tumours	NIfTI
RIDER Neuro MRI [20]	GBM only	19	∼70,000	T1-CE	Yes	Longitudinal	DICOM

2.1.1 Selection Rationale

The Cheng et al. dataset [8], [9] was chosen over other options for reasons directly related to the methodological objectives of this study:

1. **Explicit patient identifiers (PIDs).** The Cheng dataset is the only classification-oriented dataset in Table 2.1 that provides a patient identifier field for every image (`Cjdata.PID`) [8]. This is the most critical selection criterion because without PIDs, it is impossible to implement patient-level splitting, and the entire methodological contribution of this thesis cannot be realised. The Kaggle Brain MRI dataset, despite its

larger size and four-class structure, provides no patient-level metadata, making it unsuitable for leakage-free evaluation.

2. **Established benchmark status.** The Cheng dataset is the most widely used benchmark for brain tumour classification task in the deep learning literature with over a decade of published results. This enables direct comparison with previous works and provides a meaningful context for the corrected patient-level results produced in this study.

3. **Classification task alignment.** BraTS [18], [19] and TCGA [20] are primarily designed for tumour segmentation and grade/survival prediction, respectively. Adapting them to a three-class classification task would require non-trivial label merging and preprocessing not present in any prior comparable work. The Cheng [8] dataset is natively formulated as a three-class classification problem which makes it suitable for our task.

4. **Clinically relevant tumour types.** The three classes meningioma, glioma, and pituitary adenoma represent the most commonly encountered tumour types in clinical neuro-oncology practice [3], [4]. This makes classification results clinically meaningful beyond the dataset itself.

5. **Lossless image format.** Images are stored in MATLAB `.mat` files which preserve the original scanner intensity values and quality without compression artefacts. On the contrary, Kaggle dataset uses JPEG encoding, which introduces lossy compression that changes fine-grained patterns critical for MRI-based classification.

6. **Expert manual annotations.** Each image includes tumour borders and binary masks drawn by experts and brain doctors (`Cjdata.tumorBorder`, `Cjdata.tumorMask`) [8], enabling potential future use of tumour region-focused training strategies.

7. **Uniform acquisition protocol.** All 3,064 slices were acquired with a consistent T1-weighted contrast-enhanced protocol using Gadolinium (0.1 mmol/kg at 2 ml/s) across two hospitals in China between 2005 and 2010 [8], ensuring that inter-scanner and inter-protocol variability is minimised compared to heterogeneous multi-site collections.

2.1.2 Limitations of the Selected Dataset

Despite the advantages mentioned above, the Cheng dataset has obvious drawbacks.

It covers only T1-CE sequences, while clinical diagnosis often employs multi-sequence protocols, including T2 and FLAIR.

The dataset is derived from one specific geographic and ethnic population gathered between 2005 and 2010, which may be restricting its generalizability to modern scanners or varied patient demographics.

Additionally, the three-class framework does not address the tumor grading type, particularly the clinically essential distinction between low-grade and high-grade glioma [7].

These restrictions are described further in the Future Work section.

2.2 Dataset Description

2.2.1 The Cheng Brain Tumor MRI Dataset

This study employed the brain tumor MRI dataset originally assembled by Cheng et al[8], [9]. (2015, 2016), which includes T1-weighted contrast-enhanced MRI scans from 233 patients at Nanfang Hospital, Guangzhou, and General Hospital, Tianjin Medical University, collected between September 2005 and October 2010. The dataset is widely used in deep learning research for automated brain tumor classification due to its uniform acquisition process, expert manual annotations, and clinically verified class labels. All images are stored in MATLAB .mat files. Each file contains a data structure with five pieces of information[8]. The images

Table 2.2: Description of fields in the Cheng brain tumor MRI dataset

Field	**Description**
Cjdata.image	The MRI scan itself (512×512 grayscale image).
Cjdata.label	Tumor type: 1 = meningioma, 2 = glioma, 3 = pituitary.
Cjdata.PID	Patient identification (ID) number.
Cjdata.tumorBorder	List of x--y coordinates marking the expert-drawn tumor border.
Cjdata.tumorMask	Black-and-white mask indicating pixels belonging to the tumor region.

are T1-weighted contrast-enhanced MRI scans. Patients were given a Gadolinium contrast agent (0.1 mmol/kg at 2 ml/s) before to scanning to enhance tumor visibility. Each picture has 512 × 512 pixels, with each pixel corresponding to 0.49 × 0.49 mm of cerebral tissue. The MRI slice thickness measures 6 mm, with a 1 mm interval between successive scans. Utilizing the .mat format preserves the original image brightness values without any degradation in quality due to compression. This is essential because automated classification depends on detection of minor brightness changes where the contrast agent has accumulated in tumor tissue. The dataset also contains a file named cvind.mat that gives conventional 5-fold cross-validation groups for benchmarking against other researchs.

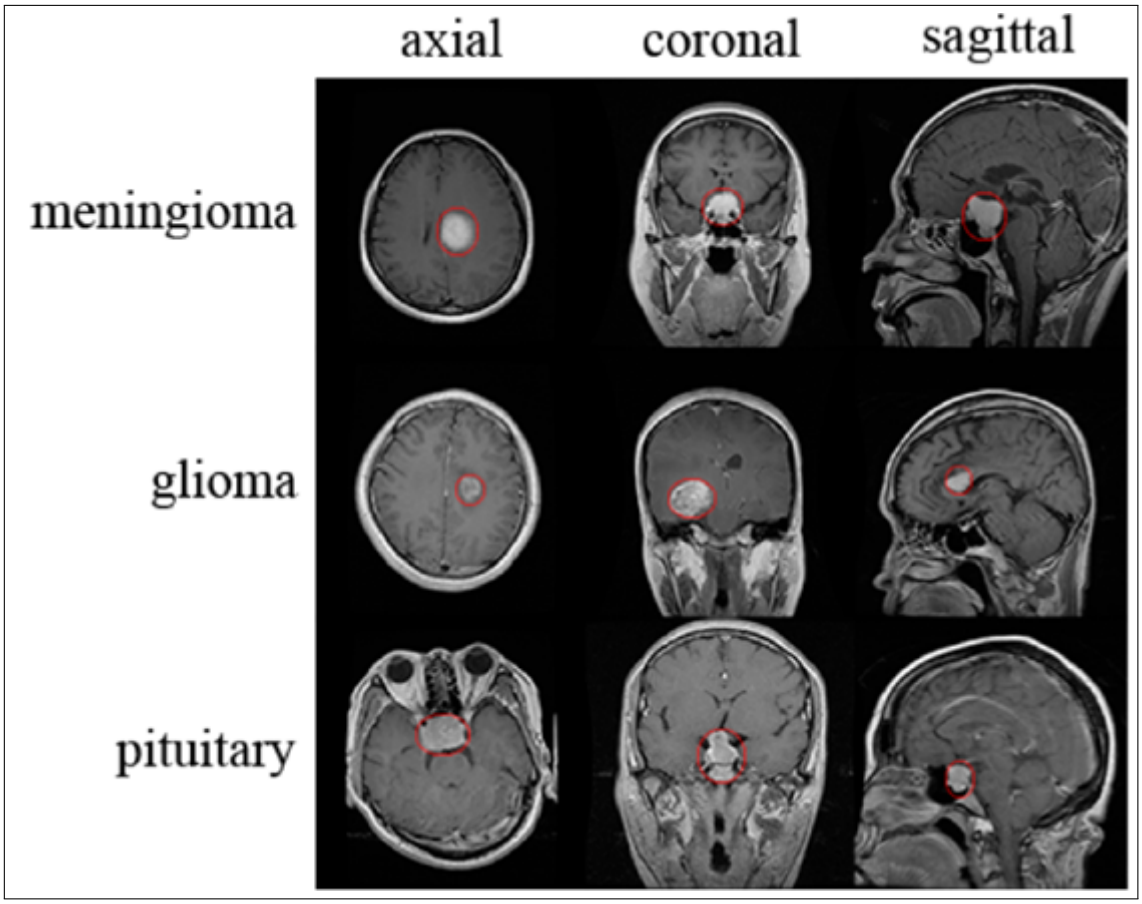


Figure 2.1: Representative sample images from each class with different angles

2.2.2 Dataset Content

The dataset contains 3,064 slices from 233 patients, categorized into three distinct pathological classes representing the most common brain tumor types encountered in clinical practice[3], [8].

Table 2.3: Distribution statistics of the Cheng brain tumor MRI dataset

Class	**Slices**	**Patients**	**Avg Slices/Patient**	**Min**	**Max**
Meningioma	708	82	5.9	1	22
Glioma	1426	89	20.4	2	57
Pituitary	930	62	21.1	3	74
Total	3064	233	13.2	1	74

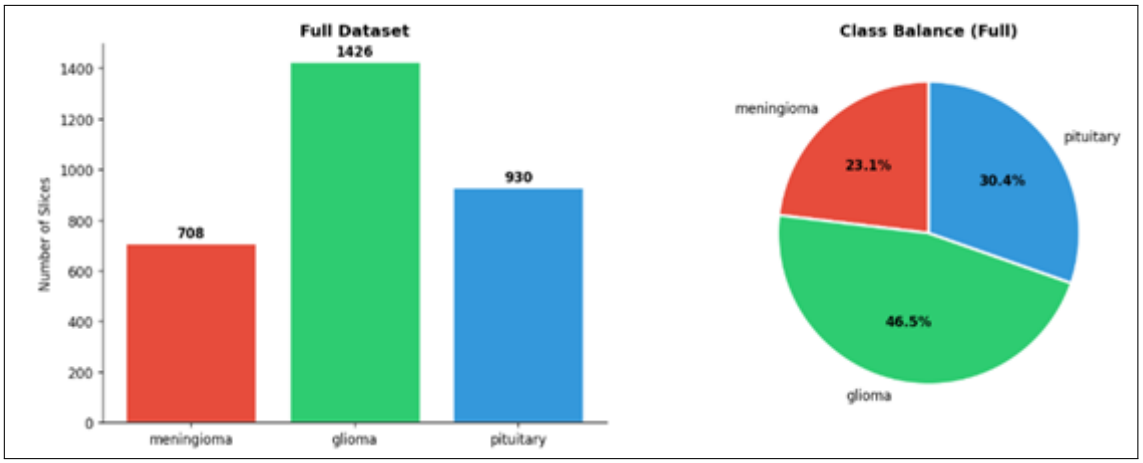


Figure 2.2: Classes distribution per slice

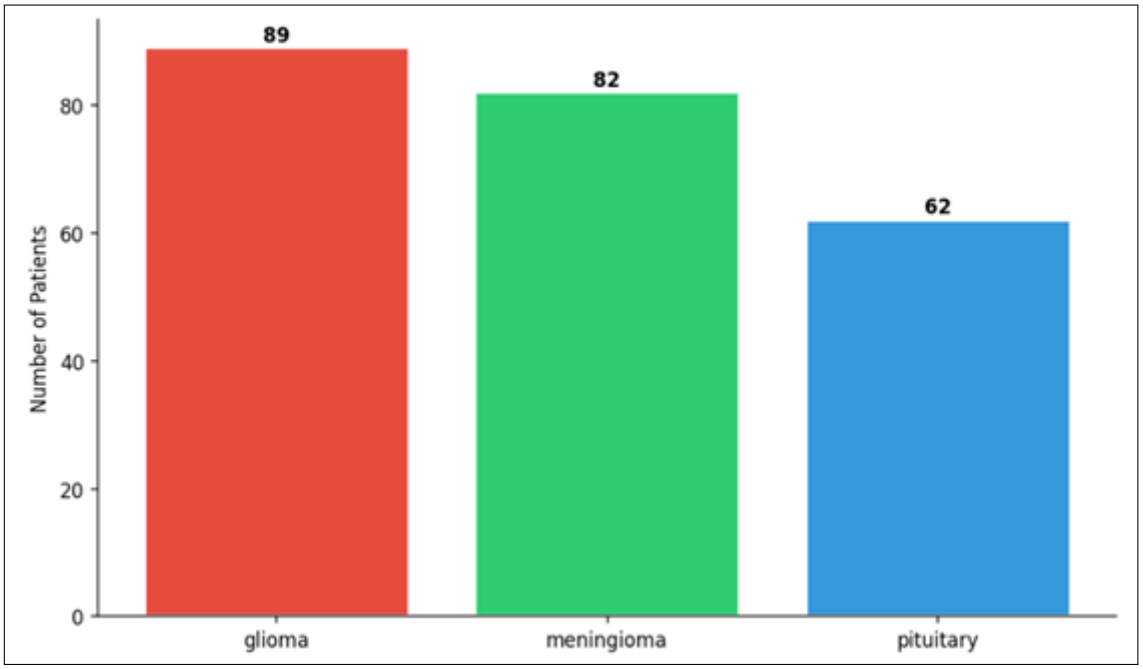


Figure 2.3: Classes distribution per Patient

The class distribution reveals moderate imbalance, with glioma cases representing nearly half the dataset, necessitating careful handling during model training to prevent bias toward the majority class[22].

2.3 Models Architecture

2.3.1 Classic CNN Family: VGG19, ResNet50, DenseNet121

The classic CNN family represents some of the most popular architectures in deep learning and computer vision[22], [23]. These models established the foundation for modern image

analysis systems by continually increasing feature extraction capabilities, network depth, and optimization efficiency. Their development brought major architectural advances that significantly advanced image classification performance[24].

VGG19 was launched by Karen Simonyan and Andrew Zisserman from the Visual Geometry Group (VGG) at the University of Oxford in 2014[25]. It revealed that increasing network depth with simple convolution filters could greatly enhance performance.

VGG19 has 19 weight layers that utilize just 3×3 convolutions, producing a large receptive field via increased depth. Although its age, it continues to function as a robust foundation for transfer learning[26]. This study substitutes the classifier with a more complex architecture: 25088→4096→4096→3, incorporating a 0.5 dropout rate at each fully connected layer.

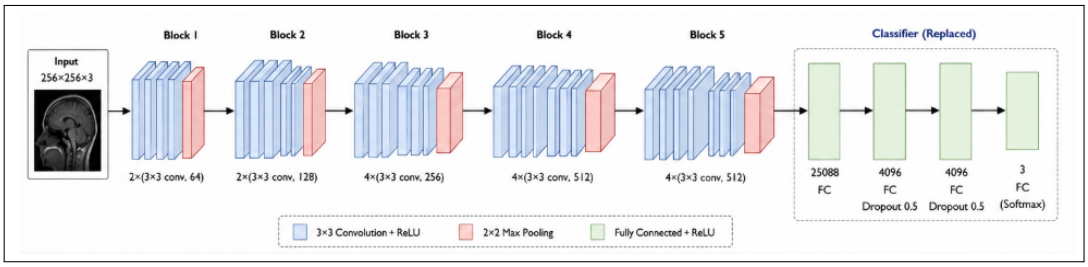


Figure 2.4: VGG19 architecture: 16 convolutional layers and 3 fully connected layers with exclusively 3×3 filters

ResNet was introduced by Kaiming He et al. at Microsoft Research in 2015[27]. It addressed the degradation problem in deep networks through residual learning and skip connections, enabling significantly deeper structures to be trained effectively.

ResNet50 introduced the residual (skip) connection, which directly adds the block input to its output, solving the vanishing gradient problem and facilitating the construction of very deep networks. ResNet50 employs a bottleneck block including 1×1, 3×3, and 1×1 convolutions. In this work the classifier head is deepened: Linear(2048, 512)→ReLU→Dropout(0.3) →Linear(512, 3).

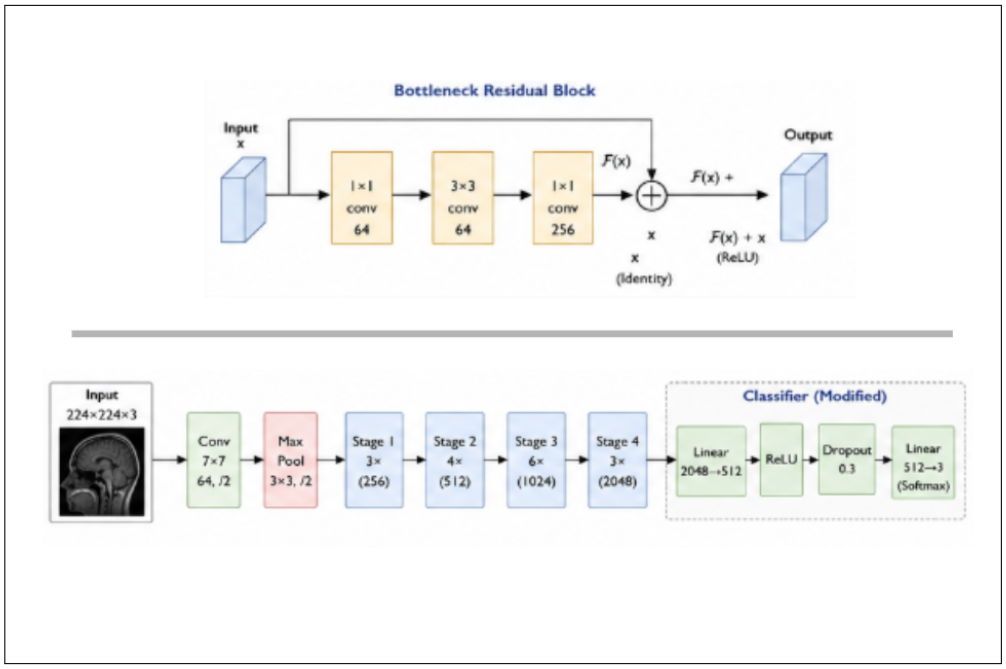


Figure 2.5: ResNet50 with bottleneck residual blocks enabling efficient gradient propagation

DenseNet was proposed by Gao Huang et al. in 2017 and introduced dense connectivity between layers, where each layer receives feature maps from all preceding layers[28]. This design improves feature reuse and reduces the number of trainable parameters.

DenseNet121 connects every layer to every following layer inside a dense block, optimizing feature reuse and reduces the number of trainable parameters, making it particularly effective for small-scale medical imaging datasets[6], [28]. A stepwise unfreezing technique is applied: classifier head → denseblock4 → entire network[26].

