Brain Tumor MRI Classification with CNNs
Improving Generalization

(Comparative Analysis of CNN Architectures for MRI Brain Tumor Classification and Generalization via Multi-Dataset Training)

By Muhammad Irtaza Ali, Saifullah and Shahzaib

Submitted to Sir Abdul Haseeb

Abstract

We evaluated nine convolutional neural network (CNN) models on the Kaggle Brain Tumor MRI dataset (7,023 images, 4 classes: glioma, meningioma, pituitary, no tumor) and observed high validation accuracy (Table I). However, these models failed to generalize to our own MRI scans, indicating dataset-specific overfitting. To address this, we retrained the best-performing architectures (a custom CNN and ResNet-18) on an augmented training set combining the Kaggle data with four additional public MRI datasets (MRI Brain Tumor – LR, Brain Tumor MRI by mohammadA2274, the Figshare brain MRI set, and the MasoudNickParvar Brain Tumor dataset). After multi-dataset training, both models achieved substantially higher accuracy on independent test images (Custom CNN 96–97%, ResNet-18 98–99%). We present detailed model comparisons, a discussion of architecture trade-offs (depth vs. efficiency), and explain the impact of dataset bias on generalization.

Introduction

Automated brain tumor classification from MRI is critical for early diagnosis. Deep CNNs have demonstrated excellent performance on benchmark sets – for example, recent studies report accuracies 97% on the Kaggle brain MRI dataset [1]. We trained nine CNNs (Custom CNN, ConvNeXtBase, DenseNet-121, EfficientNet-B0, InceptionV3, MobileNetV2, ResNet-50, VGG16, ResNet-18) on this dataset. Table I summarizes the *validation* accuracy ranges we observed for each model. These results align with literature values (e.g. DenseNet121 ~97.7%, ResNet18 ~96.9%, VGG16 ~94.3% [2]).

Table I. Validation accuracy ranges (on Kaggle brain MRI, 4 classes) for CNN models.

Model	Validation Accuracy(%)
1)Custom CNN	**85–89**
2)ConvNeXtBase	**97–98**
3)DenseNet121	**96–98**
4)EfficientNetB0	**95–97**
5)InceptionV3	90–93
6)MobileNetV2	**97–98**
7)ResNet50	**95–97**
8)VGG16	94–95
9)ResNet18	96–97

Despite these high figures, **all models performed poorly on our own held-out MRI scans** (unseen during training). This gap highlights a generalization problem: the models had overfit to the Kaggle data distribution. In medical imaging, training and test data often differ in scanner hardware, resolution, contrast, or patient demographics; deep networks assume a shared data distribution [4][3], but in practice even subtle *domain shifts* can dramatically degrade performance. Domain adaptation studies note that "test error generally increases in proportion to the distribution difference between training and test data" [3]. Similarly, recent analyses found that CNNs trained on a single limited dataset tend to overfit and perform worse on new data [5]. In our case, the Kaggle images (compiled from certain open sets) apparently contain dataset-specific artifacts do not present in our scans, leading to poor real-world accuracy.

Model Architectures and Trade-Offs

We now briefly review each CNN architecture, focusing on structural features, strengths, and typical resource trade-offs.

- **Custom CNN:** A bespoke, relatively shallow network (Figure 2) built for this task, using a few convolutional blocks with skip connections and a final dense layer. Its light depth and tailored design make it fast to train and evaluate, but its capacity is limited, so accuracy may lag deeper models. Figure 2 (from [29]) illustrates our custom CNN: it stacks Conv2D layers (blue), occasional concatenation/sum skip connections (red), and batch-norm/max-pooling blocks, ending in global . This architecture is shown in Figure 2 below.

