Cost-Effective Brain Tumor Detection with CNNs

Jefrey Bergl[1], Yanni Etchi[1], Tripp Whaley[1], and Jeffrey Zhu[1]

[1]University of North Carolina at Chapel Hill

1 Introduction

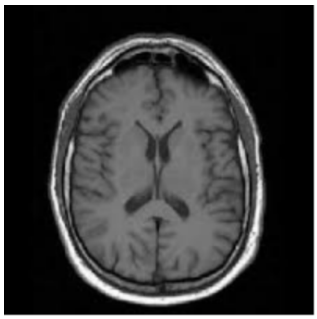
Our goal is to achieve accurate results in classifying MRI images of human brains to classify whether they show a tumor. We ultimately achieve performance close to more modern, deeper neural networks using a simple hyperparameter-tuned Convolutional Neural Network (CNN). Additionally, we achieve a lower False Negative rate than all other architectures, which we argue is critical in the domain of healthcare. Our code is made available through a Github repository.

2 Motivation

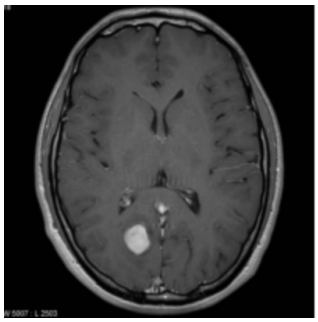
Cancer is the second most leading cause of death within the United States. The most effective way to prevent cancer is through aggressive early screening and early treatment, preventing any tumor from potentially becoming a problem. Unfortunately, the percentage of people who regularly screen for cancer has actually decreased over the years. Additionally, rates of cancer are increasing year over year, and the number of qualified oncologists cannot keep up. Although current machine learning models cannot and should not directly replace medical oncologists, they can serve as useful tools for early screening to help oncologists manage a heavier workload faster than previously required. As a screening tool, the goal should be to ensure that tumors are classified as tumors as often as possible. Ideally accuracy would always be perfect, but we aim to err on the side of caution. While reporting too many false positives puts an unnecessary burden on oncology teams, the risk of missing a true positive is potentially life-and-death.

3 Datasets

For this task we select a dataset of Magnetic Resonance Imaging (MRI) images from KaggleChakrabarty (2018). The dataset contains a small number of anonymous images in PNG and JPEG formats broken down into two classes: 155 positive (image contains a brain tumor) instances and 98 negative instances (image does not contain a brain tumor) for a total of 253 images. Each image is mono-channel (black and white) with varying resolutions.



(a) Negative examples from the dataset (No tumor)



(b) Positive examples from the dataset (Tumor)

Figure 1: Samples of negative and positive tumor classifications

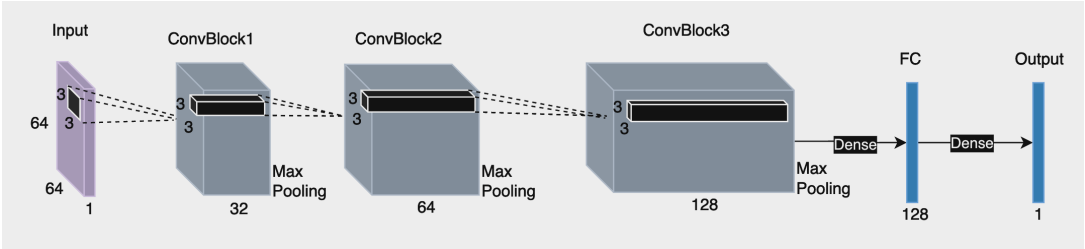


Figure 2: Baseline CNN architecture

We split this dataset into a train/test/validation split of 70/15/15. Additionally, due to the low number of samples in the dataset, we perform a series of augmentations five times for each image: rotation of 20 degrees, horizontal flip, width shift of 20%, height shift of 20%, zoom (in and out) of 20%, shear: 20%, and finally scale to 64x64 pixels for consistency.

Figure 1a and Figure 1b show negative and positive examples respectively from the dataset without any data augmentations applied.