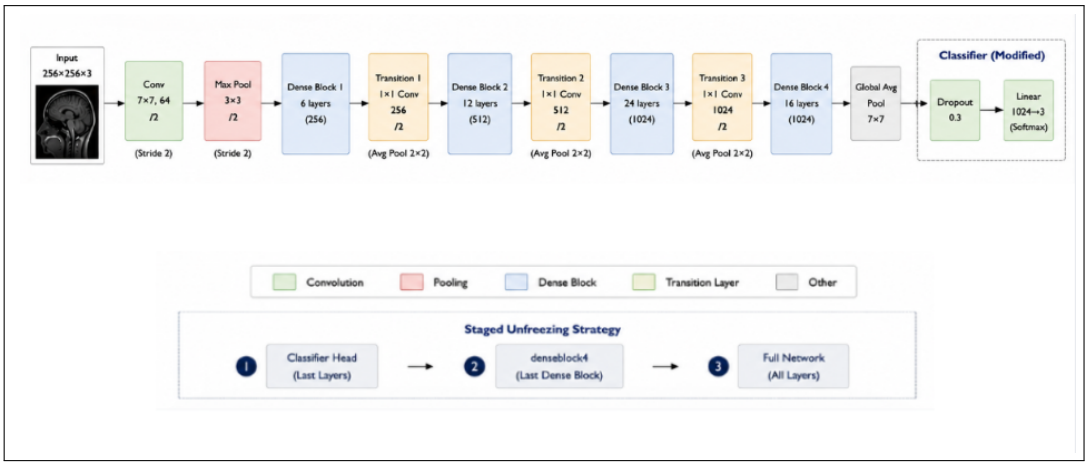
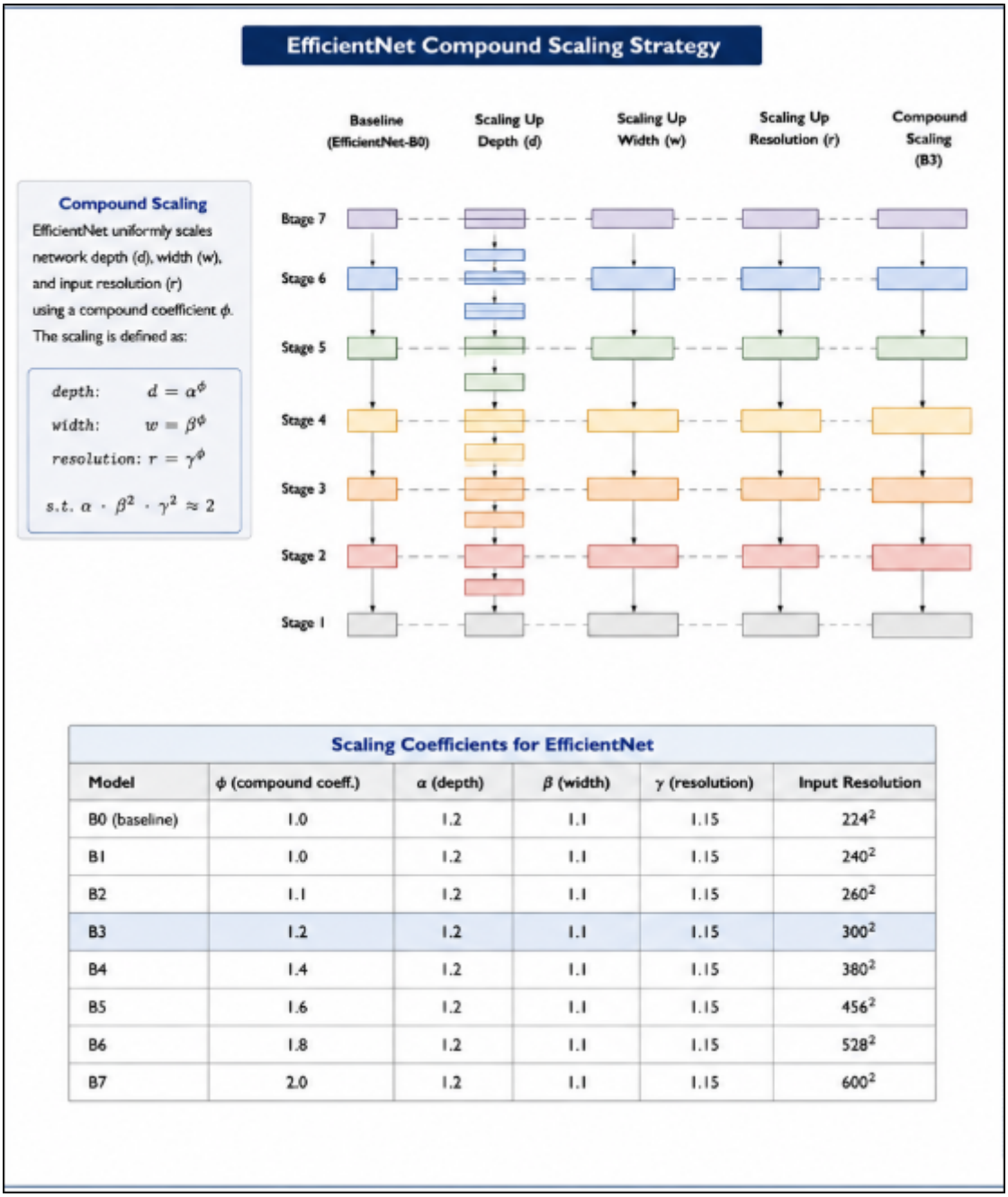


Figure 2.6: DenseNet121: each layer receives feature maps from all preceding layers

2.3.2 Efficient CNN Family: EfficientNet-B3, EfficientNetV2-S

EfficientNet-B3 was introduced by Mingxing Tan and Quoc V. Le from Google Research in 2019[29]. It was developed to address the limitations of conventional scaling methods, where network depth, width, and image resolution were typically increased independently. EfficientNet demonstrated that a more balanced scaling strategy could achieve improved classification performance while requiring fewer computational resources and parameters than many previous CNN architectures.

EfficientNet-B3 introduced compound scaling, concurrently scaling network depth, width, and input resolution. Squeeze-and-Excitation (SE) blocks are included throughout, recalibrating channel-wise feature responses. Note that SE blocks are incompatible with PyTorch DataParallel on this CUDA version, forcing single-GPU training.



Scaling Coefficients for EfficientNet

Model	ϕ (compound coeff.)	α (depth)	β (width)	γ (resolution)	Input Resolution
B0 (baseline)	1.0	1.2	1.1	1.15	224^2
B1	1.0	1.2	1.1	1.15	240^2
B2	1.1	1.2	1.1	1.15	260^2
B3	1.2	1.2	1.1	1.15	300^2
B4	1.4	1.2	1.1	1.15	380^2
B5	1.6	1.2	1.1	1.15	456^2
B6	1.8	1.2	1.1	1.15	528^2
B7	2.0	1.2	1.1	1.15	600^2

Figure 2.7: EfficientNet-B3: compound scaling architecture.

EfficientNetV2 was proposed by Mingxing Tan and Quoc V. Le in 2021 as the next generation of EfficientNet architectures. The model was designed with emphasis on improving training speed and practical deployment efficiency while maintaining strong classification performance. It introduced several modifications intended to reduce computational overhead and accelerate convergence during training[30].

EfficientNetV2-S refines EfficientNet by replacing early depthwise convolutions with Fused-MBConv blocks and implementing progressive learning. V2-S works at native 384×384 resolution, requiring batch size 32 to fit GPU memory.

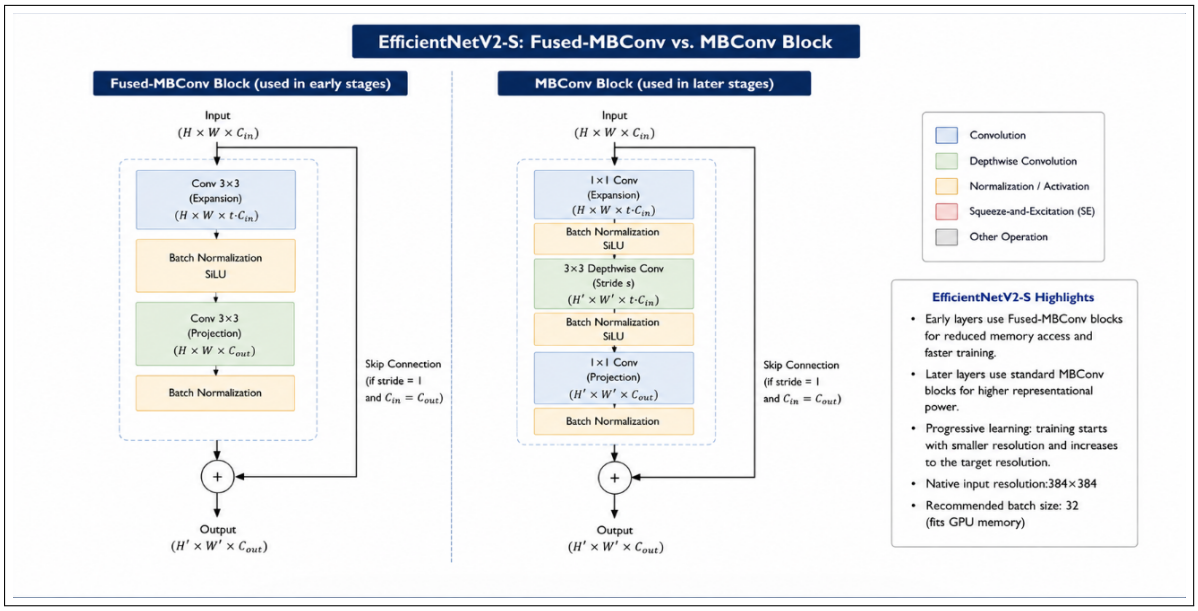


Figure 2.8: EffecientNetV2-S: replaces early MBConv blocks with Fused-MBConv for faster training.

2.3.3 Modern CNN Family: ConvNeXt-Tiny, ConvNeXt-Small

ConvNeXt was introduced by Zhuang Liu et al. from Meta AI Research in 2022 as a modernized convolutional architecture designed to close the performance gap between traditional CNNs and Vision Transformers[31]. Rather than introducing transformer mechanisms directly, ConvNeXt revisited the design of conventional CNNs and incorporated several architectural concepts inspired by Vision Transformers while preserving a fully convolutional structure[32]. ConvNeXt-Tiny and ConvNeXt-Small represent scaled variants of this architecture and are included in this study to evaluate whether modernized CNN designs can achieve improved generalization and classification performance in brain tumor MRI analysis.

ConvNeXt is a pure CNN enhanced by taking components from Vision Transformers: depthwise convolutions with large 7×7 kernels, inverted bottleneck blocks, Gaussian Error Linear Unit (GELU) activations, layer normalization, and a decreased amount of activation and normalization layers per block. ConvNeXt-Tiny (28.6M) and ConvNeXt-Small (50.2M) differ solely in the number of blocks in each stage. Both employ label smoothing (0.1) and dropout (0.5) in this investigation.

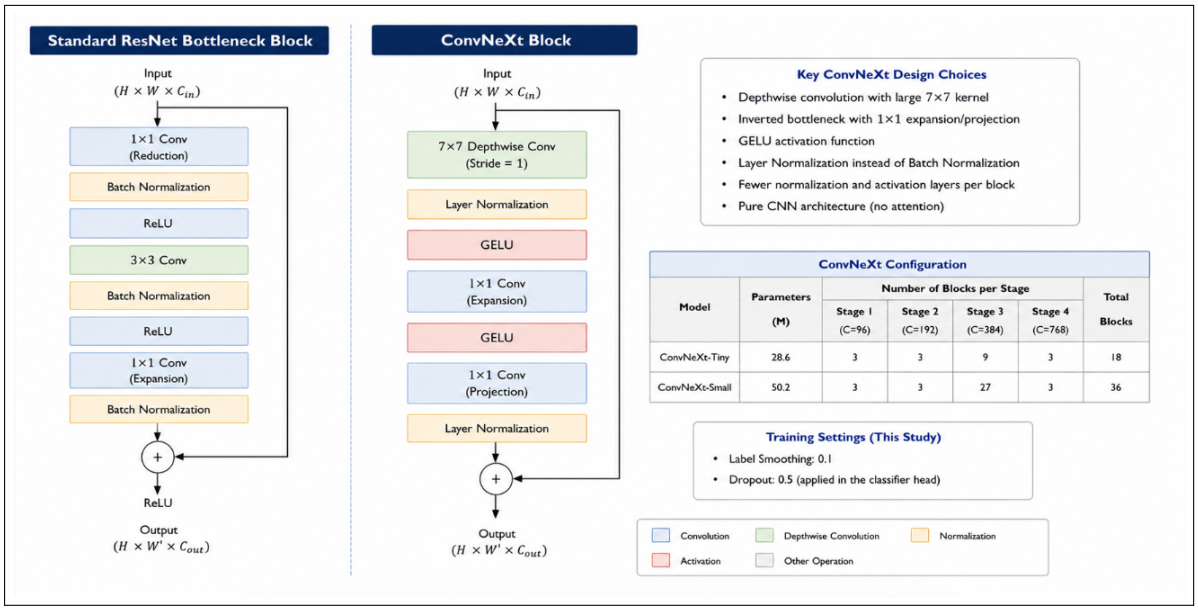


Figure 2.9: ConvNeXt block (right) modernised with depthwise 7×7 conv, GELU, and Layer Norm vs standard ResNet block (left)

2.3.4 Vision Transformer: Swin-T

Swin Transformer was introduced by Ze Liu et al. from Microsoft Research Asia in 2021 as a hierarchical Vision Transformer architecture designed to improve computational efficiency and scalability in visual tasks[14]. Unlike conventional Vision Transformers that apply global self-attention across the entire image, Swin Transformer introduced a local window-based attention mechanism that significantly reduces computational complexity while preserving strong representation capability. The architecture was developed to combine the advantages of transformer-based learning with practical efficiency for high-resolution image analysis and has demonstrated strong performance across various computer vision tasks.

Swin Transformer Tiny uses hierarchical shifted-window self-attention. The image is broken into 4×4 patches processed through four phases with increasingly integrated patches, generating feature maps at H/4×W/4 through H/32×W/32. Self-attention is computed inside local windows, with windows switched between succeeding layers. Swin-T runs at 224×224 resolution.

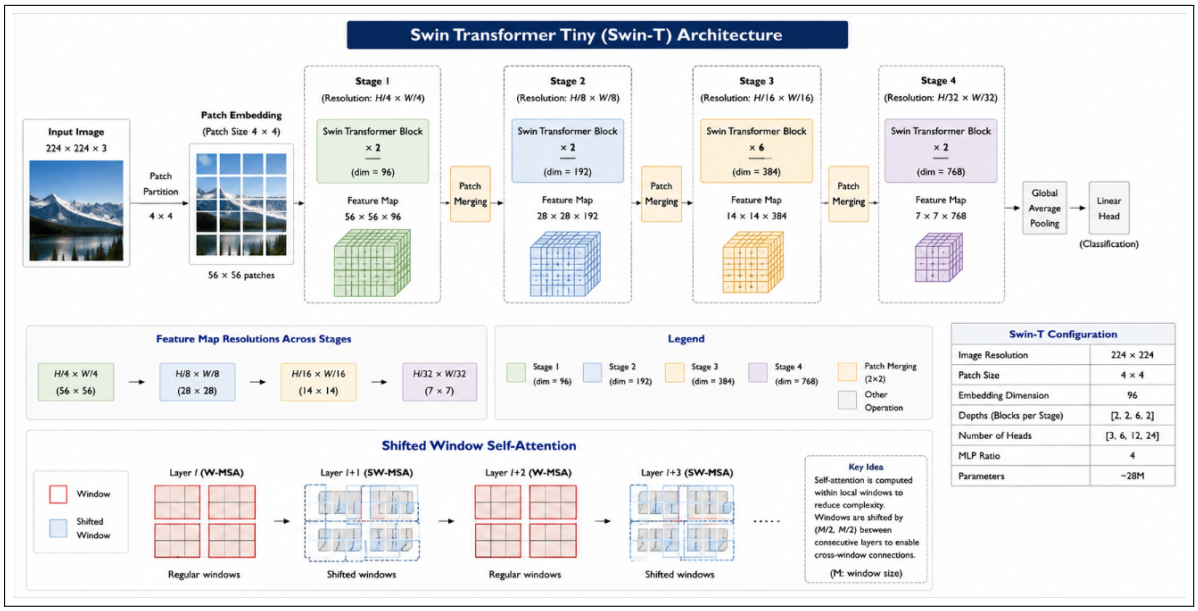


Figure 2.10: Swin Transformer hierarchical architecture with shifted windows

2.4 Conclusion

This chapter has described the dataset and the eight model architectures used in this study. The Cheng MRI dataset, with its 3,064 slices from 233 patients and three tumor classes, provides a well-established benchmark whose moderate class imbalance and high per-patient slice variance directly motivated the preprocessing and training choices detailed in Chapter 3. The eight selected architectures span a deliberate cross-section of the current deep learning landscape from the well-established VGG19 and ResNet50 to modern Transformer-inspired designs making this comparison informative both in accuracy terms and in architectural philosophy.

Chapter 3: Methodology

3.1 Introduction

This chapter describes the complete experimental methodology adopted in this study. All design decisions are motivated by two overarching goals: clinical validity and reproducibility. The chapter covers the end-to-end workflow, data preprocessing, patient-level data splitting, cross-validation strategy, data augmentation (including an ablation study), class imbalance handling, the two-phase transfer learning protocol, and the evaluation metrics used throughout all experiments.

3.2 General Workflow

The process begins with loading raw brain MRI .mat files and preprocessing them into a standardised tensor format. To ensure clinical validity, the dataset is partitioned at the patient level[16], with 80 % of unique patients reserved for development and 20 % held out as an entirely unseen test cohort. The development pool undergoes stratified 5-fold cross-validation[17], [33], where data augmentation is applied strictly to the training folds[34]. Eight distinct architectures—spanning CNNs, vision transformers, and lightweight hybrids—are trained from ImageNet pretrained weights[35] using a common two-stage transfer learning strategy[26]. Each model is first evaluated via cross-validation, followed by a final unbiased assessment on the independent holdout set. The workflow concludes with a comparative analysis to select the optimal model for 3-class brain tumour classification. Figure 3.1 illustrates the end-to-end pipeline adopted in this study.

