Brain Tumor Classification Using Convolutional Neural Networks

Lathitha Nongauza
Johannesburg, South Africa
Lathithanongauza17@gmail.com

Abstract

Brain tumors are serious medical conditions where early and accurate diagnosis significantly improves patient survival rates. Magnetic Resonance Imaging (MRI) plays a crucial role in tumor detection, but manual interpretation can be challenging even for medical experts. This study explores the application of deep learning, specifically custom Convolutional Neural Networks (CNNs), for automated classification of brain tumors into three categories: glioma, meningioma, and pituitary tumors. We present a complete pipeline from data preprocessing and augmentation to model design, training, and evaluation. Using 5-fold cross-validation and various optimization techniques including early stopping and adaptive optimizers, our custom CNN model achieved 85.10% test accuracy while effectively mitigating overfitting. The results demonstrate the potential of lightweight, custom CNN architectures for medical image classification tasks.

1 Introduction

Brain tumors represent a critical medical challenge where timely and accurate diagnosis is paramount for effective treatment and improved patient outcomes. MRI scans serve as the primary diagnostic tool for detecting and identifying brain tumors, but interpreting these complex medical images requires considerable expertise and can be subject to human error and variability. Recent advancements in deep learning, particularly Convolutional Neural Networks (CNNs), have shown remarkable success in various medical image analysis tasks. Unlike traditional machine learning approaches that require handcrafted features, CNNs can automatically learn hierarchical representations directly from raw image data, making them particularly suitable for medical image classification. In this study, we investigate the application of CNNs for brain tumor classification using MRI scans. Rather than employing pre-trained models, we designed and trained a custom CNN architecture specifically tailored to our dataset and classification task. This approach allows for greater control over model complexity, computational efficiency, and interpretability. Our contributions

include: a complete pipeline for brain tumor MRI classification using custom CNNs; implementation of data augmentation techniques to enhance model generalization; application of 5-fold cross-validation for robust performance estimation; development of optimization strategies to mitigate overfitting; and achievement of competitive classification accuracy with a lightweight architecture.

2 Dataset and Preprocessing

2.1 Dataset Description

We utilized the Brain Cancer MRI dataset available on Kaggle, which contains color MRI scans of brain tumors across three categories: glioma, meningioma, and pituitary tumors. The dataset presents several challenges typical of medical imaging data, including varying image resolutions, different scanning protocols, and class imbalance.

2.2 Preprocessing Pipeline

To ensure consistency and facilitate model training, we implemented a comprehensive preprocessing pipeline. All images were resized from their original dimensions to a uniform size of 128×128 pixels. This standardization reduces computational complexity while preserving essential diagnostic features. Pixel values were normalized to the range $[0, 1]$ using min-max scaling, followed by standardization to have zero mean and unit variance. Images were converted to PyTorch tensors for efficient GPU processing, with all images converted to RGB format with three channels to maintain consistency with the CNN architecture.

3 Data Augmentation

To address the limited size of medical imaging datasets and improve model generalization, we implemented several data augmentation techniques. Each training image was randomly transformed during training using random vertical flipping with probability 0.5 to increase orientation invariance, and random rotation where images were rotated by $90°$, $180°$, or $270°$ to enhance rotational invariance. These augmentation strategies are supported by previous studies in medical image analysis. For instance, [2] successfully applied similar augmentations to brain tumor MRI images, tripling their effective dataset size and improving model robustness. The augmented images were combined with the original dataset, effectively expanding our training data and reducing the risk of overfitting while encouraging the model to learn invariant features.

4 Dataset Splitting and Validation Strategy

To ensure reliable performance estimation and prevent overfitting, we employed 5-fold cross-validation. The complete dataset was partitioned into five mutually exclusive subsets of approximately equal size. During each training iteration, four folds were used for training while one fold was reserved for validation. This process was repeated five times, with each fold serving as the validation set exactly once. This strategy provides several advantages: it maximizes data utilization for both training and validation, reduces variance in performance estimation, provides insight into model stability across different data partitions, and helps identify potential overfitting to specific data splits. After cross-validation, we reserved a completely independent test set (20% of the total data) for final model evaluation, ensuring that our reported accuracy reflects true generalization performance.

5 Model Architecture

We designed a custom CNN architecture with careful consideration of the trade-off between model complexity and generalization ability. The architecture is deliberately kept simple to prevent overfitting while maintaining sufficient capacity to learn discriminative features.

5.1 Architecture Details

The architecture consists of an input layer accepting $128 \times 128 \times 3$ RGB images, followed by two convolutional blocks. The first convolutional block includes a convolutional layer with 8 filters, 3×3 kernel, padding=1, ReLU activation function, and max pooling with 2×2 window and stride 2. The second convolutional block contains a convolutional layer with 16 filters, 3×3 kernel, padding=1, ReLU activation function, and max pooling with 2×2 window and stride 2. This is followed by fully connected layers: a flatten layer to convert spatial features to vector representation, a dense layer with 64 units and ReLU activation, a dropout layer with rate 0.5 for regularization, and an output layer with 3 units (one per class) with softmax activation.

