Brain Tumor Detection with DNN's

Andrew Harrop
Faculty of Engnineering
Western University
London, Canada
[redacted]

Arsh Lalani
Faculty of Engineering
Western University
London, Canada
[redacted]

***Abstract*—Brain Tumors are an exceptionally lethal cancer with overrepresentation in children and young adults. Tumor detection at early stages is ideal. Using deep neural networks to classify images has been an effective use case in the past decade, and it achieves high accuracy even with multiclass datasets. The constraint for deep neural networks is generally computational resources. This study achieved 96% accuracy on a residual neural network using 3200 images in the training, validation, and testing processes. The extraordinarily high accuracy achieved demonstrates the power of convolutional neural networks in binary classification use cases with consumer-grade computational resources. The goal of this project was to discover a robust model that can be referenced by future researchers seeking to improve tumor detection.**

Keywords—***Brain Tumor Detection, Medical Image Classification, Medical Image Processing, MRI Cross-Sections, Machine Learning***

I. Introduction

The proportion of new cancer cases represented by brain cancer is 1.35% for all Canadian demographics, which translates to roughly 3100 Canadians diagnosed with it every year. However, it causes a concerning 2400 deaths per year, which means the overall survival rate is approximately 22.4% [1]. The lifetime probability of developing brain cancer is 1 out of every 155 Canadians, 17% of new cancer cases in children from childbirth to age 14 are in the brain, and five percent of new cancer cases between ages 14 to 29 are brain tumors. The severity here is made clear by the 41% of cancer-related deaths in children under the age of fourteen that are attributed to brain cancer [2].

Because of the complexity and importance of the brain, it is necessary that brain tumors are detected as soon as possible. Early-stage tumors may be dismissed as a gyrus feature or not noticed at all even by trained professionals. If a technological solution exists to improve the probability of the detection of tumors, it is necessary to leverage computational resources to catch these early-stage growths, especially given that brain tumors are excessively destructive to humans in the early stages of life.

In this paper, different methods to classify tumors are proposed. The data is sourced from several different public datasets from Kaggle. Python is used with common libraries such numpy, matplotlib, cv2, tensorflow, and scikit-learn used to reduce development time, readability, and data usability. The target accuracy is 90% for the best model discovered, which is in line with models found in similar academic studies.

II. Related Work

In addition to the past works to classify brain tumors with machine learning from MRI cross-sections, attempts have also been made for early Alzheimer's detection. This pertains to the larger field of medical image processing, where analysts seek to gain medical insight from medical images.

Notable works regarding brain tumors are Çinar & Yildirim, who found a convolutional neural network that achieved 97.2% accuracy [3]. Lotlikar et al. perform a broader analysis on the effectiveness of a variety of machine learning models applied to brain tumor detection from images of MRI cross-sections [4]. It is important to compare other metrics such as specificity, sensitivity, and precision between different studies to develop a comprehensive picture of what has already been done and what can be improved on.

The academic consensus is that convolutional neural networks are the best option for this use case. Optimizing hyperparameters will distinguish future studies from existing ones, so that will be a focal point for this study. Grid search, Bayesian optimization, and gradient-based optimization are all viable methods to determine the optimal hyperparameters.

III. Methods and Materials

The methods used to experiment on the data are native to machine learning projects and structured according to best practices while maintaining readability through a useful level of abstraction.

A. Technical Overview

The purpose of this study is to determine which deep learning classification models best classify a specific genre of medical images. The purpose of these models is to perform classification on MRI images regarding whether a brain tumor is visible. Hyperparameter optimization will be performed extensively to achieve exceptional results.

Figure 1. describes the steps that will be repeated to generate models and optimize hyperparameters. Step two and three of figure one includes a focus on hyperparameter optimization, and most of the steps outlined are expected to be iterated over.

1. Dataset loading, organization, and analysis.
2. Preprocessing the data
3. Defining a model
4. Training the model with training data
5. Testing the model with testing data
6. Comparing the results with those achieved by other models

Fig. 1. Proposed development process

B. Dataset

There are two popular datasets for MRI images of the brain with and without tumors. The datasets can be found on Kaggle, a highly reputed source in the Data Science community. The content is very similar in both datasets, and for the purpose of data diversity, it was decided that two datasets be used instead of one. The first dataset was published by Navoneel Chakrabarty and contains 155 MRI cross-sections with tumors present and 98 MRI cross-sections without tumors [5]. The second dataset, Br35H, was published by Ahmed Hamada and contains 3000 labeled MRI cross-sections in total [6] .