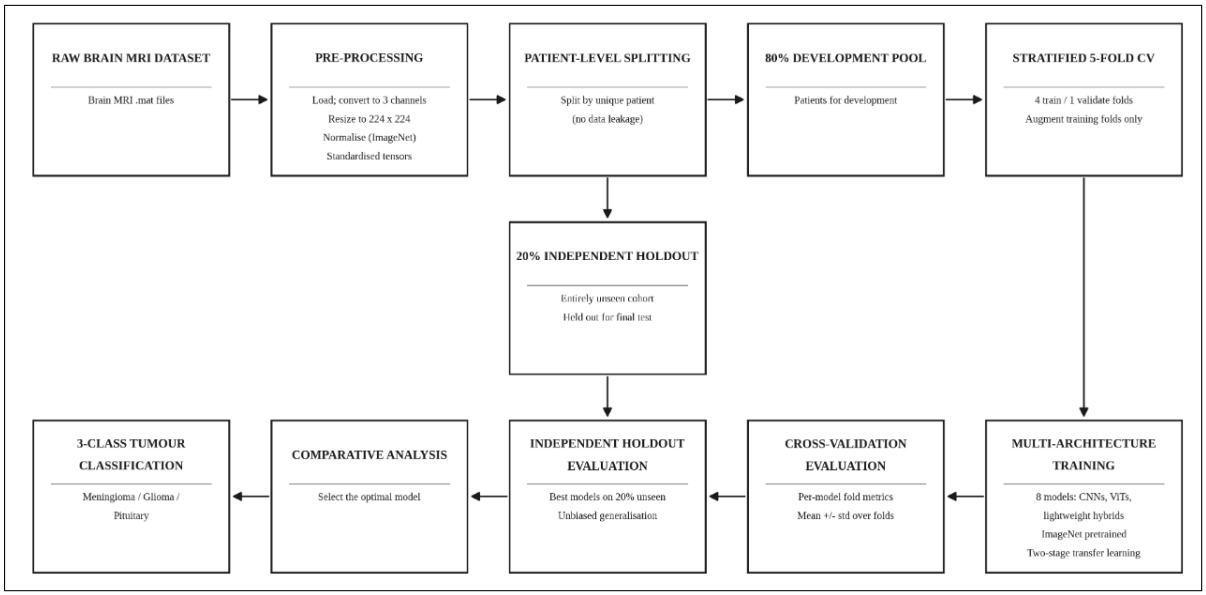


Figure 3.1: End-to-end experimental pipeline: from raw .mat file loading through preprocessing, patient-level splitting, cross-validation, two-phase training, and final holdout evaluation.

3.3 Data Preprocessing

3.3.1 Metadata Extraction

The dataset archives each MRI slice as a MATLAB .mat file that contains a structure cjdata with the following fields: image (pixel matrix), label (integer class: 1=meningioma, 2=glioma, 3=pituitary), PID (patient identifier represented as a Unicode uint16 character array), tumorMask (binary mask), and tumorBorder (contour coordinates)[8]. The patient ID is derived by transforming the uint16 array into its string representation (e.g., MR0507918). This PID functions as the categorical key for all patient-level segmentation procedures. Class labels are remapped from 1, 2, 3 to 0, 1, 2 for compliance with PyTorch's loss functions[36].

3.3.2 Pixel Normalisation and Channel Conversion

Raw pixel intensities are normalized to the range $[0, 255]$ using per-image min--max normalization[22]:

$$I_{\text{norm}} = \left\lfloor \frac{255}{I_{\max} - I_{\min}} \times (I - I_{\min}) \right\rfloor \tag{3.1}$$

where I denotes the original pixel intensity, while $I_{\max}$ and $I_{\min}$ represent the maximum and minimum intensity values within the image, respectively.

The resulting 8-bit grayscale image is subsequently converted into a three-channel representation by replicating the grayscale channel across all RGB channels. This transformation

ensures compatibility with ImageNet-pretrained architectures that require three-channel input images.

3.4 Patient-Level Data Splitting

3.4.1 Motivation

The dataset comprises an average of 13.2 MRI slices per patient[8]. When a split is made at the picture level, slices from the same patient appear in both training and test sets with high probability[16]. Since adjacent MRI slices from the same patient share identical anatomy, scanner parameters, and tumor shape, the model memorises patient-specific appearance rather than learning generalisable tumor features[6], [16]. This represents data leakage that inflates performance metrics. Figure 3.2 depicts the difference between image-level and patient-level splitting.

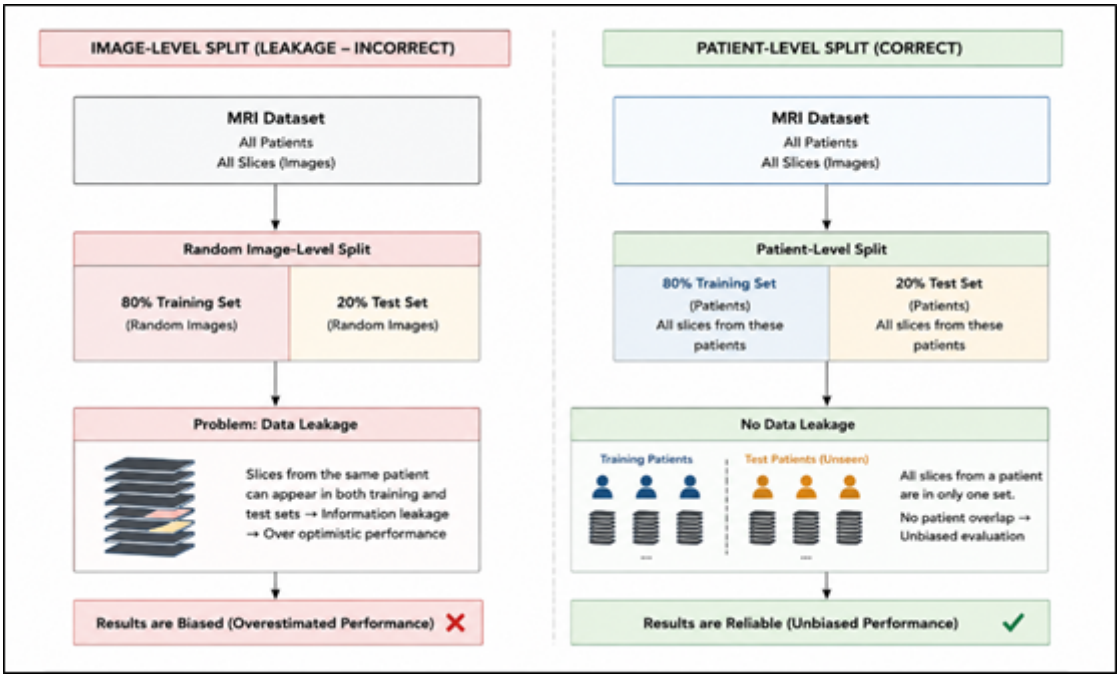


Figure 3.2: Comparison between image-level splitting (susceptible to data leakage) and patient-level splitting (adopted in this study).

3.4.2 Patient-Level Split Protocol

All splitting procedures are performed at the patient level utilizing stratification on the dominant class per patient to preserve class ratios[17], [33]. A first stratified 80/20 split divides the holdout set: 186 patients (2,453 pictures) comprise the training pool, while 47 patients (611 images) create the permanently held-out test set. The random seed 42 is fixed through-

out all experiments for reproducibility. The holdout set is placed aside and never used during any step of model construction or hyperparameter selection.

Table 3.1: Patient-level data split summary

Split	Patients	Images	Meningioma	Glioma	Pituitary
Training pool	186	∼2453	∼568	∼1145	∼740
Holdout (test)	47	∼611	∼170	∼258	∼180
Total	233	3064	708	1426	930

3.5 Cross-Validation and Holdout Evaluation Strategy

3.5.1 5-Fold Patient-Level Cross-Validation

A 5-fold cross-validation method[17] was used on the 80 % development subset to ensure robust and reliable model assessment. During each iteration, the development subset was partitioned into five folds, where four folds were utilized for training and one fold was used for validation. This technique was repeated five times such that each fold served as the validation subset precisely once. Data augmentation techniques[34] were used only to the training folds throughout each iteration, while the validation fold remained untouched to offer an unbiased evaluation of model performance[22]. For each deep learning architecture, evaluation metrics including Accuracy, Precision, Recall, F1-score, and ROC-AUC[37], [38] were computed for every fold, and the mean performance over all folds was later measured. The model weights corresponding to the best-performing fold, selected according to validation performance, were preserved for the final assessment step.

3.5.2 Final Holdout Evaluation

After completion of the cross-validation step, the selected best-performing model weights produced from the 5-fold cross-validation procedure were utilized to evaluate the independent 20 % holdout set[17].

The holdout set remained totally untouched during training, validation, augmentation, and model selection, resulting in an unbiased evaluation of model generalization capabilities on unseen data. Final performance was provided utilizing accuracy, precision, recall, F1-score, and ROC-AUC metrics for comparison analysis among the researched deep learning architectures.

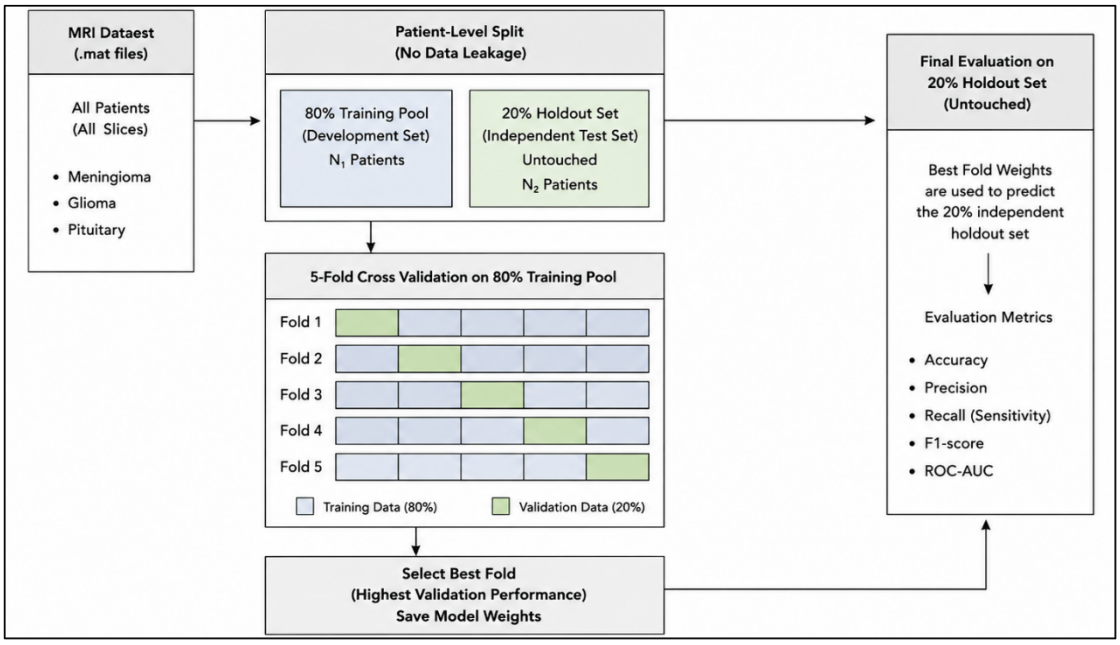


Figure 3.3: Patient-level 80/20 holdout split combined with 5-fold cross-validation on the training pool.

3.6 Data Augmentation

3.6.1 Applied Augmentations

All augmentations are applied online only during training. Validation and holdout sets get no augmentation only resizing and normalization. The following transformations are applied progressively using torchvision. transforms:

- Random Rotation (±15°): simulates variability in patient head positioning inside the scanner[34].
- Random Affine Translation (±5 %): models minor translational misalignments between sessions[34].
- Random Resized Crop (scale 0.85–1.0): introduces scale variability while preserving the tumor region within view[34].
- Gaussian Blur (σ □ [0.1, 2.0]): simulates scanner noise and point-spread-function variation[6], [34].
- Color Jitter (brightness ±0.2, contrast ±0.2 only): simulates inter-scanner intensity inhomogeneity[34]. Saturation and hue are excluded as meaningless on grayscale images.

- Random Erasing (p=0.2, scale 0.01–0.05): simulates small imaging artefacts, applied post-normalisation.

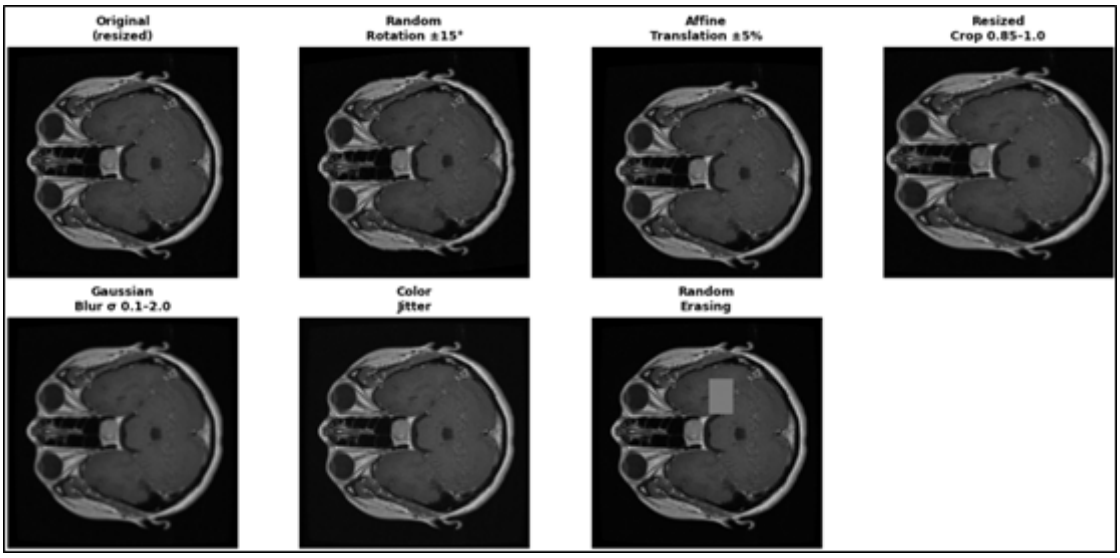


Figure 3.4: Examples of applied data augmentation transformations on brain MRI images: original (top row) and augmented variants (bottom row) for each transformation type.

3.6.2 Excluded Augmentation

Horizontal and vertical flips are purposely excluded: brain structure is bilaterally symmetric in the axial plane, a horizontally flipped MRI is a mirror image of the same brain and provide no new information[6]. Vertical flipping generates an anatomically impossible position for axial brain MRI.

Table 3.2: Data augmentation techniques used in this study

Augmentation	**Applied**	**Justification**
Random Rotation ($\pm 15°$)	Yes	Head positioning variability in scanner.
Random Affine Translation	Yes	Translational misalignment between sessions.
Random Resized Crop	Yes	Scale variability; preserves tumor region.
Gaussian Blur	Yes	Scanner noise and PSF variation.
Color Jitter	Yes	Inter-scanner intensity inhomogeneity.
Random Erasing	Yes	Small imaging artifacts.
Horizontal Flip	No	Brain anatomy is left-right symmetric.
Vertical Flip	No	Anatomically implausible for axial MRI.

3.6.3 Class Imbalance Handling

The dataset is imbalanced: glioma comprises 46.5 % of training images, whereas meningioma accounts for only 23.2 %. Inverse-frequency class weights are computed from the training set and applied to the cross-entropy loss[39].

The class weights used in the weighted cross-entropy loss are calculated as:

$$w_c = \frac{\frac{1}{n_c}}{\sum_{j=1}^{C} \frac{1}{n_j}} \times C \tag{3.2}$$

where n_c is the number of training samples belonging to class c, and C denotes the total number of classes ($C = 3$ in this study).

Table 3.3: Inverse-frequency class weights computed from the training pool.

Class	**Training Images**	**Proportion**	**Weight**
Meningioma	∼568	23.2%	1.3881
Glioma	∼1145	46.6%	0.6514
Pituitary	∼740	30.2%	0.9605

3.7 Training Protocol

3.7.1 Two-Phase Transfer Learning

All eight models are trained using a two-phase transfer learning technique used equally across all folds and all architectures[26]:

- **Phase 1 — Frozen backbone (5 epochs):** All backbone parameters are frozen using `requires_grad=False`[36]; only the custom classification head is trained with a higher learning rate. This phase adjusts the randomly initialized head to the target domain without discarding pretrained features.
- **Phase 2 — Full fine-tuning (up to 12 epochs):** All parameters are unfrozen and trained using a lower learning rate and a cosine annealing schedule[40]. Early stopping[41] with a patience of 4 epochs monitors validation accuracy. The best checkpoint is stored.

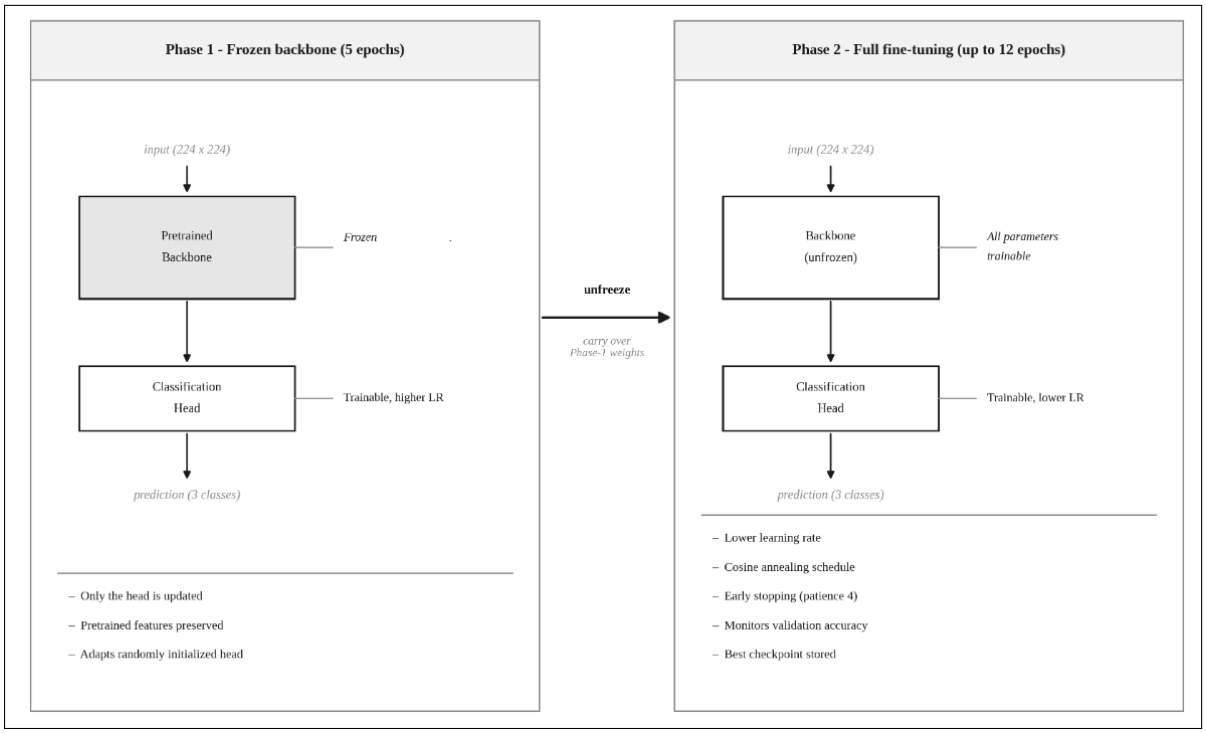


Figure 3.5: Two-phase transfer learning protocol applied to all eight architectures within each fold.