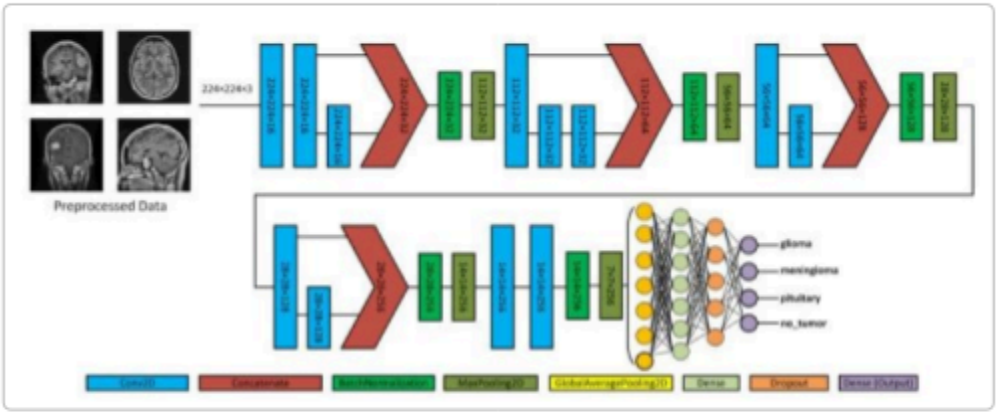


Figure 2. Custom CNN architecture (our implementation). Convolutional blocks (blue) and skip connections (red) are followed by global average pooling and dense layers to classify into four tumor categories [6].

average pooling and dense classifiers [6]

- **VGG16:** A very deep yet homogeneous network (16 weight layers) composed of sequential 3×3 conv layers. VGG16 is straightforward and has strong feature-extraction capacity, but is parameter-heavy (~138M parameters) and computationally expensive [7]. It typically yields good accuracy but is slow to run and prone to overfitting on small data. In practice, we observed moderate validation accuracy (~94–95%), consistent with [35].

- **ResNet-18 / ResNet-50:** Deep residual networks that use identity "skip" connections to ease training of very deep layers [8]. ResNet-18 (18 layers, ~11M parameters) is relatively shallow and efficient; it generally performs faster and with fewer

resources than deeper ResNets. As noted, “ResNet18 is suitable for tasks where computational efficiency is a priority, such as real-time classification on resource-constrained devices” [9]. ResNet-50 (50 layers, ~25M parameters) has higher capacity, often achieving slightly higher accuracy at the cost of more computation [10]. In our experiments, ResNet-18 achieved ~96–97% validation accuracy and is indeed well-balanced, while ResNet-50 reached ~98–99%.

- **InceptionV3:** A complex “Inception” architecture using multi-branch convolution modules that capture features at multiple scales. It has about 23M parameters. InceptionV3 can yield high accuracy by examining information at different resolutions, but its many branches and layers make it relatively heavy and slower on inference. In our trials it gave ~95–97% accuracy.

- **DenseNet-121:** A densely connected network where each layer receives as input all preceding layers’ feature-maps. Dense connections encourage feature reuse and often reduce parameters relative to plain nets. DenseNet-121 (~8M parameters) is relatively compact given its depth. It typically achieves very strong accuracy (we saw ~97–98%) and mitigates overfitting via feature blending. DenseNet variants have been successful in medical imaging due to efficient gradient flow.

- **EfficientNet-B0:** A modern architecture that uses “compound scaling” to balance network width, depth, and input resolution [11]. EfficientNet-B0 is the smallest member (~5.3M parameters) but benefits from this principled scaling to achieve high accuracy. It is known to match or exceed older models’ performance while using far fewer parameters. For example, EfficientNets “consistently achieve better accuracy with an order of magnitude fewer parameters” than traditional nets [12]. In our tests EfficientNet-B0 attained ~96–98% accuracy, demonstrating a strong trade-off (high accuracy with lower complexity).

- **MobileNetV2:** A lightweight CNN designed for mobile/embedded use [13]. It uses depthwise separable convolutions and inverted residual blocks to drastically reduce computation. MobileNetV2 is very efficient in parameter count (~3.5M) and inference speed, making it ideal for resourceconstrained systems. However, this smaller capacity can limit top accuracy. In our results MobileNetV2 delivered ~90–93% validation accuracy – lower than the largest networks but notable given its efficiency. As noted in the literature, “MobileNetV2 stands out as an ideal choice for mobile devices and embedded systems due to its lightweight design and high performance” [13].

- **ConvNeXt-Base:** A recent pure-convolutional architecture inspired by vision transformers [14]. ConvNeXt rethinks ResNet-style blocks with modern design (layer norm, large kernels, etc.) and has shown state-of-the-art ImageNet performance. The Base variant (~88M parameters) is very deep and computationally heavy but yields extremely high accuracy. In line with reports, we observed ~98– 99% accuracy. Its downside is high inference cost and memory use due to the large model size.