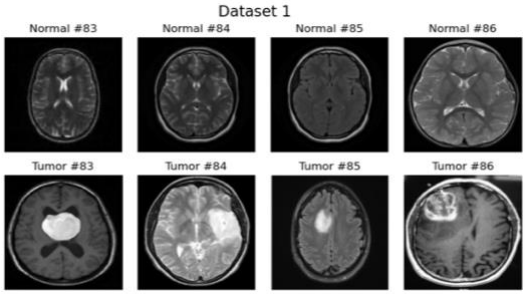

Fig. 2. A random sample of labeled MRI cross-sections from the first dataset.

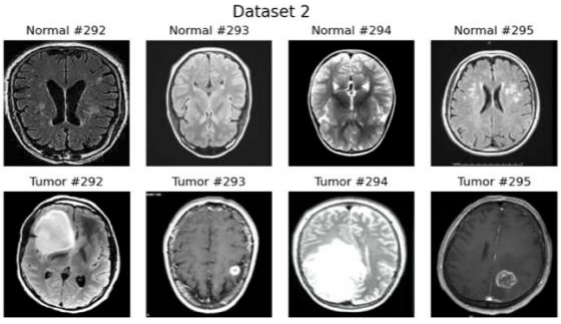

Fig. 3. A random sample of labeled MRI cross-sections from the second dataset.

C. Data Preprocessing

The goal of preprocessing is to prepare the image data in a generic and robust way. In this stage, the raw image files are converted into cv2 image objects and stored in labeled datasets. In figure 2 and figure 3, it is evident that a significant amount of unused space exists in some images beyond the extreme bounds of the brain.

The first step in modifying the images taken was to crop the images to fill with the extreme edges of the brain on the edges of the images. The specific process used is detailed in "Finding extreme points in contours with OpenCV" by Adrian Rosebrock [7]. The purpose of this step was to improve model efficiency by removing input attributes that do not contribute to the problem at hand.

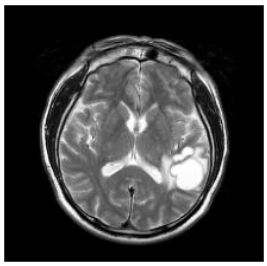

Fig. 4. Uncropped MRI.

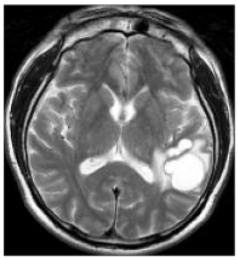

Fig. 5. Cropped MRI.

The next step was data augmentation. The angle, width, height, scale, and brightness were all randomly varied, as well as reflections over the horizontal and vertical axis. The purpose of the augmentation was to increase the robustness of models trained on it.

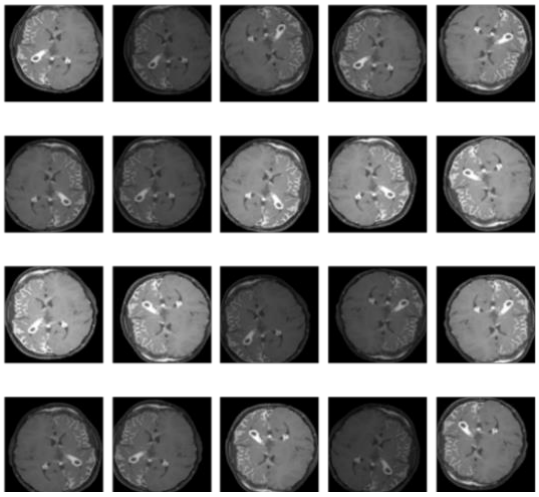

Fig. 6. Twenty augmentations of an MRI scan.

One important step taken in preprocessing was the creation of two sets of data to train, test, and validate models. The composite set consists of images from both datasets, which is about 3200 images in total. The small set consists of approximately 250 images, which is useful when testing new procedures because it involves ten percent of the composite set.

The steps after the generic preprocessing vary depending on the model used. The model training process varies because different attributes are valuable to different algorithms. Some of the data is placed into tensorflow generators specific to certain models to increase speed and optimize training. Other data is provided to models directly in covariate-label pairs. In the final iterations of training the models, the entirety of the approximately 3200 images were used to train, validate, and test the models, but for most of the project, only dataset 1 is used for the sake of time and computational resource preservation.