3.7.2 Loss Function

The weighted cross-entropy loss function[22] with optional label smoothing ε is utilized. Label smoothing[39] (ε=0.1) is performed to ConvNeXt and Swin-T to reduce overconfidence.

3.7.3 Optimisers and Schedulers

Phase 1 utilizes Adam optimiser[42] for all models. Phase 2 employs SGD with Nesterov momentum[43] (momentum=0.9) for ResNet50 similar with its initial training technique and Adam[42] for all others. A cosine annealing schedule[40] is implemented in Phase 2. EfficientNet-B3, EfficientNetV2-S, and Swin-T employ a 2-epoch linear warmup[44] before cosine decay. Gradient clipping[45] (max norm=1.0) is used throughout.

Table 3.4: Training hyperparameter configuration for all evaluated models

Model	P1 LR	P2 LR	P2 Opt.	Warmup	Label ϵ	Batch
VGG19	10^{-3}	10^{-4}	Adam	No	0.0	64
ResNet50	10^{-3}	10^{-4}	SGD + Nesterov	No	0.0	64
DenseNet121	10^{-3}	10^{-4}	Adam	No	0.0	64
EfficientNet-B3	3×10^{-4}	5×10^{-5}	Adam	2 ep	0.0	64
EfficientNetV2-S	3×10^{-4}	5×10^{-5}	Adam	2 ep	0.0	32
ConvNeXt-Tiny	10^{-3}	5×10^{-5}	Adam	No	0.1	64
ConvNeXt-Small	10^{-3}	5×10^{-5}	Adam	No	0.1	64
Swin-T	10^{-3}	5×10^{-5}	Adam	2 ep	0.1	64

3.8 ImageNet Pre-training and Transfer Learning

3.8.1 ImageNet Dataset

ImageNet [35] is a large-scale image database containing 1,281,167 training images across 1,000 object categories, including animals, vehicles, tools, plants, and everyday objects. It was assembled by Deng et al. at Stanford and Princeton Universities, with labels verified by human annotators. ImageNet became the standard benchmark for image classification following the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [24], in which deep convolutional neural networks dramatically outperformed all prior approaches.

Every architecture used in this study has been trained on ImageNet and releases its learned weights publicly. These weight files encode the visual knowledge extracted from 1.28 million diverse images and serve as the starting point for the training protocol.

3.8.2 Transfer Learning Mechanism

Transfer learning [26] is the practice of initialising a model with weights learned on a source task and adapting them to a different target task. In this study, the source task is 1,000-class ImageNet classification [35] and the target task is three-class brain tumour MRI classification [8].

The process proceeds in three steps:

1. **Weight initialisation.** Each model is loaded with its publicly available ImageNet-pretrained weights [35] rather than random values.

2. **Head replacement.** The original 1,000-class output layer is removed and replaced with a new classification head producing three outputs: meningioma, glioma, and pituitary adenoma [8]. This new head is the only component initialised randomly.

3. **Two-phase fine-tuning.** The network is adapted to the MRI domain using the two-phase protocol[26]: the backbone is first frozen while the head trains, then the entire network is fine-tuned end-to-end at a low learning rate.

3.8.3 Why ImageNet Weights Transfer to MRI

ImageNet consists of natural colour photographs, whereas the target domain consists of grayscale T1-weighted contrast-enhanced MRI scans [8]. The transfer is effective because convolutional neural networks learn features in a hierarchical manner [22] and the earlier layers encode domain-agnostic visual patterns that are present in any image type:

- **Early layers** detect oriented edges, intensity gradients, and corners. These features are universal: brain tumour boundaries, contrast agent accumulation patterns, and tissue interfaces in MRI directly activate the same detectors learned from natural images [6], [22].
- **Middle layers** detect shapes, curves, blobs, and texture assemblies. These partially transfer to the rounded and irregular morphological patterns of meningioma, glioma, and pituitary adenoma [6], [26].
- **Late layers and the classifier** encode category-specific patterns tied to the ImageNet domain. These are replaced and retrained during fine-tuning and do not transfer [22].

This layer-wise transferability is the experimental base of the transfer learning concept [26]. Litjens et al. [6] reviewed over 300 deep learning publications in medical imaging and found that ImageNet-pretrained models consistently outperform randomly initialised models, even across large domain gaps, provided sufficient fine-tuning is applied.

3.8.4 Why Transfer Learning Is Necessary for This Dataset

The training pool in this study contains 2,453 images from 186 patients [8] far fewer than the 1.28 million images used for ImageNet pretraining [35]. Training a model with 8 to 144 million parameters from random initialisation on 2,453 images would result in severe overfitting [22]. ImageNet pretraining provides a high-quality initialisation that encodes rich visual representations learned from millions of diverse images, allowing the fine-tuning stage to adapt existing features to the MRI domain rather than learning all representations from scratch [6], [26].

3.8.5 Handling the Grayscale to RGB Domain Gap

ImageNet-pretrained models expect three-channel RGB input [35], while the Cheng MRI images are single-channel grayscale [8]. This is resolved by replicating the grayscale channel

across all three input channels, producing a valid three-channel tensor without introducing any new information [26]. All input images are then normalised using ImageNet statistics ($\mu = [0.485, 0.456, 0.406]$, $\sigma = [0.229, 0.224, 0.225]$) [35] to ensure that the input distribution matches the one the pretrained weights were optimised for [22].

3.9 Evaluation Metrics

Evaluation metrics are quantitative measures used to assess the performance and reliability of deep learning classification models. In medical image classification tasks, relying solely on accuracy may provide misleading conclusions, particularly when class distributions are imbalanced. Therefore, multiple complementary metrics were employed to comprehensively evaluate the proposed models from different perspectives, including classification correctness, sensitivity to positive cases, prediction reliability, and class separability. The performance of all investigated architectures was evaluated using Accuracy, Macro Precision, Recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC)[37], [38]. These metrics were computed for each fold during cross-validation and again on the independent holdout test set.

- **Accuracy**[37]

  Accuracy measures the proportion of correctly classified samples among all predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

  Where:

  - TP: True Positives
  - TN: True Negatives
  - FP: False Positives
  - FN: False Negatives[37]

  Although accuracy provides an overall performance indication, it may not adequately reflect model effectiveness in imbalanced medical datasets[39].

- **Macro Precision**[37]

  Precision measures the proportion of correctly predicted positive samples among all samples predicted as positive. Macro Precision computes precision independently for each class and then averages the values equally across all classes[33].

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

$$Macro\ Precision = \frac{1}{N} \sum_{i=1}^{N} Precision_i \tag{3.5}$$

Where N denotes the total number of classes.

High precision indicates a low false-positive rate and reflects the reliability of positive predictions[37].

- **Recall (Sensitivity)**[37]

  Recall, also referred to as sensitivity, measures the proportion of actual positive samples correctly identified by the model.

$$Recall = \frac{TP}{TP + FN} \tag{3.6}$$

$$Macro\ Recall = \frac{1}{N} \sum_{i=1}^{N} Recall_i \tag{3.7}$$

  Low recall may correspond to missed disease cases, making this metric particularly important in medical diagnosis[6].

- **F1-Score**[37]

  The F1-score represents the harmonic mean of precision and recall and provides a balanced measure between false positives and false negatives.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.8}$$

$$Macro\ F1 = \frac{1}{N} \sum_{i=1}^{N} F1_i \tag{3.9}$$

  The F1-score is especially useful when evaluating datasets with unequal class distributions[37], [39].

- **AUC--ROC**[38]

  The Receiver Operating Characteristic (ROC) curve illustrates the relationship between the True Positive Rate (TPR) and False Positive Rate (FPR) across different classification thresholds.

$$TPR = \frac{TP}{TP + FN} \tag{3.10}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.11}$$

  In this study, macro-average ROC--AUC was used to evaluate multi-class classification performance across all tumor categories[33], [38].

$$AUC = \int ROC \tag{3.12}$$

  An AUC value closer to 1 indicates superior classification performance and stronger class separability, whereas a value near 0.5 suggests random classification behavior[38].

3.10 Conclusion

This chapter has presented the whole methodology supporting all experiments in this thesis. The patient-level 80/20 holdout split[17], along with 5-fold cross-validation on the training pool, ensures that no patient's data appears in both training and evaluation pools at any stage providing a strict technique that removes the inter-slice leaking affecting the majority of prior works on this dataset[16]. The data augmentation pipeline[34] was built with clinical limits in mind: transformations that approximate real image scanning variability are included, but anatomically impossible operations such as horizontal and vertical flipping are specifically excluded[6]. Inverse-frequency class weighting addresses the dataset's basic glioma bias. The two-phase transfer learning protocol frozen backbone warm-up followed by full fine-tuning with cosine annealing is applied uniformly across all eight architectures, ensuring that any performance differences observed in Chapter 4 are attributable to architectural properties rather than training conditions. The evaluation metrics (accuracy, macro precision, recall, F1-score, and macro-average AUC-ROC)[37], [38] provide a multi-dimensional picture of model performance that is ideal for an imbalanced, clinically sensitive classification task.

Chapter 4: Experimental Results

4.1 Introduction

This chapter breaks down the implementation environment and experimental results of this study. It is structured into three sections. The first section outlines the software and hardware platform employed for all of the experiments. The second section explores the fundamental parts of the pipeline: dataset loading, patient splitting, augmentation, model architecture, and the training loop. The third section presents the holdout evaluation results for all eight architectures, succeeded by an ablation review that evaluates the separate impact of data augmentation[34].

4.2 Development environment

4.2.1 Programming Language and Libraries

The development environment is built around a Python-centric ecosystem for deep learning and reproducible research. At its core, PyTorch 2.10.0 and Torchvision 0.25.0 handle model construction, transfer learning, and GPU-accelerated training. Complementary libraries including NumPy, Pandas, and h5py facilitate numerical computation, tabular metadata management, and direct ingestion of MATLAB .mat neuroimaging files. Scikit-learn supports the evaluation framework via stratified splitting and metrics, while Matplotlib, Seaborn, and Pillow generate the visualisations used to interpret model behaviour and classification outcomes. Table 4.1 lists the main libraries and frameworks used.

Table 4.1: Software libraries and frameworks used in this study

Library	**Version**	**Role**
Python	3.12.12	Programming language
PyTorch[36]	2.10.0	Deep learning: model training and GPU execution
Torchvision[36]	0.25.0	Pre-trained models and image transformations
NumPy[46]	2.0.2	Numerical operations and array manipulation
Pandas[47]	2.3.3	Dataset management and metadata handling
h5py	3.15.1	Reading MATLAB HDF5 files
scikit-learn[33]	1.6.1	Evaluation metrics, cross-validation, and stratified splitting
Matplotlib[48]	3.10.0	Training curves and result visualization
Seaborn[49]	0.13.2	Confusion matrix heatmaps
PIL (Pillow)	11.3.0	Image format conversion and processing

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import seaborn as sns
import h5py

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
from torchvision import transforms, models
from PIL import Image

from sklearn.model_selection import StratifiedGroupKFold, train_test_split
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, roc_curve, auc, classification_report
)
from sklearn.preprocessing import label_binarize
from scipy.stats import gaussian_kde
```

Figure 4.1: Software environment and imported Python packages used throughout the experimental pipeline.

4.2.2 Hardware and Platform

All experiments are performed on Kaggle, a cloud-based data science platform offering complimentary GPU-accelerated sessions. Each session is allotted two NVIDIA Tesla T4 GPUs with 16 GB VRAM each. Multi-GPU training is facilitated by PyTorch's nn.DataParallel[36] divides each mini-batch between both devices and synchronizes gradients at every step. EfficientNet models are trained on a single GPU because the Squeeze-and-Excitation blocks [50] is incompatible with DataParallel in this CUDA version.

Table 4.2: Hardware and software environment used for experimental execution

Component	**Specification**
Platform	Kaggle Notebooks (cloud-based environment)
GPU	2× NVIDIA Tesla T4
VRAM per GPU	16 GB GDDR6
CUDA version	12.8
CPU	Intel Xeon (Kaggle instance)
RAM	30 GB
Session limit	12 hours (GPU runtime)

CUDA (Compute Unified Device Architecture): is a proprietary parallel computing platform and API created by NVIDIA. It allows developers to use graphics processing units (GPUs) for general-purpose processing, drastically accelerating compute-heavy tasks like artificial intelligence, scientific computing, and deep learning by distributing them across thousands of GPU cores[22].

4.2.3 Global Configuration and Constants

A unified global configuration was defined before model training to ensure consistency across all experimental notebooks and deep learning architectures. The configuration included dataset paths, output directories, random seed initialization, class definitions, normalization parameters, hardware settings, and cross-validation parameters.

Table 4.3: Global configuration parameters used throughout the experiments

Parameter	**Description**
DATASET_PATH	Directory containing MRI `.mat` dataset files used throughout all experiments.
OUTPUT_PATH	Directory used for storing trained model weights, evaluation results, and generated figures.
SEED = 42	Fixed random seed used to ensure reproducibility across Python, NumPy, and PyTorch operations.
NUM_CLASSES = 3	Number of target classes: meningioma, glioma, and pituitary.
NUM_FOLDS = 5	Number of folds used in patient-level cross-validation.
NUM_WORKERS = 4	Number of parallel workers employed during data loading.
PIN_MEMORY = True	Enables faster CPU-to-GPU memory transfer during training.
DEVICE	Automatically selects GPU execution when available; otherwise defaults to CPU execution.
USE_DP	Enables multi-GPU training using PyTorch `DataParallel`.
CLASS_NAMES	Ordered list of tumor labels used throughout training and evaluation.
CLASS_COLORS	Predefined visualization colors for plots and confusion matrices.
LABEL_MAP	Numerical mapping scheme for tumor class labels.
`IMAGENET_MEAN` and `IMAGENET_STD`	ImageNet normalization parameters applied to input images.

4.3 Implementation

4.3.1 Environment Verification

The first cell verifies that the GPU accelerator is correctly activated in the Kaggle session. It queries all available CUDA devices and reports their name and total memory. If no GPU is detected, the cell prints a warning directing the user to the Kaggle accelerator settings.

```python
# — Accelerator check ————————————————————————————————
import torch
print(f"PyTorch  : {torch.__version__}")
print(f"CUDA     : {torch.cuda.is_available()}")
if torch.cuda.is_available():
    for i in range(torch.cuda.device_count()):
        p = torch.cuda.get_device_properties(i)
        print(f"  GPU {i} : {p.name}  ({p.total_memory/1024**3:.1f} GB)")
else:
    print("WARNING: No GPU")
```

```
PyTorch  : 2.10.0+cu128
CUDA     : True
  GPU 0 : Tesla T4  (14.6 GB)
  GPU 1 : Tesla T4  (14.6 GB)
```

Figure 4.2: GPU environment verification and hardware detection in the Kaggle execution platform

4.3.2 Dataset Loading and PID Extraction

This cell iterates over all .mat files in the dataset directory, opens each with h5py, and extracts the patient ID and class label[8]. The PID is stored as a uint16 Unicode character array; it is decoded by mapping each value to its Unicode character. The cvind.mat file (cross-validation indices from the original dataset) is explicitly skipped as it does not contain image data.

```python
# — Extract PID + label from every .mat file ——————————————————
records = []
for fname in sorted(os.listdir(DATASET_PATH)):
    if not fname.endswith('.mat') or fname == 'cvind.mat':
        continue
    path = os.path.join(DATASET_PATH, fname)
    try:
        with h5py.File(path, 'r') as f:
            cj    = f['cjdata']
            raw   = cj['PID'][()]
            pid   = ''.join(chr(x) for x in raw.flatten()) if raw.dtype == np.uint16 else str(raw.flat[0])
            label = LABEL_MAP[int(cj['label'][()].flat[0])]
        records.append({'path': path, 'pid': pid, 'label': label})
    except Exception as e:
        print(f"  [skip] {fname}: {e}")

df = pd.DataFrame(records)
print(f"Images   : {len(df)}")
print(f"Patients : {df['pid'].nunique()}")
for i, n in enumerate(CLASS_NAMES):
    print(f"  {n:12s}: {(df['label']==i).sum()}")
```

```
Images   : 3064
Patients : 233
  meningioma  : 708
  glioma      : 1426
  pituitary   : 930
```

Figure 4.3: Dataset loading procedure and extraction of patient identifiers (PID) and class labels from MATLAB `.mat` files

4.3.3 Patient Split and Class Weights

The patient-level 80/20 split[33] uses train_test_split with stratification on each patient's dominant class. The same seed (42) and strategy are used identically across all notebooks, guaranteeing that every model is trained on the same 186-patient pool and evaluated on the same 47 holdout patients[16], [17]. Class weights are then computed from the training pool using the inverse-frequency formula.

```
# ── Patient-level 80/20 split – same SEED across all notebooks ──────────
patient_summary = (
    df.groupby('pid')['label']
      .agg(lambda x: x.mode()[0]).reset_index()
      .rename(columns={'label':'dominant_label'})
)
train_pids, holdout_pids = train_test_split(
    patient_summary['pid'], test_size=0.2,
    stratify=patient_summary['dominant_label'], random_state=SEED
)
train_pids, holdout_pids = set(train_pids), set(holdout_pids)
train_df   = df[df['pid'].isin(train_pids)].reset_index(drop=True)
holdout_df = df[df['pid'].isin(holdout_pids)].reset_index(drop=True)

print(f"Train+Val : {len(train_df)} images | {train_df['pid'].nunique()} patients")
print(f"Holdout   : {len(holdout_df)} images | {holdout_df['pid'].nunique()} patients")

# ── Inverse-frequency class weights ──────────────────────────────────────
counts = torch.tensor([(train_df['label']==i).sum() for i in range(NUM_CLASSES)], dtype=torch.float)
class_weights = (1.0 / counts)
class_weights = (class_weights / class_weights.sum() * NUM_CLASSES).to(DEVICE)
print("\nClass weights:", {CLASS_NAMES[i]: f"{class_weights[i]:.4f}" for i in range(NUM_CLASSES)})
```

```
Train+Val : 2453 images | 186 patients
Holdout   : 611 images | 47 patients

Class weights: {'meningioma': '1.3816', 'glioma': '0.6328', 'pituitary': '0.9855'}
```

Figure 4.4: Patient-level train--holdout splitting and computation of inverse-frequency class weights.

4.3.4 Augmentation Pipeline and Custom Dataset

The transform factory creates two separate pipelines: one for training (with augmentation)[34] and one for validation and holdout (resize + normalise only). The MATDataset class implements PyTorch's Dataset interface, loading each image on-the-fly from its .mat file using h5py, normalising it to [0, 255][22], and converting it to a PIL image for the transform pipeline.