4 Methods

4.1 Baseline Convolutional Neural Network

We start by building a simple CNN architecture illustrated by Figure 2. This model consisted of three convolutional blocks. The first block used 32 3x3 pixel filters, a batch normalization step, a max pooling step 2x2 pixels, and a dropout rate of 0.3 to reduce over-fitting. The second block was the same architecturally but with 64 3x3 filters, and the final block used 128 3x3 filters and increased the dropout rate to 0.4. All convolutional layers uses ReLU activation to mitigate issues caused by vanishing gradients. After the convolutional blocks, a GlobalAveragePooling2D layer also used to prevent overfitting. The next layer is a fully dense 128 neuron layer with ReLU activation. Finally, the last classification layer is the sigmoid function, which simply outputs a value from 0 to 1 to represent probability of a positive classification (existence of tumor). The final baseline model was trained for 50 epochs with a batch size of 32.

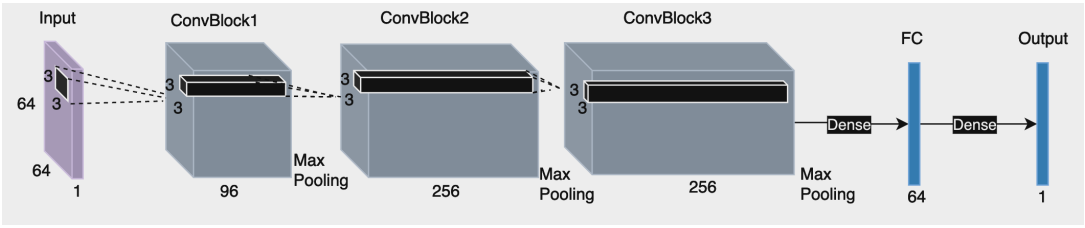


Figure 3: Hyperparameter-tuned CNN architecture

Hyperparam	Min Value	Max Value	Step Size
Block 1 Filters	32	128	32
Block 2 Filters	64	256	64
Block 3 Filters	128	512	128
Block 1 Dropout	0.2	0.5	0.1
Block 2 Dropout	0.2	0.5	0.1
Block 3 Dropout	0.3	0.6	0.1
Dense Layer Units	64	256	64
Dense Layer Dropout	0.3	0.6	0.1
Learning Rate	1e-5	1e-2	logspace

Table 1: Hyperparameter search space

4.2 Tuning Convolutional Neural Network

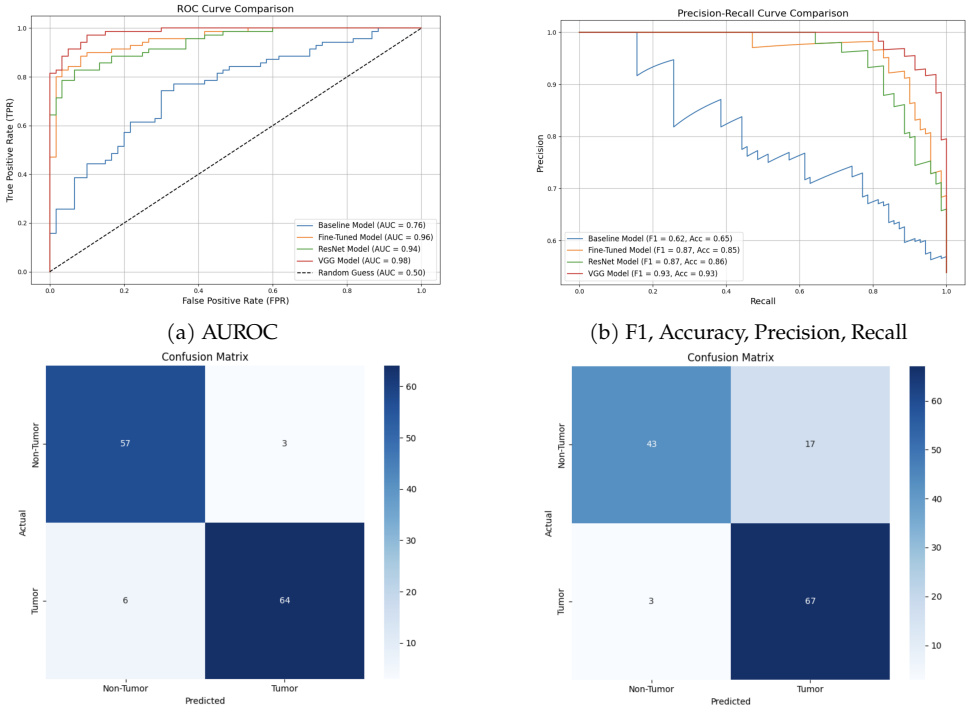
Hyperparameter tuning was completed with Keras' RandomSearch with 30 different trials with hyperparameters ranging as noted in Table 1. Each of the 30 models was evaluated, and the best was chosen as our hyperparameter tuned model with 96 3x3 filters and dropout rate of 0.3 for the first block, 256 3x3 filters and dropout rate of 0.2 for the second block, 256 3x3 filters dropout rate of 0.2 for the third block, then 64 neurons with a dropout rate of 0.4 in the final fully connected layer with a learning rate of 1.22e-3, which yielded significant gains in validation accuracy. The final model architecture is also illustrated in Figure 3.