IV. Models

A. VGG-16

The VGG-16 model is a 16-layer convolutional neural network proposed by Simonyan and Zisserman in 2015. It emphasizes increasing network depth with small convolution filters, which is proven to be effective but requires significant time and computational resources [8]. Because of the

computational overhead required, a pretrained network is implemented and optimized instead of training the network from scratch, allowing lower runtime and more potential for application on large datasets with limited resources.

The loss function used was binary cross entropy. The binary cross entropy loss function is defined as:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\big(p(y_i)\big) + (1 - y_i) \cdot \log\big(1 - p(y_i)\big)$$

Fig. 7. Binary cross entropy loss function.

B. *Basic CNN*

This model is untrained and build from the traditional definition of a CNN containing convolutional and pooling layers. For this specific model, only one convolutional layer is used. The convolutional layer is configured with a kernel size of 7x7. There are 32 filters in the convolutional layer, a number that was optimized after trial and error.

This model employs the binary cross entropy loss function seen in figure 7.

C. *ResNet50*

This model is a modification of ResNet, proposed by experts at Microsoft in 2015. Residual Neural Networks solve the vanishing gradient problem that arises in training when gradients approach negligible values and no useful work is done by backpropagation. The solution is to skip over layers, which provides a method to maintain a deeper network than other models while achieving the same computational performance.

This model employs the binary cross entropy loss function seen in figure 7.

V. Experemental Results

A. *Quantification*

The experimentation is quantified in generic proportional representations of the results achieved. The values used are unique to binary classification problems and provide insight into different aspects of model effectiveness.

TP: True Positives
TN: True Negatives
FP: False Positives
FN: False Negatives

Which are used to construct indicators such as Accuracy, Specificity, Sensitivity, Precision, and the F1-Score.

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

$$Sensitivity = \frac{TP}{(TP+FN)}$$

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

$$F1\ Score = 2 * \frac{(Precision * Sensitivity)}{(Precision + Sensitivity)}$$

B. *Model Results*

Model	Epochs	Time to train (s)
Basic CNN	20	2177
ResNet50	20	577
VGG-16	20	3616

Fig. 8. The number of epochs and runtime for each model.

1) *Basic CNN*

The basic CNN achieved an accuracy of 95% when trained with the full dataset. These values were achieved with 20 epochs and a batch size of 32. Figure 8, Figure 9, and Figure 10 contain some of the information obtained after and during the model training. After testing the model with 100 epochs, it was determined that there is no testing performance improvement after 20 epochs and testing performance degrades after 40 epochs. The model generated from the training can be found in the project repository [9].

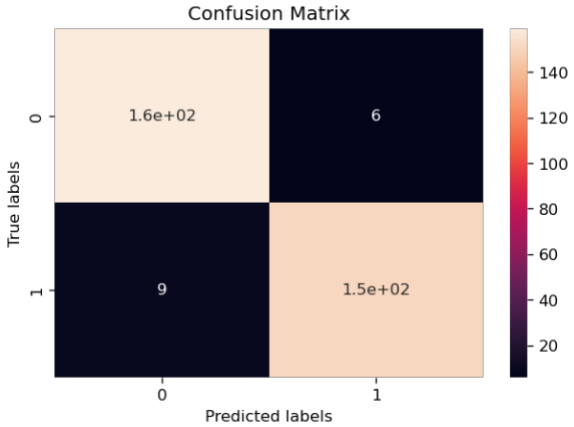

Fig. 9. The confusion matrix of the predicted labels from the basic CNN on the testing data.

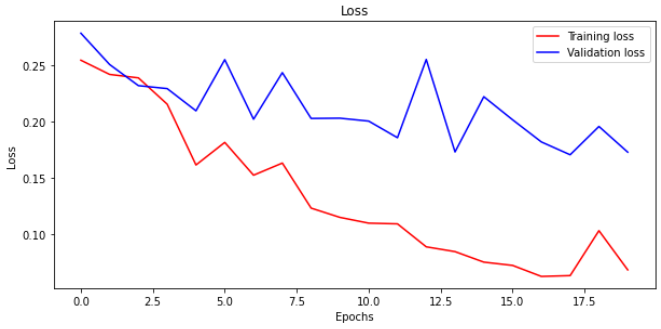

Fig. 10. The loss over 20 epochs of model training with the basic CNN.