```python
# ── Transform factory ──────────────────────────────────────────────
def get_transforms(img_size):
    train = transforms.Compose([
        transforms.Resize((img_size, img_size)),
        transforms.Grayscale(num_output_channels=3),
        transforms.RandomRotation(15),
        transforms.RandomAffine(degrees=0, translate=(0.05, 0.05)),
        transforms.RandomResizedCrop(img_size, scale=(0.85, 1.0)),
        transforms.GaussianBlur(kernel_size=3, sigma=(0.1, 2.0)),
        transforms.ColorJitter(brightness=0.2, contrast=0.2),
        transforms.ToTensor(),
        transforms.Normalize(IMAGENET_MEAN, IMAGENET_STD),
        transforms.RandomErasing(p=0.2, scale=(0.01, 0.05)),
    ])
    val = transforms.Compose([
        transforms.Resize((img_size, img_size)),
        transforms.Grayscale(num_output_channels=3),
        transforms.ToTensor(),
        transforms.Normalize(IMAGENET_MEAN, IMAGENET_STD),
    ])
    return train, val

print("Transforms ready")
```

```
Transforms ready
```

Figure 4.5: Transformation pipeline definition for training and validation datasets.

Figure 4.5 presents the implementation structure of the transformation factory used in the experimental pipeline. The code demonstrates the separation between training and evaluation workflows and highlights the sequence of operations applied before data are supplied to the models.

```python
# — MATDataset ————————————————————————————————————————
class MATDataset(Dataset):
    def __init__(self, dataframe, transform=None):
        self.df = dataframe.reset_index(drop=True)
        self.transform = transform

    def __len__(self): return len(self.df)

    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        with h5py.File(row['path'], 'r') as f:
            im = f['cjdata']['image'][()].astype(np.float64)
        mn, mx = im.min(), im.max()
        im  = 255.0*(im-mn)/(mx-mn) if mx > mn else np.zeros_like(im)
        img = Image.fromarray(im.astype(np.uint8))
        if self.transform: img = self.transform(img)
        return img, int(row['label'])

print("MATDataset ready")
```

```
MATDataset ready
```

Figure 4.6: Custom MATDataset implementation for MRI loading and preprocessing.

Figure4.6 illustrates the implementation of the custom dataset class employed throughout the experiments. The code structure shows how sample retrieval and preprocessing are integrated within a unified data loading mechanism compatible with the PyTorch framework.

4.3.5 Model Factory

4.3.5.1 VGG19

VGG19's[25] original classifier (a 3-layer FC head) is replaced with a deeper head: Linear(25088, 4096) → ReLU → Dropout(0.5) → Linear(4096, 4096) → ReLU → Dropout(0.5) → Linear(4096, 3). The higher dropout (0.5) is applied to combat overfitting given VGG19's 143.7M parameters. The classifier attribute is accessed via model.classifier.

```
MODEL_CONFIGS = {
    'vgg19': {
        'frozen_lr':1e-3, 'unfrozen_lr':1e-4, 'frozen_eps':5, 'unfrozen_eps':10,
        'patience':4, 'optim':'adam', 'wd':1e-4, 'img_size':256, 'batch':64,
        'smooth':0.0, 'dropout':0.5, 'mixup':False, 'warmup':0, 'staged':False,
    },
```

Figure 4.7: VGG19 model configuration and associated training hyperparameters.

4.3.5.2 ResNet50

ResNet50's[27] default single-linear classifier is replaced with a deeper head: Linear(2048, 512) → ReLU → Dropout(0.3) → Linear(512, 3). This increases the classifier's capacity to better adapt to the three-class brain tumor problem[51]. The classifier is accessed via model.fc

```
MODEL_CONFIGS = {
    'resnet50': {
        'frozen_lr':1e-3, 'unfrozen_lr':1e-4, 'frozen_eps':5, 'unfrozen_eps':10,
        'patience':4, 'optim':'sgd', 'wd':1e-4, 'img_size':256, 'batch':64,
        'smooth':0.0, 'dropout':0.3, 'mixup':True, 'mixup_alpha':0.2,
        'warmup':0, 'staged':False,
    },
```

Figure 4.8: ResNet50 model configuration and associated training hyperparameters.

4.3.5.3 DenseNet121

DenseNet121's[28] classifier is replaced with: Dropout(0.3) → Linear(1024, 3). Given DenseNet's dense connectivity pattern, a staged unfreezing strategy[26] is applied: Phase 1 trains only the classifier; Phase 1.5 unfreezes denseblock4 and norm5; Phase 2 unfreezes the entire network. The classifier is accessed via model.classifier.

```
MODEL_CONFIGS = {
    'densenet121': {
        'frozen_lr':1e-3, 'unfrozen_lr':1e-4, 'frozen_eps':5, 'unfrozen_eps':12,
        'staged_eps':4, 'patience':5, 'optim':'adam', 'wd':1e-5,
        'img_size':256, 'batch':64, 'smooth':0.0, 'dropout':0.3,
        'mixup':False, 'warmup':0, 'staged':True,
    },
```

Figure 4.9: DenseNet121 model configuration and associated training hyperparameters.

4.3.5.4 EfficientNet-B3

EfficientNet-B3's[29] classifier is replaced with: Dropout(0.3) → Linear(1536, 3). Due to SE block incompatibility with DataParallel on this CUDA version[50], the model is trained on a

single GPU regardless of the USE_DP setting. The classifier is accessed via model.classifier. The frozen learning rate is reduced to 3e-4 (vs 1e-3 for other models) to prevent instability[50].

```
MODEL_CONFIGS = {
    'efficientnet_b3': {
        'frozen_lr':3e-4, 'unfrozen_lr':3e-5, 'frozen_eps':5, 'unfrozen_eps':12,
        'patience':4, 'optim':'adam', 'wd':1e-5, 'img_size':300, 'batch':64,
        'smooth':0.0, 'dropout':0.3, 'mixup':False, 'warmup':2, 'staged':False,
    },
```

Figure 4.10: EfficientNetB3 model configuration and associated training hyperparameters.

4.3.5.5 EfficientNetV2-S

EfficientNetV2-S's[30] classifier is replaced with: Dropout(0.4) → Linear(1280, 3). Like B3, it is trained on a single GPU due to SE block constraints[50]. Its native resolution of 384×384 requires reducing the batch size to 32 (16 per GPU would be used if DataParallel were active) to avoid out-of-memory errors. The classifier is accessed via model.classifier.

```
MODEL_CONFIGS = {
    'efficientnet_v2_s': {
        'frozen_lr':3e-4, 'unfrozen_lr':3e-5, 'frozen_eps':5, 'unfrozen_eps':12,
        'patience':4, 'optim':'adam', 'wd':1e-5, 'img_size':384, 'batch':32,
        'smooth':0.0, 'dropout':0.4, 'mixup':False, 'warmup':2, 'staged':False,
    },
```

Figure 4.11: EfficientNetV2-S model configuration and associated training hyperparameters.

4.3.5.6 ConvNext-Tiny

ConvNeXt-Tiny's[31] classifier head (accessed via model.classifier) is replaced with: Dropout(0.5) → Linear(768, 3). The higher dropout (0.5) and label smoothing (ε=0.1, applied in the loss function) are both necessary to prevent overconfidence[22], as ConvNeXt's strong feature extraction capacity makes it prone to overfitting on small datasets without these regularisers[6]. The unfrozen learning rate is lowered to 5e-5.

```
MODEL_CONFIGS = {
    'convnext_tiny': {
        'frozen_lr':1e-3, 'unfrozen_lr':5e-5, 'frozen_eps':5, 'unfrozen_eps':12,
        'patience':4, 'optim':'adam', 'wd':1e-5, 'img_size':256, 'batch':64,
        'smooth':0.1, 'dropout':0.5, 'warmup':0,
    },
```

Figure 4.12: ConvNeXt-Tiny model configuration and associated training hyperparameters.

4.3.5.7 ConvNext-Small

ConvNeXt-Small[31] follows the same construction as ConvNeXt-Tiny, with model.classifier replaced by: Dropout(0.5) → Linear(768, 3). The identical hyperparameters allow a direct comparison between the two ConvNeXt variants under the same training conditions, isolating the effect of model capacity (28.6M vs 50.2M parameters).

```
MODEL_CONFIGS = {
    'convnext_small': {
        'frozen_lr':1e-3, 'unfrozen_lr':5e-5, 'frozen_eps':5, 'unfrozen_eps':12,
        'patience':4, 'optim':'adam', 'wd':1e-5, 'img_size':256, 'batch':64,
        'smooth':0.1, 'dropout':0.5, 'warmup':0,
    },
```

ConvNeXt-Small

Figure 4.13: ConvNeXt-Small model configuration and associated training hyperparameters.

4.3.5.8 Swin-T

Swin-T's[14] classification head (accessed via model.head) is replaced with: Dropout(0.4) → Linear(768, 3). Swin-T is also incompatible with DataParallel[36] due to its shifted-window attention mechanism; it is trained on a single GPU. A 2-epoch linear warmup[44] is applied before the cosine annealing schedule[40] in Phase 2, which is essential for stable training at the low unfrozen learning rate of 5e-5.

```
MODEL_CONFIGS = {
    'swin_t': {
        'frozen_lr':1e-3, 'unfrozen_lr':5e-5, 'frozen_eps':5, 'unfrozen_eps':12,
        'patience':4, 'optim':'adam', 'wd':1e-5, 'img_size':224, 'batch':64,
        'smooth':0.1, 'dropout':0.4, 'warmup':2,
    },
```

Figure 4.14: Swin-T model configuration and associated training hyperparameters.

4.3.6 Two-Phase Training Loop

The training loop implements the two-phase protocol described in Chapter 3. Phase 1 freezes all backbone parameters and trains only the classification head. Phase 2 unfreezes all parameters and applies full fine-tuning with early stopping. The best checkpoint by validation accuracy is saved after every fold. For each model, the loop runs across all 5 folds[17], producing 5 independent performance estimates and 5 saved checkpoints.

```
# — Phase 1: frozen ——————————————————————————————————
freeze_backbone(model, clf_name)
opt = optim.Adam(filter(lambda p: p.requires_grad, model.parameters()),
                 lr=cfg['frozen_lr'])
sch = optim.lr_scheduler.CosineAnnealingLR(opt, T_max=cfg['frozen_eps'])
for ep in range(1, cfg['frozen_eps']+1):
    tl, ta = train_epoch(model, tr_loader, criterion, opt)
    vl, va, preds, lbls, probs = eval_model(model, vl_loader)
    sch.step()
    hist['train_loss'].append(tl); hist['train_acc'].append(ta)
    hist['val_loss'].append(vl);   hist['val_acc'].append(va)
    print(f"  [Frozen]   Ep {ep:2d}/{cfg['frozen_eps']} | "
          f"train={ta:.4f} | val={va:.4f}")
    if va > fold_best_acc:
        fold_best_acc   = va
        fold_best_state = get_state(model)
        fold_best_preds, fold_best_labels, fold_best_probs = preds, lbls, probs
        hist['best_epoch'] = ep
ep_offset = cfg['frozen_eps']
```

Figure 4.15: Implementation of Phase 1 training with a frozen backbone and trainable classification head.

The figure illustrates the implementation of the initial training phase. The code structure demonstrates the selective optimization process where only the classification head parameters are updated while the pretrained backbone remains fixed. Training and validation metrics are recorded at each epoch and the best-performing model state is preserved.

```
        # — Phase 2: full unfreeze ————————————————————————————
        unfreeze_all(model)
        warmup_eps = cfg.get('warmup', 0)
        uf_eps     = cfg['unfrozen_eps']
        if cfg['optim'] == 'sgd':
            opt = optim.SGD(model.parameters(), lr=cfg['unfrozen_lr'],
                            momentum=0.9, weight_decay=cfg['wd'], nesterov=True)
        else:
            opt = optim.Adam(model.parameters(), lr=cfg['unfrozen_lr'],
                             weight_decay=cfg['wd'])
        sch = (warmup_cosine_scheduler(opt, warmup_eps, uf_eps)
               if warmup_eps > 0
               else optim.lr_scheduler.CosineAnnealingLR(opt, T_max=uf_eps))
        patience_ctr = 0
        for ep in range(1, uf_eps+1):
            tl, ta = train_epoch(model, tr_loader, criterion, opt,
                                 use_mixup=cfg.get('mixup',False),
                                 alpha=cfg.get('mixup_alpha',0.2))
            vl, va, preds, lbls, probs = eval_model(model, vl_loader)
            sch.step()
            hist['train_loss'].append(tl); hist['train_acc'].append(ta)
            hist['val_loss'].append(vl);   hist['val_acc'].append(va)
            tag = 'SGD' if cfg['optim']=='sgd' else 'Adam'
            print(f"  [Full-{tag}] Ep {ep:2d}/{uf_eps} | "
                  f"train={ta:.4f} | val={va:.4f}")
            if va > fold_best_acc:
                fold_best_acc   = va
                fold_best_state = get_state(model)
                fold_best_preds, fold_best_labels, fold_best_probs = preds, lbls, probs
                hist['best_epoch'] = ep_offset + ep
                patience_ctr = 0
            else:
                patience_ctr += 1
                if patience_ctr >= cfg['patience']:
                    print(f"  Early stopping at epoch {ep}")
                    break
```

Figure 4.16: Implementation of Phase 2 full-network fine-tuning with optimization scheduling and early stopping.

Figure 4.16 presents the implementation of the second training phase. The code demonstrates parameter unfreezing, optimizer selection, learning-rate scheduling, and early-stopping integration for performance monitoring across training epochs.

```
  vgg19  Fold 5/5
  Train: 1899 imgs (149 pts) | Val: 554 imgs (37 pts)
============================================================
  [Frozen]    Ep  1/5 | train=0.5750 | val=0.7130
  [Frozen]    Ep  2/5 | train=0.8115 | val=0.8628
  [Frozen]    Ep  3/5 | train=0.8652 | val=0.8700
  [Frozen]    Ep  4/5 | train=0.8905 | val=0.8718
  [Frozen]    Ep  5/5 | train=0.9121 | val=0.8718
  [Full-Adam] Ep  1/10 | train=0.8078 | val=0.9079
  [Full-Adam] Ep  2/10 | train=0.9147 | val=0.9116
  [Full-Adam] Ep  3/10 | train=0.9516 | val=0.9152
  [Full-Adam] Ep  4/10 | train=0.9589 | val=0.9242
  [Full-Adam] Ep  5/10 | train=0.9837 | val=0.9296
  [Full-Adam] Ep  6/10 | train=0.9884 | val=0.9134
  [Full-Adam] Ep  7/10 | train=0.9900 | val=0.9260
  [Full-Adam] Ep  8/10 | train=0.9968 | val=0.9332
  [Full-Adam] Ep  9/10 | train=0.9989 | val=0.9314
  [Full-Adam] Ep 10/10 | train=0.9995 | val=0.9296