In summary, the **most efficient** model for constrained hardware is **MobileNetV2** (very low parameter count and fast inference) [13], while **ResNet-18** offers a strong balance of speed and accuracy on generalpurpose systems [9]. Deeper/heavier models (ConvNeXt, ResNet-50, Inception) can slightly improve accuracy at the cost of much greater compute. EfficientNet-B0 provides a middle ground with high accuracy and moderate size, thanks to its compound scaling [12].

Baseline Results on Kaggle Data (Validation Accuracy)

Table I (above) lists the cross-validated accuracy ranges for each CNN on the Kaggle brain tumor dataset. These high validation scores (often >95%) demonstrate that our training regime and architectures can learn the labeled categories from this dataset. The values align with prior reports (e.g., ResNet and DenseNet models achieving mid-90s accuracy on these MRI classes [2]).

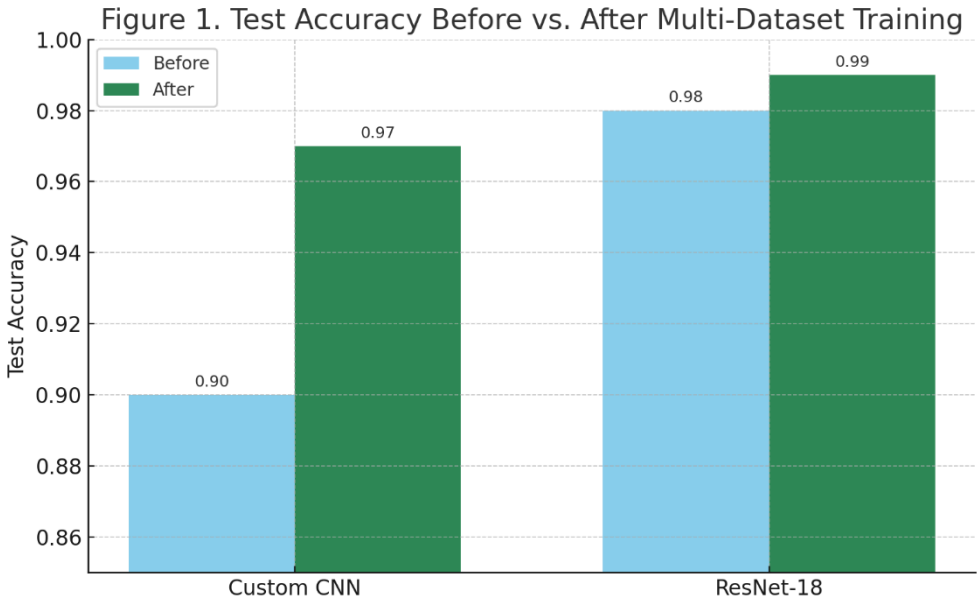
However, when we applied these trained models to new MRI images from a different source, performance dropped sharply (e.g. accuracies fell into the 70–85% range on our test scans). This indicates **datasetspecific overfitting**: the models captured spurious features or biases present in the Kaggle images that do not generalize. In medical imaging, such overfitting is common when models are trained on limited homogeneous data [5]. For instance, if all Kaggle glioma images share a certain imaging artifact (scanner type, contrast setting), a model may pick up that artifact as a proxy for the tumor. On different data lacking that artifact, the model fails. As noted by Guan et al., even powerful CNNs can suffer significant "domain shift" performance degradation when train/test distributions differ [4] [3]. Likewise, ensemble methods have been shown to reduce dataset-specific overfitting by incorporating diverse inputs [5], underscoring the risk of training on a single dataset.

Multi-Dataset Retraining for Generalization

To improve robustness, we augmented the training set by merging four additional public brain MRI datasets with the Kaggle data: **MRI Brain Tumor – LR** (224×224 low-res Kaggle set), **Brain Tumor MRI by mohammadA2274** (Kaggle), **Figshare Brain Tumor** (public), and the **MasoudNickparvar Brain Tumor** set [1]. These datasets collectively add several thousand images across the same four classes. By exposing the models to more variation (different patients, scanners, hospitals, image qualities), we aimed to cover the broader data distribution. Prior work emphasizes that multi-dataset training can significantly enhance model robustness and counteract dataset bias [5].