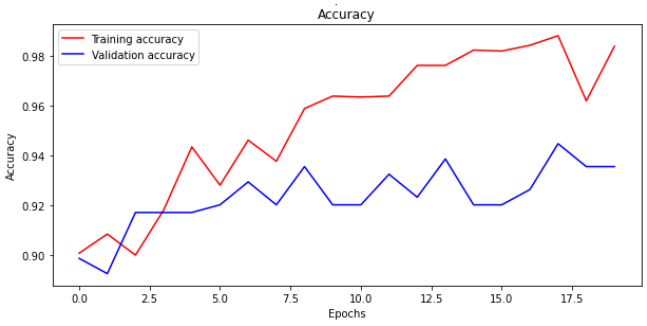

Fig. 11. The accuracy over 20 epochs of model training with the basic CNN.

2) ResNet50

The ResNet50 model achieved a 96% accuracy when trained with the full dataset. These values were achieved with 20 epochs and a batch size of 8. The high accuracy came at a performance trade-off but is still feasible for consumer grade hardware. The robust model produced by the training on approximately 2500 images in one hour and 16 seconds proves that it is possible to simplify critical image classification problems into accurate and robust neural networks. Figure 12, Figure 13, and Figure 14 display some of the information pertaining to this model.

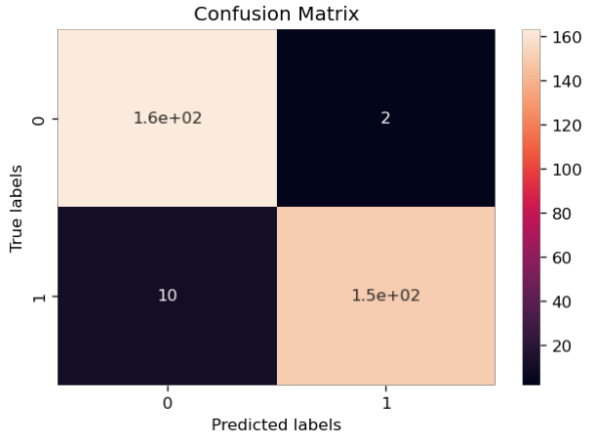

Fig. 12. The confusion matrix of the predicted labels from the ResNet50 model on the testing data.

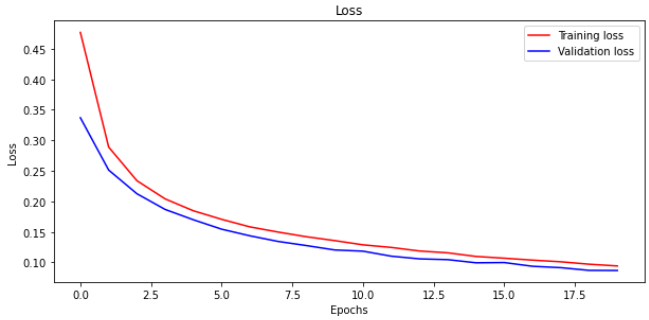

Fig. 13. The Loss over 20 epochs of model training with ResNet50.

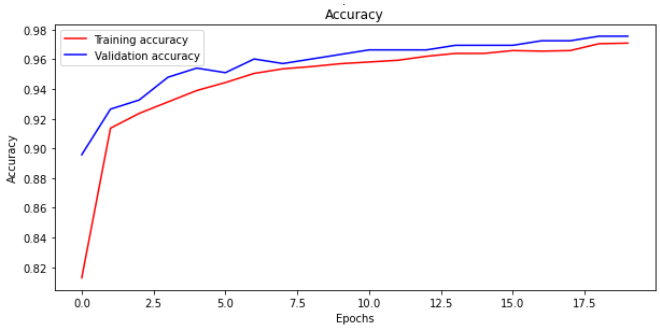

Fig. 14. The accuracy over 20 epochs of model training with ResNet50.

3) VGG-16

The VGG-16 model produced the most disappointing results from the models tested. 20 epochs with a batch size of 32 were used for training. An accuracy of 82% was achieved, but it took significantly less time to train than the basic CNN or ResNet50. The trade-off between speed and accuracy is not important if the VGG-16 model cannot achieve better computational and time performance with only a slightly lower accuracy. The difference between 82% and over 95% is significant enough to warrant either extensive optimization and increased runtime or the discontinuation of research with VGG-16 for brain tumor classification. Figure 14, Figure 15, and Figure 16 display some of the information generated during the training and testing of the VGG-16 model.