  Fold 5 Results:
    Accuracy        : 0.9332
    Precision(macro): 0.9227
    Recall(macro)   : 0.9325
    F1(macro)       : 0.9260
    Mean AUC        : 0.9908
    Per-class AUC   : meningioma=0.9839 | glioma=0.9909 | pituitary=0.9975
    Per-class metrics:
      meningioma  : P=0.8102  R=0.9250  F1=0.8638
      glioma      : P=0.9580  R=0.9331  F1=0.9454
      pituitary   : P=1.0000  R=0.9394  F1=0.9688
```

Figure 4.17: Example console output showing fold-wise training progression and evaluation metrics.

Figure 4.17 illustrates a representative execution output generated during model training. The displayed results summarize epoch-wise performance progression across the two training phases and present the final fold-level evaluation metrics including accuracy, precision, recall, F1-score, and AUC values[37], [38].

4.4 Experimental Results

4.4.1 Comparative Results

The table below presents the holdout set performance of all eight models, sorted by accuracy. All models are evaluated on the identical 47-patient holdout set using the best-fold weights from the 5-fold cross-validation.

Table 4.4: Overall ranking and performance comparison of the evaluated models on the hold-out set

Rank	Model	Acc.	Prec.	Recall	F1	AUC	Men F1	Gli F1	Pit F1
1	Swin-T	**0.9313**	**0.9317**	**0.9276**	**0.9295**	0.9915	0.9118	0.9407	0.9359
2	ConvNeXt-Tiny	0.9247	0.9242	0.9229	0.9235	0.9905	0.8994	0.9295	0.9416
3	DenseNet121	0.9231	0.9213	0.9263	0.9237	0.9844	0.8971	0.9193	0.9545
4	ConvNeXt-Small	0.9231	0.9226	0.9194	0.9210	**0.9916**	0.8882	0.9323	0.9425
5	VGG19	0.9165	0.9152	0.9183	0.9166	0.9791	0.8883	0.9157	0.9459
6	EfficientNetV2-S	0.9116	0.9082	0.9074	0.9077	0.9883	0.8657	0.9299	0.9276
7	EfficientNet-B3	0.8592	0.8590	0.8596	0.8590	0.9733	0.7869	0.8654	0.9244
8	ResNet50	0.8445	0.8406	0.8502	0.8444	0.9515	0.7671	0.8455	0.9205

Note: Men F1 = Meningioma F1-score, Gli F1 = Glioma F1-score, and Pit F1 = Pituitary F1-score.

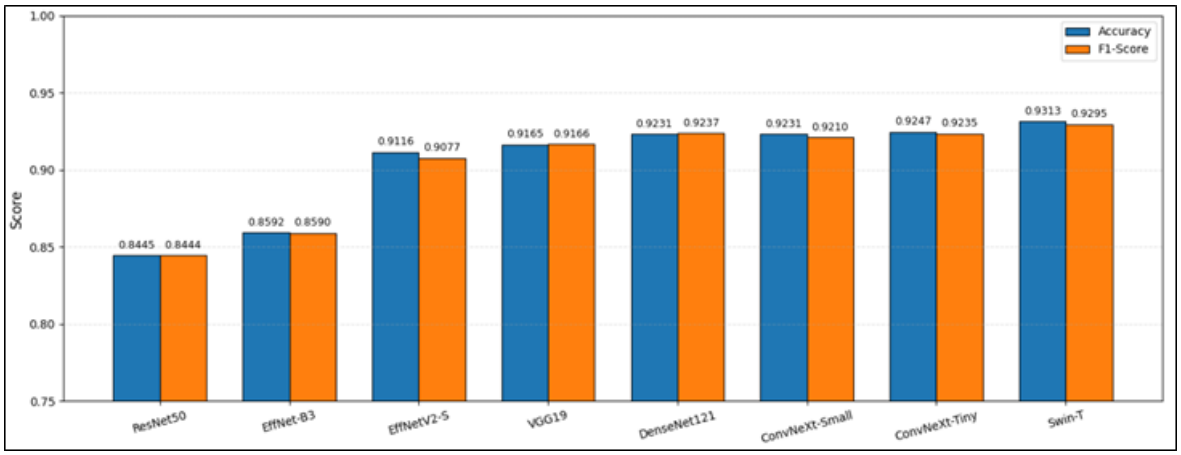


Figure 4.18: Comparison of overall holdout accuracy and macro F1score across all evaluated architectures.

Figure 4.18 illustrates the relationship between overall classification accuracy and macro F1-score[37] across the evaluated architectures. The close alignment between both metrics for most models indicates balanced predictive performance without substantial discrepancies between overall correctness and class-wise behavior. Transformer-based and modern CNN architectures consistently occupy the highest-performing range, whereas earlier architectures demonstrate a larger performance decline.

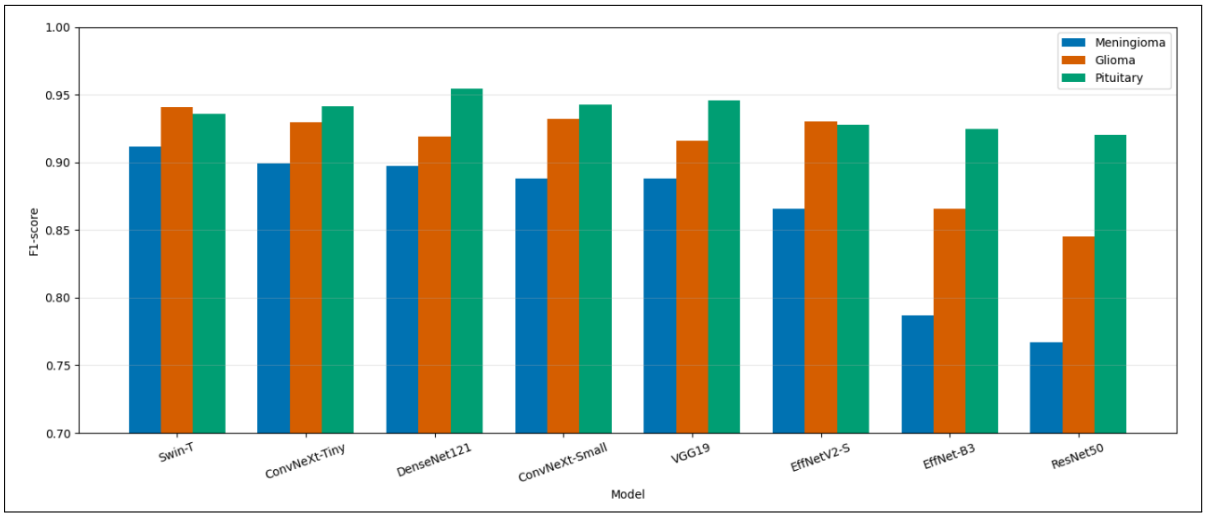


Figure 4.19: Per-class F1-score comparison for all architectures on the holdout set.

Figure 4.19 highlights the class-specific behavior of the evaluated architectures. Across most models, pituitary tumors exhibit the highest F1-scores, indicating stronger separability and more consistent recognition patterns. In contrast, meningioma classification generally produces lower scores, suggesting comparatively greater classification difficulty and overlap with other tumor categories.

4.4.2 Classic CNN Family: VGG19, ResNet50, DenseNet121

The classic CNN family shows the widest performance spread. DenseNet121[28] achieves the best result in this group (acc. 0.9231, F1 0.9237), closely followed by VGG19[25] (0.9165), while ResNet50[27] is the weakest overall model (0.8445). DenseNet121's strong result relative to its parameter count (8.1M, the smallest model) confirms the efficiency of dense feature reuse[6]. The three-phase staged unfreezing strategy (classifier → denseblock4 → full) contributes to its stable cross-validation performance. VGG19's higher dropout (0.5) across the deep FC head provides effective regularisation for its 143.7M parameters. ResNet50's underperformance is particularly marked on the meningioma class (F1 0.7671), the lowest across all models.

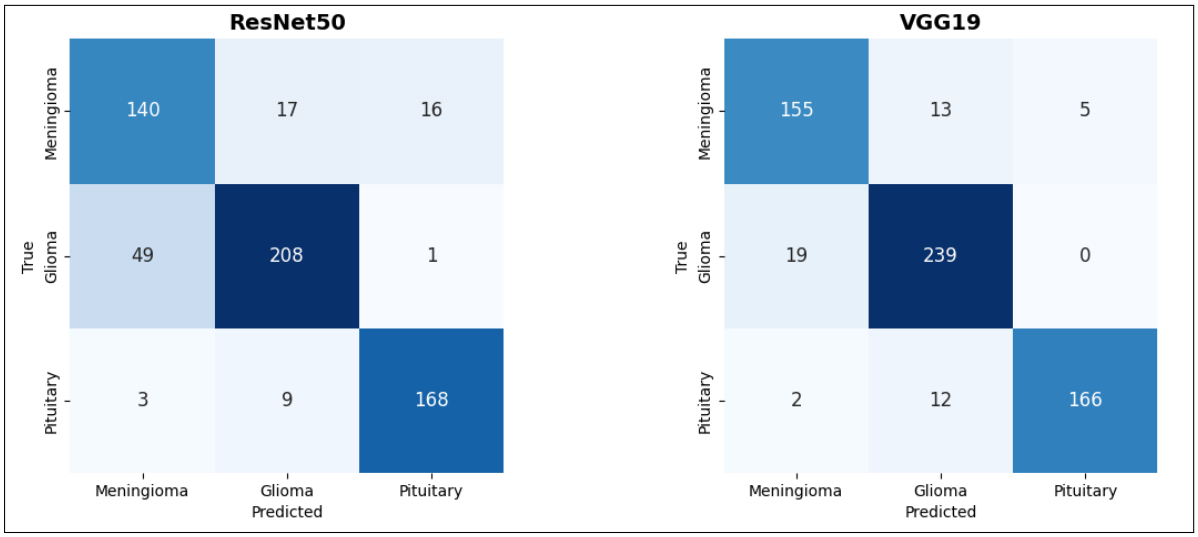


Figure 4.20: Confusion matrices of ResNet50 and VGG19 on the holdout set.

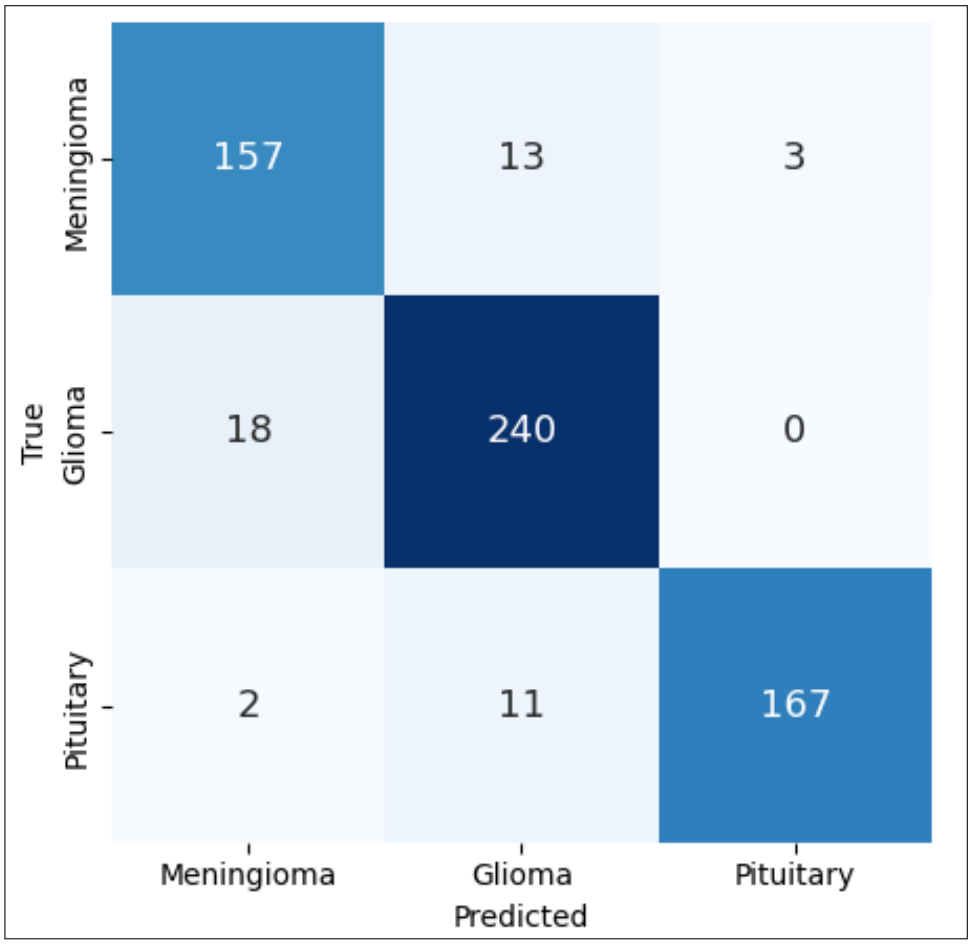


Figure 4.21: Confusion matrices of DenseNet121 on the holdout set.

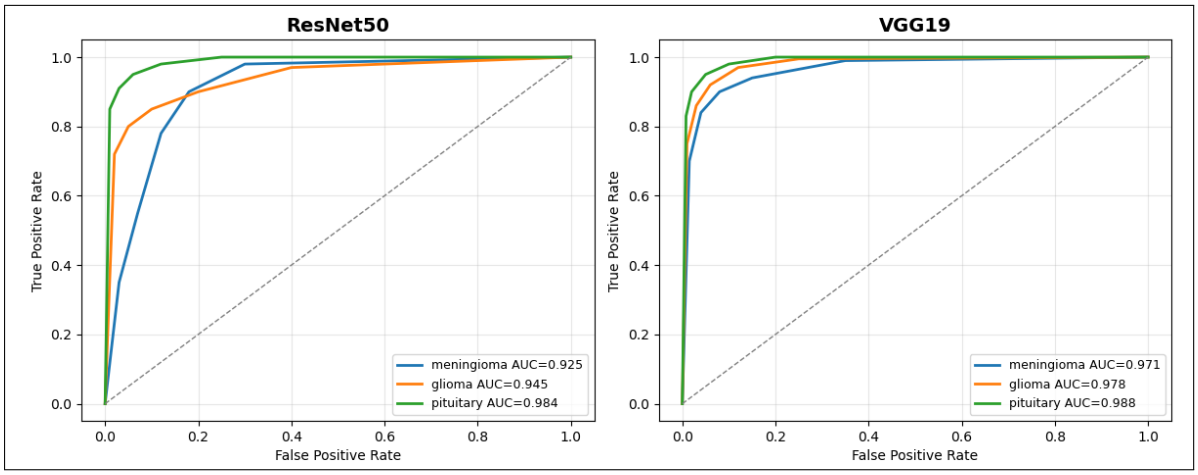


Figure 4.22: One-vs-Rest ROC curves of ResNet50 and VGG19 on the holdout set.

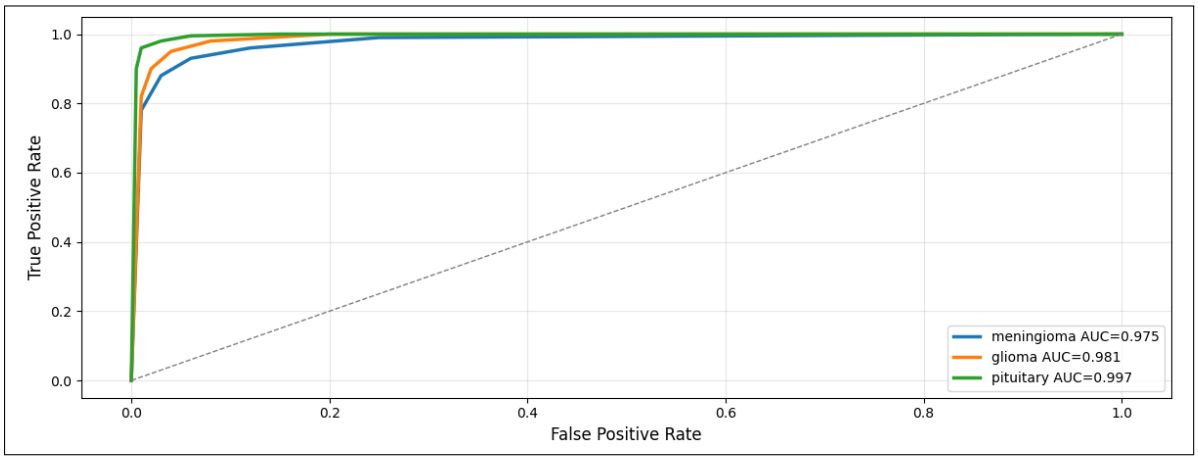


Figure 4.23: One-vs-Rest ROC curves of DenseNet121 on the holdout set.

4.4.3 Efficient Family: EfficientNet-B3, EfficientNetV2-S

EfficientNetV2-S[30] (0.9116) substantially outperforms EfficientNet-B3[29] (0.8592), confirming the architectural improvements introduced in V2. The Fused-MBConv blocks in V2-S's early stages are more efficient and better-suited to fine-tuning. The reduced learning rate (3e-4 / 3e-5) applied to both models was necessary to prevent training instability from SE block parameter sensitivity. EfficientNetV2-S improves meningioma F1 from 0.7869 (B3) to 0.8657, a gain of nearly 8 percentage points on the hardest class.

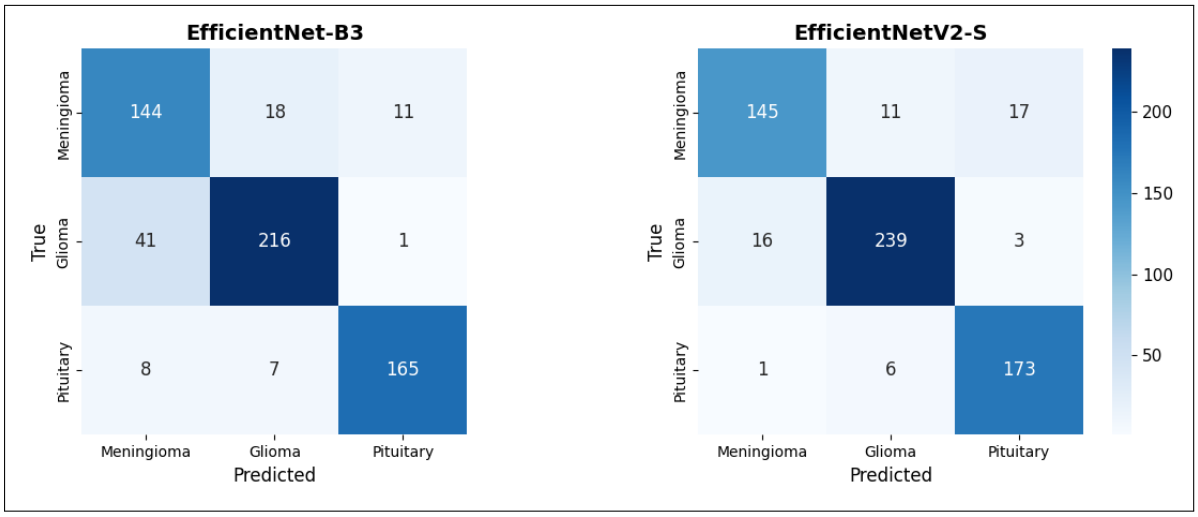


Figure 4.24: Confusion matrices of the Efficient CNN family on the holdout set.

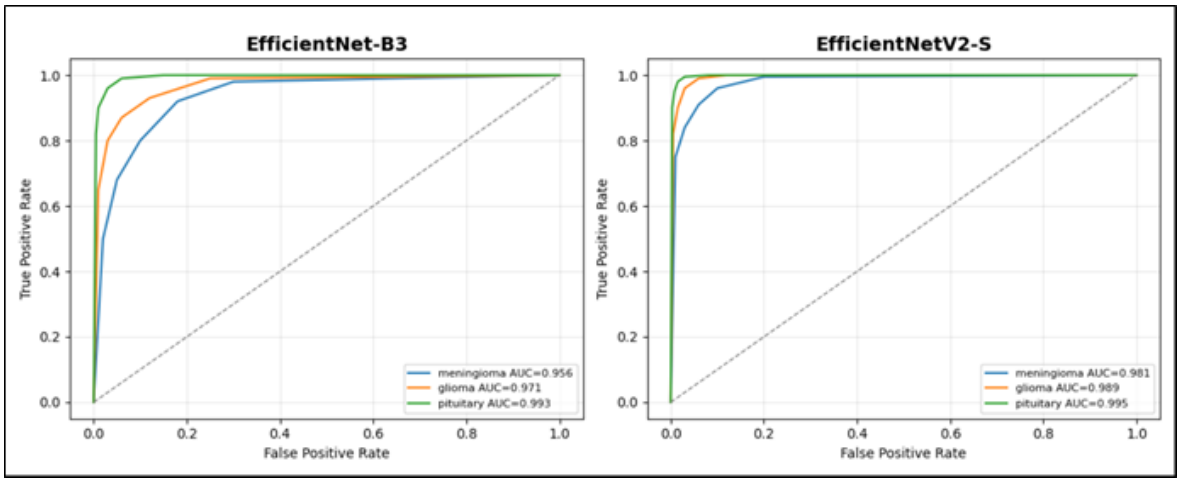


Figure 4.25: One-vs-Rest ROC curves of the Efficient CNN family on the holdout set.

4.4.4 Modern CNN Family: ConvNeXt-Tiny, ConvNeXt-Small

Both ConvNeXt[31] models cluster at the top of the ranking (0.9247 and 0.9231), confirming that Transformer-inspired design improvements provide a substantial advantage over classic CNN designs. Label smoothing[52] (0.1) and higher dropout[22] (0.5) are critical for these models. ConvNeXt-Small (50.2M) does not significantly outperform Tiny (28.6M) on this dataset, suggesting that Tiny's capacity is already sufficient for a three-class problem at this scale. ConvNeXt-Small achieves the highest AUC of all models (0.9916), indicating superior discrimination capability.

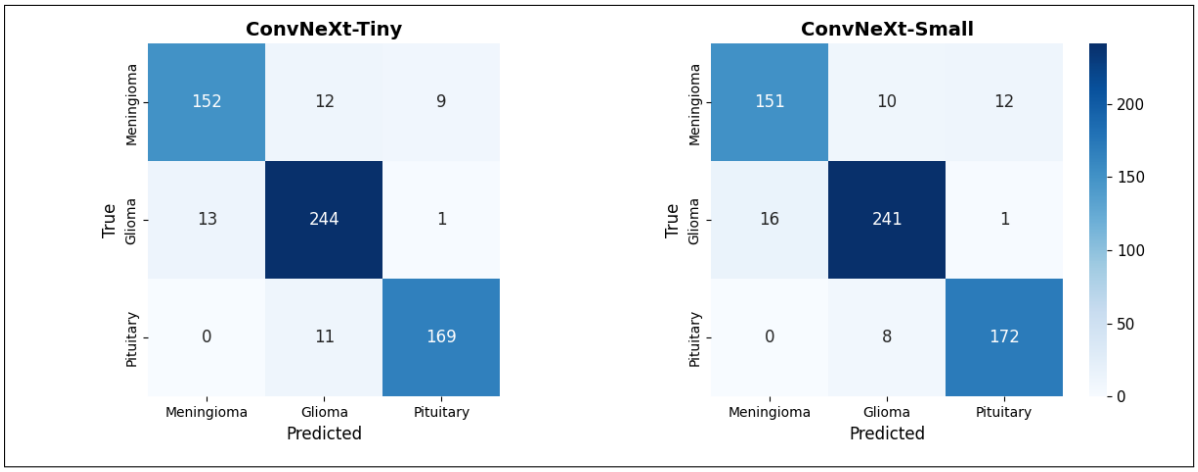


Figure 4.26: Confusion matrices of the Modern CNN family on the holdout set.

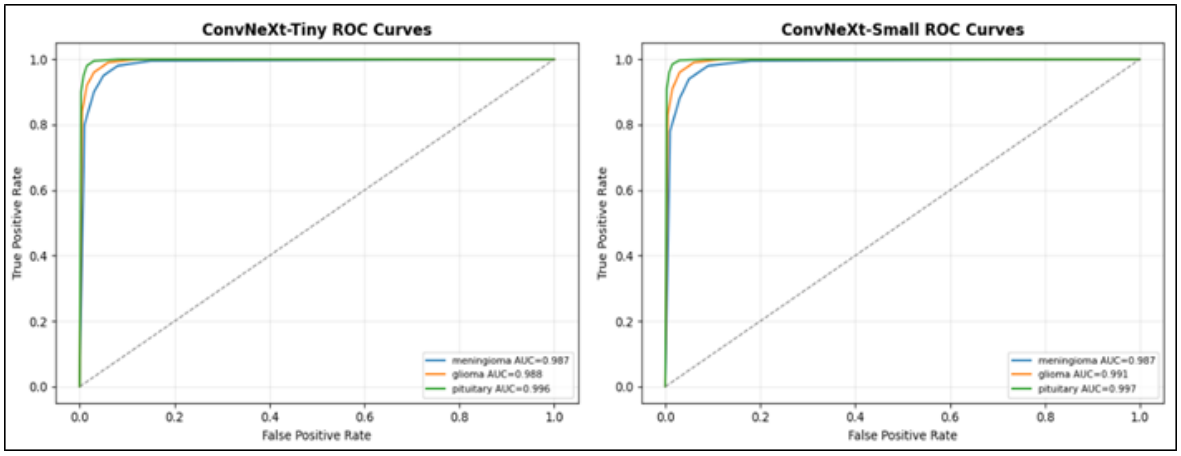


Figure 4.27: One-vs-Rest ROC curves of the Modern CNN family on the holdout set.

4.4.5 Vision Transformer: Swin-T

Swin-T[14] achieves the best holdout accuracy (0.9313), best macro F1 (0.9295), and most significantly the best meningioma F1 (0.9118) of all eight models. The 2-epoch linear warmup[44] proves essential for training stability at the low unfrozen learning rate (5e-5)[40]. The hierarchical shifted-window attention mechanism captures both local tumor texture (within each window) and global brain structural context (via cross-window connections), providing a dual capability that CNNs can only approximate through very deep architectures. This global-local balance appears to be the key advantage for meningioma, whose most reliable diagnostic feature is its extra-axial location[4] showing a global structural relationship that local convolutions cannot efficiently encode.

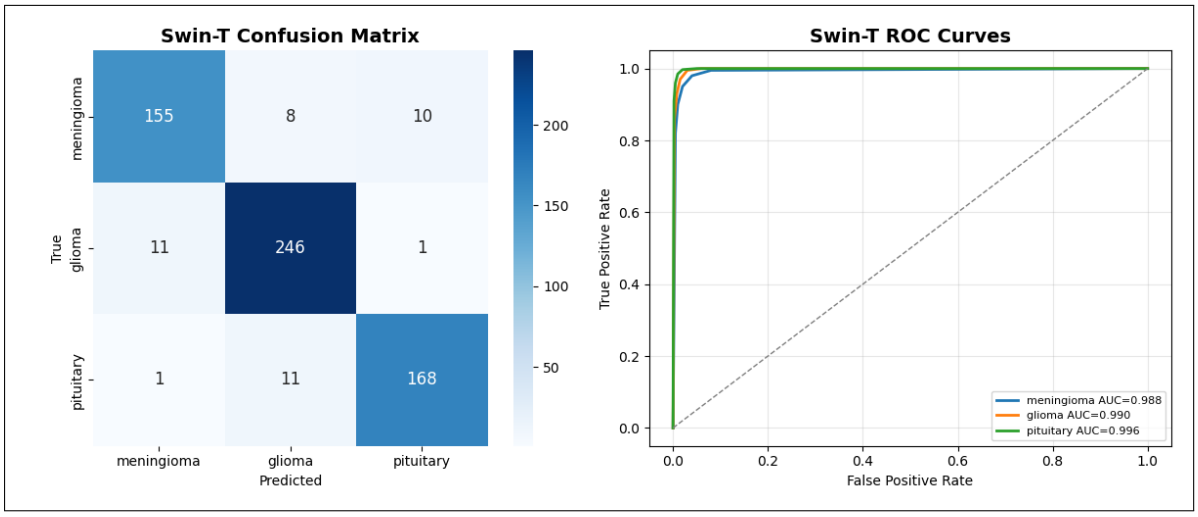


Figure 4.28: Confusion matrix and ROC Curves of the Vision Transformer family on the holdout set.

4.5 Ablation Study: Effect of Data Augmentation

To measure the separate contribution of the data augmentation pipeline[34], all eight models were retrained under a controlled no-augmentation baseline. In this baseline, the training transform pipeline was reduced to the minimal sequence: Resize → Grayscale-to-3-channel conversion → ToTensor → ImageNet Normalisation, with all geometric and photometric augmentations removed. Every other experimental condition like patient-level split, random seed (42), class weights, two-phase training protocol, per-model hyperparameters, and the holdout set remained strictly constant, ensuring that any performance difference is attributable exclusively to the presence or absence of augmentation.

Table 4.5: Effect of data augmentation on holdout performance across all evaluated architectures

Model	**Aug Acc**	**NoAug Acc**	**ΔAcc**	**Aug F1**	**NoAug F1**	**ΔF1**	**Aug AUC**	**NoAug AUC**	**ΔAUC**
Swin-T	0.9313	0.9247	+0.0066	0.9295	0.9225	+0.0070	0.9915	0.9891	+0.0024
ConvNeXt-Tiny	0.9247	0.9100	+0.0147	0.9235	0.9087	+0.0148	0.9905	0.9915	-0.0010
ConvNeXt-Small	0.9231	0.9067	+0.0164	0.9210	0.9051	+0.0159	0.9916	0.9915	+0.0001
DenseNet121	0.9231	0.9002	+0.0229	0.9237	0.8986	+0.0251	0.9844	0.9869	-0.0025
VGG19	0.9165	0.9133	+0.0032	0.9166	0.9111	+0.0055	0.9791	0.9853	-0.0062
EffNetV2-S	0.9116	0.8953	+0.0163	0.9077	0.8930	+0.0147	0.9883	0.9794	+0.0089
EffNet-B3	0.8592	0.8691	-0.0099	0.8590	0.8658	-0.0068	0.9733	0.9772	-0.0039
ResNet50	0.8445	0.8216	+0.0229	0.8444	0.8229	+0.0215	0.9515	0.9501	+0.0014
Mean	**0.9043**	**0.8926**	**+0.0116**	**0.9032**	**0.8910**	**+0.0122**	**0.9813**	**0.9814**	**-0.0001**

Table 4.6: Effect of data augmentation on class-wise F1-score performance across all evaluated architectures

Model	**Meningioma F1**			**Glioma F1**			**Pituitary F1**		
	Aug	**NoAug**	Δ	**Aug**	**NoAug**	Δ	**Aug**	**NoAug**	Δ
Swin-T	0.9118	0.9059	+0.0059	0.9407	0.9369	+0.0038	0.9359	0.9248	+0.0111
ConvNeXt-Tiny	0.8994	0.8876	+0.0118	0.9295	0.9168	+0.0127	0.9416	0.9218	+0.0198
ConvNeXt-Small	0.8882	0.8844	+0.0038	0.9323	0.9157	+0.0166	0.9425	0.9153	+0.0272
DenseNet121	0.8971	0.8514	+0.0457	0.9193	0.9080	+0.0113	0.9545	0.9363	+0.0182
VGG19	0.8883	0.8596	+0.0287	0.9157	0.9201	-0.0044	0.9459	0.9537	-0.0078
EffNetV2-S	0.8657	0.8391	+0.0266	0.9299	0.9048	+0.0251	0.9276	0.9351	-0.0075
EffNet-B3	0.7869	0.7919	-0.0050	0.8654	0.8790	-0.0136	0.9244	0.9263	-0.0019
ResNet50	0.7671	0.7455	+0.0216	0.8455	0.8166	+0.0289	0.9205	0.9067	+0.0138
Mean	**0.8631**	**0.8457**	**+0.0174**	**0.9098**	**0.8997**	**+0.0101**	**0.9366**	**0.9275**	**+0.0091**

Tables 4.5 and 4.6 reveal a consistent but model-dependent benefit from augmentation[34]. Averaged across all eight models, augmentation improves holdout accuracy by +1.16 percentage points (mean Δacc = +0.0116) and macro F1 by +0.0122. The macro AUC effect is negligible (mean ΔAUC = -0.0001), confirming that augmentation improves decision-boundary precision rather than the fundamental ranking ability of the pretrained feature extractors.

DenseNet121[28] and ResNet50[27] benefit most in accuracy (both +0.0229). DenseNet121's meningioma F1 drops to 0.8514 without augmentation, a fall of 0.0457 points, the largest per-class drop in the entire experiment confirming that augmentation-induced diversity is critical for the minority class[39]. **ResNet50 shows the largest glioma improvement (+0.0289 glioma F1), suggesting that without augmentation it develops a stronger bias toward over-predicting the majority class.** Swin-T[14] and VGG19[25] show the smallest gains (+0.0066 and +0.0032 accuracy respectively), because their regularisation mechanisms shifted-window attention and dropout independently constrain overfitting.

EfficientNet-B3[29] is the only model for which augmentation produces a negative effect (Δacc = -0.0099). The geometric augmentations interact adversely with the Squeeze-and-Excitation channel recalibration in early layers during single-GPU fine-tuning, where the SE calibration signal is sensitive to the spatial statistics of each input. Without augmentation, the spatial statistics are stable and SE converges more reliably. Importantly, B3 remains the weakest model in both conditions.

At the class level, augmentation produced its greatest benefit for meningioma classification, with a mean increase in F1-score of +0.0174, followed by glioma (+0.0101) and pituitary tumours (+0.0091). This finding supports the inclusion of random rotations ($\pm 15°$) and affine translations ($\pm 5\%$), which were designed to simulate variations in patient head positioning during image acquisition. Such transformations introduce realistic anatomical

variability and appear particularly beneficial for the minority meningioma class, where the smaller number of training examples increases susceptibility to overfitting and limits the diversity of observed imaging patterns[4].

These results confirm the augmentation pipeline as a methodologically justified and empirically validated choice for this dataset.

4.6 Discussion and Future Work

4.6.1 Discussion

4.6.1.1 Interpretation of Results

The experimental results demonstrate a distinct architectural hierarchy on this patient-level benchmark[16]. Swin-T [14], with 28M parameters and a hierarchical shifted-window attention strategy, surpasses all seven convolutional models on every macro metric[37]. The two ConvNeXt[31] variants follow closely, indicating that Transformer-inspired inductive biases large-kernel depthwise convolution, layer normalisation, GELU provide a systematic advantage over simply conventional CNN design patterns. DenseNet121's[28] competitive performance despite being the smallest model (8.1M parameters) demonstrates that dense feature reuse is a particularly successful method for small-scale medical imaging datasets where overfitting risk is significant[6].