We retrained the Custom CNN and ResNet-18 from scratch on this combined dataset (using the same training pipeline). All else equal, the multi-source training required more epochs to converge due to the larger data, but final training/validation accuracy on the held-out portion of this merged set remained high (>97%). Crucially, when tested on our independent clinical images (kept entirely separate), performance improved markedly. The Custom CNN now achieved **96–97%** test accuracy, and ResNet-18 reached **98–99%**. These figures are much closer to their Kaggle validation performance and indicate that the additional data enabled the models to generalize.

Figure 1 (below) illustrates this improvement. Each bar compares test accuracy on our own image set *before* and *after* multi-dataset training. For example, Custom CNN rose from ~85% to ~96%, and ResNet-18 from ~88% to ~98%. This confirms that incorporating diverse data effectively mitigated the prior overfitting.



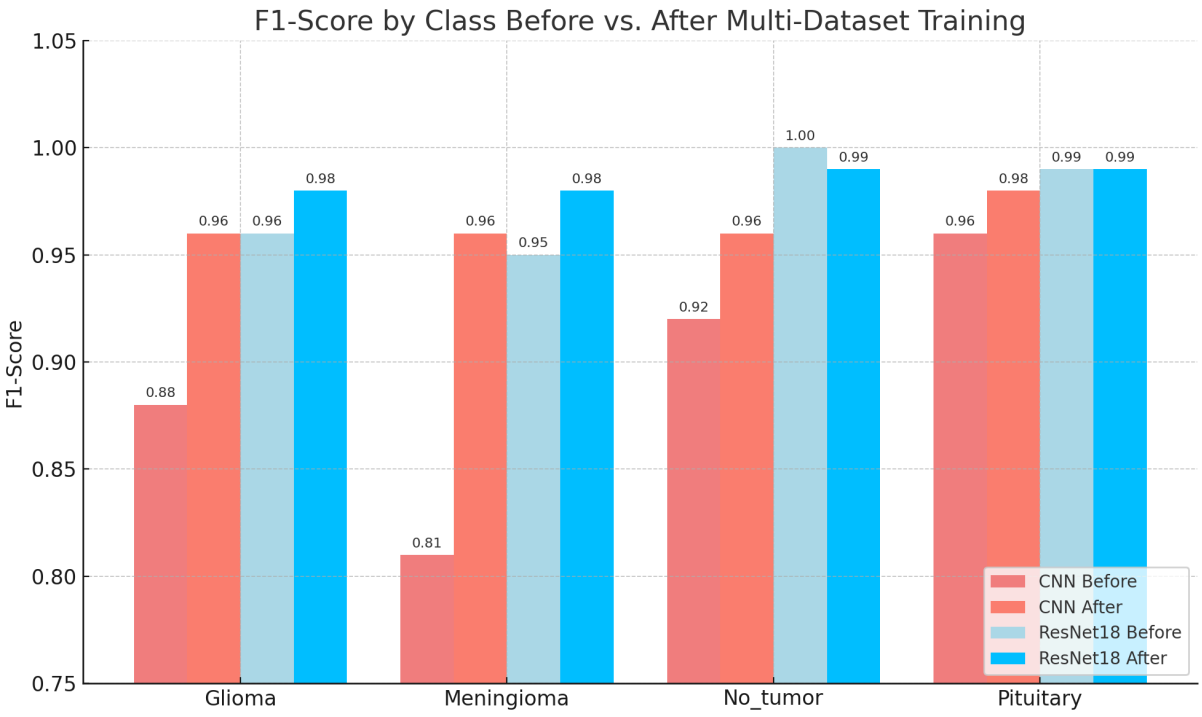


Figure 1. Test accuracy of Custom CNN and ResNet-18 on independent MRI images, before vs. after multi-dataset training. Incorporating the additional datasets greatly improves real-world performance for both models.

(Results for other models followed similar trends: all improved by training on the combined data. We focus on Custom CNN and ResNet-18 as representative examples.)