5.2 Design Rationale

The architecture was designed with several considerations. Depth was limited to two convolutional blocks to prevent overfitting on our moderately sized dataset. Filter sizes of 3×3 kernels were chosen as they provide a good balance between receptive field and parameter efficiency. Same padding was used to preserve spatial dimensions. Max pooling with stride 2 reduces spatial dimensions while retaining the most salient features. Dropout was applied to the fully connected layer to prevent co-adaptation of features.

6 Training Strategy and Optimization

6.1 Initial Challenges

During initial experiments, we observed signs of overfitting characterized by rapid decrease in training loss, plateauing validation loss at higher values, and large gap between training and validation performance.

6.2 Optimization Techniques

To address these challenges, we implemented several strategies. Early stopping was employed where training was terminated when validation loss failed to improve for three consecutive epochs, preventing overfitting and conserving computational resources. We experimented with various learning rates across different folds, with values ranging from 0.0001 to 0.005, allowing each fold to find its optimal learning dynamics. We compared Stochastic Gradient Descent (SGD) with Adam optimizer and found that Adam provided faster convergence and better final performance due to its adaptive learning rate mechanism. A batch size of 32 was selected as it provided a good balance between convergence stability and computational efficiency.

6.3 Parallel Training Implementation

To accelerate experimentation, we implemented parallel training across the five folds using Python's multiprocessing module. Each fold was trained independently with its own hyperparameters, and results were aggregated for comprehensive analysis.

7 Results

7.1 Training Performance

Table 1 shows the training and validation losses across 10 epochs for one of the folds:

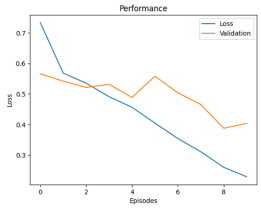
Table 1: Training and Validation Losses

Training Loss	Validation Loss
0.7331	0.5657
0.5679	0.5419
0.5352	0.5214
0.4909	0.5313
0.4565	0.4889
0.4047	0.5578
0.3542	0.5039
0.3111	0.4657
0.2602	0.3877
0.2292	0.4039

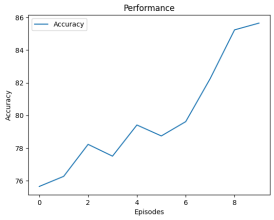
The training curves (Figure 1a) show consistent convergence across all folds, with validation loss closely tracking training loss after the initial epochs, indicating effective regularization.

7.2 Final Test Performance

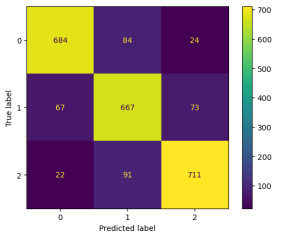
The model achieved the following performance on the held-out test set: test accuracy of 85.10%.



(a) Training and validation loss



(b) Validation accuracy



(c) Confusion matrix

Figure 1: Training progress metrics. (a) Loss curves showing convergence with minimal overfitting. (b) Accuracy improvement during training. (c) Classification across classes

These results demonstrate that our custom CNN architecture can effectively classify brain tumors with high accuracy while maintaining good generalization to unseen data.

8 Discussion

Our results demonstrate that relatively simple CNN architectures can achieve competitive performance on medical image classification tasks. The 85.10% test

accuracy, achieved without using pre-trained models or transfer learning, suggests that task-specific architectures can be effective when properly regularized and trained with appropriate data augmentation. Several factors contributed to our model's success: appropriate data augmentation using vertical flipping and rotation improved model robustness without introducing unrealistic variations; effective regularization through dropout and early stopping successfully mitigated overfitting; comprehensive validation via 5-fold cross-validation provided reliable performance estimates and guided hyperparameter selection; and optimizer selection using Adam optimizer's adaptive learning rates facilitated faster and more stable convergence.

9 Conclusion

This work presents a complete pipeline for brain tumor classification using custom CNN architectures. Through careful preprocessing, data augmentation, cross-validation, and optimization, we developed a model that achieves 85.10% accuracy in distinguishing between glioma, meningioma, and pituitary tumors from MRI scans. The findings highlight the potential of lightweight, custom-designed neural networks for medical image analysis, offering a computationally efficient alternative to large pre-trained models while maintaining competitive performance.

References

[1] Abdou, M. A. (2022). *Literature review: Efficient deep neural networks techniques for medical image analysis*. Neural Computing and Applications, 34(8), 5791-5812.

[2] Badža, M. M. & Barjaktarović, M. Č. (2020). *Classification of brain tumors from MRI images using a convolutional neural network*. Applied Sciences, 10(6), 1999.