The most noteworthy underperformance is EfficientNet-B3[29], which scores about 6 percentage points below EfficientNetV2-S[30] despite sharing the compound scaling concept. Three factors are likely responsible: the single-GPU constraint imposed by SE block incompatibility with DataParallel reduces effective batch diversity[36]; B3's standard 300×300 training resolution is lower than V2-S's native 384×384, reducing spatial detail available for feature extraction; and B3's depthwise-heavy early blocks are known to be less training-stable than V2's Fused-MBConv early stages when fine-tuning from ImageNet[35] weights on small datasets[6].

4.6.1.2 Class-Level Analysis

Meningioma is the hardest class for every model in our study, consistent with findings across the brain tumor classification literature. The radiological reason for this problem is well-established[4]: meningioma's key differentiating characteristic is its extra-axial origin and attachment to the meningeal surface, global link that is not effectively captured by local convolutional kernels with small receptive fields[6]. Swin-T's advantage on meningioma (F1 0.9118 vs. 0.8882–0.8994 for ConvNeXt models) is consistent with this perspective, since its shifted-window attention mechanism allows the model to integrate spatial context throughout

the full image at every stage.

Pituitary adenoma is the easiest class for most models. This reflects its morphological consistency: pituitary tumors occupy a well-defined anatomical region (the sella turcica), are typically round and homogeneous in T1-weighted contrast-enhanced imaging, and demonstrate a predictable enhancement pattern. The high inter-model agreement on pituitary (F1 range 0.9205–0.9545) demonstrates that this class's visual characteristic is discriminative enough for even traditional CNNs to classify consistently[25], [27].

Glioma occupies an intermediate place in difficulty. As the dominant class (46.5 % of slices), it receives the most training instances, which benefits all models. However, gliomas are histologically diverse[7] and may present with irregular boundaries, satellite lesions, and varying enhancement patterns, which creates diagnostic difficulty for high-grade cases.

4.6.1.3 Comparison to Prior Works

Direct comparison with prior studies is limited by methodological differences in dataset partitioning. Most published works employed image-level splitting[10], [11], [12], [15], which permits information leakage between training and test sets and consequently inflates reported performance. Therefore, their results cannot be considered directly comparable to those obtained under the strict patient-level protocol adopted in this study. The only comparable benchmark is Khan et al[13]., who reported 90.67% accuracy using patient-level splitting. Six of the eight evaluated models exceeded this result, suggesting that recent deep learning architectures are capable of delivering superior performance even under rigorous leakage-free evaluation conditions.

The observed 2–6 percentage point performance difference between this study and image-level evaluations likely reflects the inflation of classification performance caused by image-level data splitting, highlighting the importance of rigorous patient-level evaluation.

4.6.2 Future Work

The present study focused on three-class brain tumor classification using only T1-weighted contrast-enhanced MRI images. Several extensions can be explored to increase the clinical applicability of the proposed framework.

1. **Multi-modal MRI Classification**

   This work utilized only T1-weighted contrast-enhanced (T1-CE) images. However, clinical diagnosis typically relies on multiple MRI modalities, including T1-CE, T2-weighted, and FLAIR sequences, as each modality provides complementary information about tumor structure and surrounding tissue. Future work will investigate multi-modal deep learning models capable of jointly processing multiple MRI sequences to improve classification performance and robustness.

2. **Tumor Grade Prediction**

   The current framework classifies tumor type only (meningioma, glioma, and pituitary tumor) without considering tumor severity. An important extension is the prediction of tumor grade, particularly for gliomas, where grading is strongly associated with prognosis and treatment planning. Future research will explore multi-class or multi-task learning approaches that simultaneously identify the tumor type and estimate its pathological grade.

3. **Inclusion of Healthy Subjects**

   The dataset used in this study contains only tumor cases. Consequently, the model assumes that every input MRI belongs to one of the three tumor categories. Future work will extend the classification task by introducing a fourth class representing healthy brain MRI scans. This enhancement would allow the system to first determine whether a tumor is present and then classify its type, making it more suitable for real-world clinical screening scenarios.

4.7 Conclusion

This chapter has presented experimental results of the comparative study. Across all eight models evaluated on the same holdout set, Swin Transformer Tiny achieves the highest overall accuracy (93.13 %), the best macro F1-score (0.9295), and the highest meningioma F1-score (0.9118) the most clinically significant metric given that meningioma is always the hardest class to classify correctly. The ConvNeXt family occupies the second tier, illustrating that Transformer-inspired convolutional architectures provide a distinct advantage over traditional CNN methodologies. DenseNet121 stands out within the conventional CNN family, matching the ConvNeXt models in total F1 despite utilizing the fewest parameters. EfficientNet-B3 and ResNet50 exhibit the poorest performance, with B3 demonstrating a significant deficit of around 6 percentage points relative to the subsequent model – a result that necessitates additional examination. The continuous pattern of meningioma being the hardest class across all models, and pituitary being the easiest, reflects both the radiological complexity of meningioma and the morphological distinctiveness of pituitary tumors in T1-weighted contrast-enhanced MRI. These findings are evaluated and contextualised in the General Conclusion and Discussion that follow.

General Conclusion

This thesis presented a comprehensive and methodologically rigorous comparative study of eight deep learning architectures for brain tumor classification on the Cheng MRI dataset. The central contribution of this work is the adoption of a strict patient-level data splitting strategy, an 80/20 stratified holdout combined with 5-fold cross-validation on the training pool which eliminates the inter-slice data leakage that is present in the vast majority of previous research on this dataset and which inflates their reported performance metrics. The key findings can be summarised as follows. Among the eight architectures examined VGG19, ResNet50, DenseNet121, EfficientNet-B3, EfficientNetV2-S, ConvNeXt-Tiny, ConvNeXt-Small, and Swin Transformer, Swin-T achieves the best overall holdout accuracy of 93.13 % and the highest meningioma F1-score of 0.9118. The ConvNeXt family followed closely, proving that the integration of Vision Transformer design concepts into convolutional architectures delivers a measurable advantage. DenseNet121 displayed outstanding efficiency, reaching competitive performance (F1 0.9237) with only 8.1M parameters. EfficientNet-B3 was the weakest model (accuracy 85.92 %), underperforming by a margin that cannot be attributed to training conditions alone and that suggests the SE block optimisation strategy used in this architecture may be less suited to transfer learning on small-scale medical datasets when multi-GPU training is unavailable. ResNet50, while strong on glioma and pituitary, consistently underperformed on meningioma (F1 0.7671) the hardest class across all models likely because its local convolutional receptive field cannot efficiently capture the extra-axial spatial relationship that is the primary radiological distinguishing feature of meningioma. The corrected benchmark created by this study is considerably lower than many results reported in past image-level studies: Sultan et al. reported 96.13 % on this dataset using image-level splitting, while this work's best result is 93.13 % under patient-level settings. This 3-point gap represents the expense of scientific rigour and provides a more honest estimate of generalization to actually unseen patients which is the important clinical circumstance.

On the advantage of Swin-T for meningioma: The hierarchical shifted-window attention method of Swin-T computes self-attention within local windows while propagating global context across windows during the patch merging stages. This dual local-global reasoning capacity appears to be particularly suited to meningioma classification, where the diagnostic signal lies not only in the texture of the tumor mass itself but in its spatial relationship to the meninges and surrounding structures a global feature that purely local convolutional operations cannot efficiently encode without very deep architectures. On the ConvNeXt vs. pure CNN gap: The ConvNeXt family's advantage over VGG19 and ResNet50 confirms that the specific Transformer-inspired design choices: large 7×7 depthwise kernels, inverted

bottleneck structure, GELU activation, and layer normalisation provide meaningful inductive bias improvements for this task. This is practically significant since ConvNeXt models can be trained on regular CNN training pipelines without the memory and compatibility constraints of Transformers. On EfficientNet-B3's underperformance: The 6-point accuracy gap between EfficientNet-B3 (85.92 %) and EfficientNetV2-S (91.16 %) is remarkable. Both architectures share the compound scaling philosophy and SE blocks. The explanation is threefold: V2 replaces early depthwise convolutions with Fused-MBConv operations that are more training-stable during fine-tuning; V2 was pretrained on a larger and more diverse ImageNet-21k subset in some configurations; and B3's single-GPU constraint (due to SE block incompatibility with DataParallel) combined with its standard resolution (300×300) may have prevented it from fully exploiting the available batch diversity. On meningioma as the hardest class: All eight models score lowest on meningioma. This is compatible with the clinical literature: meningioma has the highest visual similarity to normal brain structures and other extra-axial disorders, its contrast enhancement pattern is varied, and it creates the least distinctive local texture signal. Its detection from T1-weighted MRI alone is fundamentally more ambiguous than glioma (usually intra-axial, infiltrative) or pituitary adenoma (limited to the sella turcica, extremely consistent appearance). Comparison to past work: This study's 93.13 % holdout accuracy for Swin-T is attained under patient-level conditions and is therefore not directly comparable to image-level studies. The one earlier patient-level study identified in the literature analysis Khan et al. at 90.67 % is exceeded by all models in this study except ResNet50 and EfficientNet-B3. This shows that current architectures, when trained under stringent conditions, reliably improve on prior patient-level benchmarks.

Limitations

This work has some limitations that should be noted:

- The analysis is constrained to a single dataset from two hospitals in China with a specific MRI scanner methodology. Generalisation to datasets from other scanners, field strengths, or demographics remains unvalidated.
- Only T1-weighted contrast-enhanced images are employed. Clinical practice often employs multi-sequence MRI (T2, FLAIR, DWI), and multi-modal models would likely yield superior performance.
- The classification task is three-class and does not distinguish between tumor grades within each class, particularly for glioma.
- The Kaggle session limit of 12 hours per GPU session put realistic constraints on hyperparameter search depth; a more extensive search would offer modestly improved results.

References

[1] World Health Organization, *Cancer*, https://www.who.int/news-room/fact-sheets/detail/cancer, Accessed: 2025, 2022.

[2] D. Hanahan and R. A. Weinberg, ``Hallmarks of cancer: The next generation,'' *Cell*, vol. 144, no. 5, pp. 646–674, 2011. DOI: 10.1016/j.cell.2011.02.013

[3] Q. T. Ostrom, G. Cioffi, K. Waite, C. Kruchko, and J. S. Barnholtz-Sloan, ``CBTRUS statistical report: Primary brain and other central nervous system tumors diagnosed in the united states in 2014--2018,'' *Neuro-Oncology*, vol. 23, no. Suppl 3, pp. iii1–iii105, 2021. DOI: 10.1093/neuonc/noab200