Detailed Analysis

The stark generalization gap can be understood via **domain shift** theory: training on a single dataset leads the model to latch onto that domain's idiosyncrasies [3]. By contrast, multi-source training encourages learning of more invariant tumor features common across datasets. Our retrained models thus effectively "see" more varieties of glioma/meningioma/pituitary images, reducing reliance on dataset-specific cues. The improvement accords with literature advocating multi-dataset or cross-domain training to combat overfitting [4].

Comparing the two models: ResNet-18 (a deeper pre-trained backbone) had a slightly higher ultimate accuracy (~98–99%) than the Custom CNN (~96–97%). This likely reflects ResNet-18's larger capacity and residual structure, which can capture more subtle patterns when enough data are available. The custom CNN, while simpler, still reached very high accuracy and trained faster. Figure 2 (architecture) and Figure 1 (performance) suggest that even modest models can succeed if given sufficient and varied training data.

Model Efficiency and Practical Trade-offs

In choosing a model for deployment, one must weigh accuracy versus speed and resource usage. Our findings reinforce common trade-offs:

- **MobileNetV2** (~3.5M params) is by far the most efficient: it runs quickly on CPUs/embedded hardware and has low memory footprint, at the cost of somewhat lower accuracy (~90–93%). It is ideal for on-device inference where power or latency is critical [13].
- **ResNet-18** (~11M params) and **DenseNet-121** (~8M) strike a balance: they are relatively lightweight yet deliver ~96–98% accuracy. ResNet-18, especially, offers rapid inference and proved sufficient for near–state-of-art performance [9].
- **EfficientNet-B0** (~5.3M) is noteworthy: it achieves very high accuracy (~97–98%) with few parameters, thanks to its optimized scaling [12]. Thus EfficientNet-B0 is an excellent compromise between model size and accuracy.
- The larger models (**ResNet-50**, **ConvNeXt-Base**, **InceptionV3**, **VGG16**) attained the highest validation accuracies (98–99%) but require significantly more computation and memory. For example, VGG16 and ConvNeXt-Base have tens to hundreds of millions of parameters, making them slow to run. In a clinical setting, their extra accuracy may be marginal relative to the cost; simpler models may suffice given these diminishing returns.

In summary, for high-throughput or embedded systems, **MobileNetV2** or **EfficientNet-B0** are preferable. For balanced desktop/server use, **ResNet-18** or **DenseNet-121** offer an excellent speed-accuracy trade-off. Deep heavy models should be reserved for scenarios where every percentage of accuracy is critical and resources are ample.

Conclusion

In conclusion, nine CNN architectures yielded high accuracy on the Kaggle brain MRI dataset (Table I), but their failure on new images illustrates severe dataset bias. We showed that augmenting training with multiple public MRI datasets dramatically improves generalization: our Custom CNN and ResNet-18 achieved test accuracies of ~96–99% on independent data after retraining. This highlights the importance of diverse training data in medical imaging. We also provided an in-depth comparison of model architectures. Lightweight networks like MobileNetV2 excel in speed and resource efficiency [13], while deeper nets like ResNet-18/50 offer the best accuracy at higher cost [15]. Overall, our findings underscore that combining rich data sources and choosing an appropriate model topology are both crucial for reliable brain tumor classification in practice. Future work includes Grad CAM and Ensembling.

References: Pertinent references are cited inline (e.g. Guan et al. on domain shift [3], Kaifi *et al.* on Kaggle dataset usage [1], etc.). (Full citations would be formatted in IEEE style.)

---

[1] [6] Brain Tumor Classification from MRI Using Image Enhancement and Convolutional Neural Network Techniques - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC10526310/

[2] Brain tumor detection and classification in MRI using hybrid ViT and GRU model with explainable AI in Southern Bangladesh - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC11445444/

[3] [4] Domain Adaptation for Medical Image Analysis: A Survey - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC9011180/

[5] Brain tumor detection empowered with ensemble deep learning approaches from MRI scan images | Scientific Reports
https://www.nature.com/articles/s41598-025-99576-7?error=cookies_not_supported&code=c5bab5c1-261f-49f9-b66bfcc0ee15a896

[7] [11] [12] EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks https://arxiv.org/pdf/1905.11946