4.3 Pre-trained Modern Neural Networks

We then used pre-trained ResNet-50He et al. (2015) and VGG-16Simonyan and Zisserman (2015) neural networks for more modern comparison models. For each, we unfreeze the last 20 layers, as we only need to detect the more complex features of the image and want to avoid catastrophic forgetting of base weights. Then we additionally add a few extra layers to the networks to enable them to serve as binary classifiers like our CNN: flattening, a dense layer with 256 neurons using ReLU activation, a dropout rate of 0.5, and a final 1 unit Sigmoid function activation layer for classification. Both of these models are trained for 20 epochs with a batch size of 32.

5 Results

Full results are captured in the ROC curves in Figure 4. We note that while our model did not outperform the VGG-16 model, it did perform on par with the ResNet-50 model in terms of F1 (VGG=0.93, ResNet=0.87, ours=0.87) and Accuracy (VGG=0.93, ResNet=0.86, ours=0.85) despite having significantly less layers. However, our model did significantly reduce the False Negative rate and slightly increase the rate of True Positives as can be seen comparing the VGG and our confusion matrices in Figure 4c



(a) AUROC

(b) F1, Accuracy, Precision, Recall

(c) VGG-16 Confusion Matrix

(d) Our tuned CNN Confusion Matrix

Figure 4: Full results for all models and confusion matrices for ours vs. best (VGG)

and Figure 4d.

6 Conclusions

We conclude the best model performance came from VGG-16, with our simpler hyperparameter-tuned CNN following closely behind in terms of AUROC and F1 score. However, our model did achieve a significantly lower False Negative rate compared to the VGG architecture and also correctly classified more positive instances as tumors correctly. The problem space presented may seem simplified compared to many modern machine learning problems, but many rural and low-income healthcare facilities do not have access to the latest MRI equipment, so achieving safe and reliable results with gray-scale MRI scans on a limited budget is an important and realistic problem. While the False Positive rate is fairly high, which does incur a cost for medical facilities, we argue that minimizing False Negatives (without simply creating a trivial model) is the most important criterion for a prescreening model. Given the significantly smaller parameter size of our model vs. VGG-16 and ResNet-50, healthcare systems could absolutely use this as a prescreener for oncology teams knowing that the number of False Negatives is as safe if not safer than a SOTA model and can be deployed at a significantly lower cost.

References

Navoneel Chakrabarty. Brain MRI Images for Brain Tumor Detection, 2018. URL https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015. URL http://arxiv.org/abs/1512.03385. arXiv:1512.03385 [cs].

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, April 2015. URL http://arxiv.org/abs/1409.1556. arXiv:1409.1556 [cs].