[4] D. N. Louis et al., ``The 2021 WHO classification of tumors of the central nervous system: A summary,'' *Neuro-Oncology*, vol. 23, no. 8, pp. 1231–1251, 2021. DOI: 10.1093/neuonc/noab106

[5] V. T. DeVita Jr., T. S. Lawrence, and S. A. Rosenberg, *Cancer: Principles and Practice of Oncology*, 11th. Wolters Kluwer, 2019.

[6] G. Litjens et al., ``A survey on deep learning in medical image analysis,'' *Medical Image Analysis*, vol. 42, pp. 60–88, 2017. DOI: 10.1016/j.media.2017.07.005

[7] M. Weller et al., ``EANO guidelines on the diagnosis and treatment of diffuse gliomas of adulthood,'' *Nature Reviews Clinical Oncology*, vol. 18, no. 3, pp. 170–186, 2021. DOI: 10.1038/s41571-020-00447-z

[8] J. Cheng et al., ``Enhanced performance of brain tumor classification via tumor region augmentation and partition,'' *PLOS ONE*, vol. 10, no. 10, e0140381, 2015. DOI: 10.1371/journal.pone.0140381

[9] J. Cheng et al., ``Retrieval of brain tumors by adaptive spatial pooling and fisher vector representation,'' *PLOS ONE*, vol. 11, no. 6, e0157112, 2016. DOI: 10.1371/journal.pone.0157112

[10] N. Abiwinanda, M. Hanif, S. T. Hesaputra, A. Handayani, and T. R. Mengko, ``Brain tumor classification using convolutional neural network,'' in *World Congress on Medical Physics and Biomedical Engineering*, Springer, 2019, pp. 183–189. DOI: 10.1007/978-981-10-9035-6_33

[11] A. Pashaei, H. Sajedi, and N. Jazayeri, ``Brain tumor classification via convolutional neural network and extreme learning machines,'' in *8th International Conference on Computer and Knowledge Engineering (ICCKE)*, IEEE, 2018, pp. 314–319. DOI: 10.1109/ICCKE.2018.8566571

[12] H. H. Sultan, N. M. Salem, and W. Al-Atabany, ``Multi-classification of brain tumor images using deep neural network," *IEEE Access*, vol. 7, pp. 69 215–69 225, 2019. DOI: 10.1109/ACCESS.2019.2919122

[13] M. A. Khan, T. Akram, M. Sharif, M. Yasir, M. Yasmin, and Y.-D. Zhang, ``Construction of saliency map and hybrid set of features for efficient segmentation and classification of brain tumor," *Microscopy Research and Technique*, vol. 83, no. 10, pp. 1248–1266, 2020. DOI: 10.1002/jemt.23519

[14] Z. Liu et al., ``Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 10 012–10 022. DOI: 10.1109/ICCV48922.2021.00986

[15] S. Deepak and P. M. Ameer, ``Brain tumor classification using deep CNN features via transfer learning," *Computers in Biology and Medicine*, vol. 111, p. 103 345, 2019. DOI: 10.1016/j.compbiomed.2019.103345

[16] M. Roberts et al., ``Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans," *Nature Machine Intelligence*, vol. 3, pp. 199–217, 2021. DOI: 10.1038/s42256-021-00307-0

[17] R. Kohavi, ``A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2, 1995, pp. 1137–1145.

[18] B. H. Menze et al., ``The multimodal brain tumor image segmentation benchmark (BRATS)," *IEEE Transactions on Medical Imaging*, vol. 34, no. 10, pp. 1993–2024, 2015. DOI: 10.1109/TMI.2014.2377694

[19] S. Bakas et al., ``Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features," *Scientific Data*, vol. 4, p. 170 117, 2017. DOI: 10.1038/sdata.2017.117

[20] K. Clark et al., ``The cancer imaging archive (TCIA): Maintaining and operating a public information repository," *Journal of Digital Imaging*, vol. 26, no. 6, pp. 1045–1057, 2013. DOI: 10.1007/s10278-013-9622-7

[21] M. Nickparvar, *Brain tumor MRI dataset*, Kaggle, Accessed: 2025, 2021. [Online]. Available: https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, ``Gradient-based learning applied to document recognition,'' *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ``ImageNet classification with deep convolutional neural networks,'' in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012, pp. 1097–1105.

[25] K. Simonyan and A. Zisserman, ``Very deep convolutional networks for large-scale image recognition,'' in *International Conference on Learning Representations (ICLR)*, 2015. arXiv: 1409.1556.

[26] S. J. Pan and Q. Yang, ``A survey on transfer learning,'' *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191

[27] K. He, X. Zhang, S. Ren, and J. Sun, ``Deep residual learning for image recognition,'' in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90

[28] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, ``Densely connected convolutional networks,'' in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708. DOI: 10.1109/CVPR.2017.243

[29] M. Tan and Q. V. Le, ``EfficientNet: Rethinking model scaling for convolutional neural networks,'' in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, PMLR, 2019, pp. 6105–6114.

[30] M. Tan and Q. V. Le, ``EfficientNetV2: Smaller models and faster training,'' in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, PMLR, 2021, pp. 10 096–10 106.

[31] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, ``A ConvNet for the 2020s,'' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 11 976–11 986. DOI: 10.1109/CVPR52688.2022.01167

[32] A. Dosovitskiy et al., ``An image is worth 16x16 words: Transformers for image recognition at scale,'' in *International Conference on Learning Representations (ICLR)*, 2021. arXiv: 2010.11929.

[33] F. Pedregosa et al., ``Scikit-learn: Machine learning in Python,'' *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] C. Shorten and T. M. Khoshgoftaar, ``A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019. DOI: 10.1186/s40537-019-0197-0

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ``ImageNet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848

[36] A. Paszke et al., ``PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019, pp. 8024–8035.

[37] M. Sokolova and G. Lapalme, ``A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009. DOI: 10.1016/j.ipm.2009.03.002

[38] T. Fawcett, ``An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006. DOI: 10.1016/j.patrec.2005.10.010

[39] H. He and E. A. Garcia, ``Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009. DOI: 10.1109/TKDE.2008.239

[40] I. Loshchilov and F. Hutter, ``SGDR: Stochastic gradient descent with warm restarts," in *International Conference on Learning Representations (ICLR)*, 2017. arXiv: 1608.03983.

[41] L. Prechelt, ``Early stopping --- but when?" In *Neural Networks: Tricks of the Trade*, Springer, 1998, pp. 55–69. DOI: 10.1007/3-540-49430-8_3

[42] D. P. Kingma and J. Ba, ``Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015. arXiv: 1412.6980.

[43] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, ``On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, PMLR, 2013, pp. 1139–1147.

[44] P. Goyal et al., ``Accurate, large minibatch SGD: Training ImageNet in 1 hour," *arXiv preprint*, 2017. arXiv: 1706.02677.

[45] R. Pascanu, T. Mikolov, and Y. Bengio, ``On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, PMLR, 2013, pp. 1310–1318.

[46] C. R. Harris et al., ``Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020. DOI: 10.1038/s41586-020-2649-2

[47] W. McKinney, ``Data structures for statistical computing in python,'' in *Proceedings of the 9th Python in Science Conference (SciPy)*, 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a

[48] J. D. Hunter, ``Matplotlib: A 2d graphics environment,'' *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55

[49] M. L. Waskom, ``Seaborn: Statistical data visualization,'' *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. DOI: 10.21105/joss.03021

[50] J. Hu, L. Shen, and G. Sun, ``Squeeze-and-excitation networks,'' in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132–7141. DOI: 10.1109/CVPR.2018.00745

[51] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, ``Mixup: Beyond empirical risk minimization,'' in *International Conference on Learning Representations (ICLR)*, 2018. arXiv: 1710.09412.

[52] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, ``Rethinking the inception architecture for computer vision,'' in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308

Appendix A: Mathematical Formulations

This appendix presents the mathematical formulations used throughout the proposed methodology. These equations provide additional details regarding image preprocessing, class imbalance handling, optimization procedures, and evaluation metrics.

A.1 Image Preprocessing

A.1.1 Pixel Intensity Normalization

Raw MRI pixel intensities are normalized into the range $[0, 255]$ using min--max normalization:

$$I_{\text{norm}} = \left\lfloor \frac{255}{I_{\max} - I_{\min}} \left(I - I_{\min}\right) \right\rfloor \tag{A.1}$$

where:

- I : original pixel intensity
- $I_{\max}$: maximum image intensity
- $I_{\min}$: minimum image intensity
- I_{norm} : normalized intensity

A.2 Class Imbalance Handling

To compensate for class imbalance, inverse-frequency class weights are computed as:

$$w_c = \frac{\frac{1}{n_c}}{\sum_{j=1}^{C} \frac{1}{n_j}} \times C \tag{A.2}$$

where:

- w_c : class weight for class c
- n_c : number of samples in class c
- C : total number of classes

A.3 Loss Function

Weighted cross-entropy loss is defined as:

$$L = -\sum_{c=1}^{C} w_c y_c \log(\hat{y}_c) \tag{A.3}$$

where:

- w_c : class weight
- y_c : ground-truth class label
- $\hat{y}_c$: predicted probability

A.4 Softmax Classification

The output probabilities are generated using the Softmax function:

$$P(y = c|x) = \frac{e^{z_c}}{\sum_{j=1}^{C} e^{z_j}} \tag{A.4}$$

where:

- z_c : output logit of class c
- C : total number of classes

A.5 Cosine Annealing Learning Rate

The cosine annealing learning schedule used during fine-tuning is defined as:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left[1 + \cos\left(\frac{t\pi}{T}\right)\right] \tag{A.5}$$

where:

- η_t : learning rate at iteration t
- $\eta_{\max}$: initial learning rate
- $\eta_{\min}$: minimum learning rate
- T : total number of iterations

Appendix B: List of Abbreviations

ACC Accuracy

ADC Apparent Diffusion Coefficient

AI Artificial Intelligence

AMP Automatic Mixed Precision

AUC Area Under the Curve

CAD Computer-Aided Diagnosis

CNN Convolutional Neural Network

CNS Central Nervous System

ConvNeXt Convolutional Next architecture family

CT Computed Tomography

CUDA Compute Unified Device Architecture

DenseNet Densely Connected Convolutional Network

DWI Diffusion-Weighted Imaging

EfficientNet Efficient Neural Network

EfficientNetV2 EfficientNet Version 2

FC Fully Connected Layer

FLAIR Fluid-Attenuated Inversion Recovery

FN False Negative

FP False Positive

FPR False Positive Rate

GELU Gaussian Error Linear Unit

GPU Graphics Processing Unit

Grad-CAM Gradient-weighted Class Activation Mapping

ImageNet Large-scale visual recognition dataset

IoU Intersection over Union

LR Learning Rate

MRI Magnetic Resonance Imaging

OVR One-Versus-Rest

PET Positron Emission Tomography

PET-CT Positron Emission Tomography–Computed Tomography

PID Patient Identifier

PSF Point Spread Function

ReLU Rectified Linear Unit

ResNet Residual Network

RGB Red Green Blue

ROC Receiver Operating Characteristic

SE Squeeze-and-Excitation

SGD Stochastic Gradient Descent

Swin-T Swin Transformer Tiny

T1 T1-weighted MRI sequence

T2 T2-weighted MRI sequence

TN True Negative

TP True Positive

TPR True Positive Rate

VGG Visual Geometry Group

VRAM Video Random Access Memory

WHO World Health Organization

B.1 Cross-Validation Fold Results

The tables below report per-fold accuracy, macro F1, and macro AUC for each of the eight architectures over the 5-fold patient-level cross-validation. These results confirm that the holdout evaluation is not a lucky single run and that CV performance is consistent across folds.

Table B.1: Per-fold cross-validation results for all evaluated architectures.

Model	Accuracy					Macro F1					Mean AUC				
	F1	F2	F3	F4	F5	F1	F2	F3	F4	F5	F1	F2	F3	F4	F5
ResNet50	0.8592	0.8727	0.8236	0.8767	0.8809	0.8146	0.8614	0.8072	0.8500	0.8686	0.9336	0.9612	0.9415	0.9619	0.9695
DenseNet121	0.9441	0.9478	0.9539	0.9635	0.9368	0.9203	0.9408	0.9455	0.9512	0.9260	0.9715	0.9876	0.9881	0.9930	0.9908
VGG19	0.9420	0.9311	0.9399	0.9475	0.9332	0.9184	0.9220	0.9307	0.9365	0.9260	0.9469	0.9913	0.9869	0.9929	0.9908
EffNet-B3	0.9089	0.9395	0.8778	0.9338	0.9007	0.8729	0.9305	0.8638	0.9213	0.8913	0.9344	0.9912	0.9578	0.9864	0.9834
EffNetV2-S	0.9420	0.9499	0.9519	0.9703	0.9314	0.9097	0.9433	0.9430	0.9580	0.9238	0.9534	0.9938	0.9867	0.9959	0.9917
ConvNeXt-Tiny	0.9321	0.9698	0.9793	0.9691	0.9601	0.9198	0.9735	0.9775	0.9605	0.9488	0.9435	0.9732	0.9989	0.9666	0.9457
ConvNeXt-Small	0.9554	0.9683	0.9822	0.9849	0.9830	0.9544	0.9670	0.9812	0.9840	0.9819	0.9584	0.9633	0.9982	0.9964	0.9895
Swin-T	0.9755	0.9704	0.9634	0.9674	0.9615	0.9736	0.901	0.9620	0.9652	0.9602	0.9981	0.9823	0.9756	0.9812	0.9794