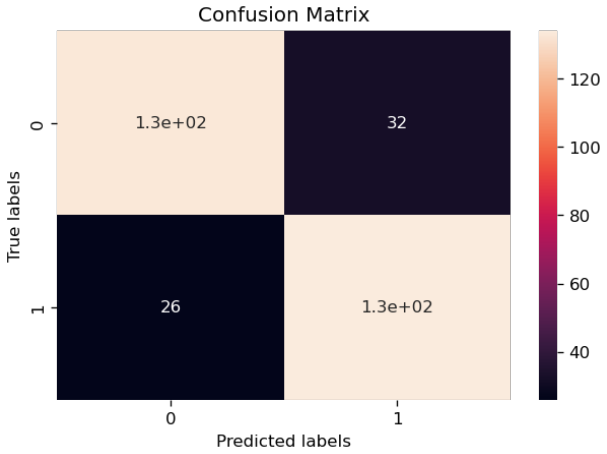

Fig. 15. The confusion matrix of the predicted labels from the VGG-16 model on the testing data.

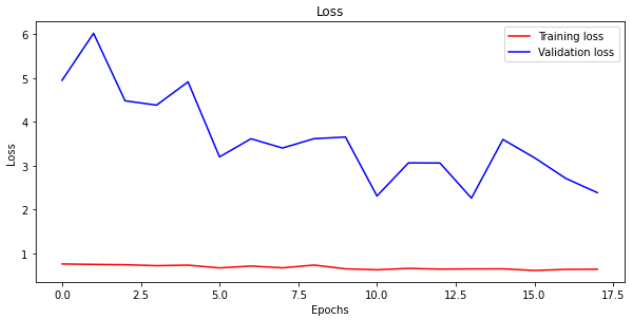

Fig. 16. The loss over 20 epochs of model training with VGG-16.

Fig. 17. The accuracy over 20 epochs of model training with VGG-16.

VI. Conclusion

The custom CNN and ResNet50 achieved similar results, however the custom CNN took over 30% less time to train, suggesting that it may be the best model given more resources and research for this specific case. In future, research into optimizing the ResNet50 model and improving the accuracy

of simple CNN's could yield extensive benefits in medical image classification. Classifying brain tumors is a problem that could benefit society significantly if reliably modelled. The algorithms used in this project could also be applied to other medical image classification problems such as general tumor detection, fetal abnormality detection, and degenerative disorder pattern recognition.

VII. References

[1] S. Lee, "Brain and spinal cord cancer statistics," *Canadian Cancer Society*. [Online]. Available: https://cancer.ca/en/cancer-information/cancer-types/brain-and-spinal-cord/statistics. [Accessed: 08-Apr-2022].

[2] "Cdn.cancer.ca," *Canadian Cancer Society*. [Online]. Available: https://cdn.cancer.ca/-/media/files/research/cancer-statistics/2021-statistics/2021-pdf-en-final.pdf. [Accessed: 08-Apr-2022].

[3] Ç. A. Y. M; "Detection of tumors on brain MRI images using the hybrid convolutional neural network architecture," *Medical hypotheses*. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/32240877/. [Accessed: 08-Apr-2022].

[4] L. V. S. S. N. G. A; "Brain tumor detection using machine learning and Deep Learning: A Review," *Current medical imaging*. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/34561990/. [Accessed: 08-Apr-2022].

[5] N. Chakrabarty, "Brain MRI images for Brain tumor detection," *Kaggle*, 14-Apr-2019. [Online]. Available: https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection. [Accessed: 08-Apr-2022].

[6] A. Hamada, "BR35H :: Brain tumor detection 2020," *Kaggle*, 14-Nov-2021. [Online]. Available: https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection. [Accessed: 08-Apr-2022].

[7] A. Rosebrock, "Finding extreme points in contours with opencv," *PyImageSearch*, 17-Apr-2021. [Online]. Available: https://pyimagesearch.com/2016/04/11/finding-extreme-points-in-contours-with-opencv/. [Accessed: 08-Apr-2022].

[8] "Abstract arxiv:1409.1556v6 [CS.CV] 10 apr 2015." [Online]. Available: https://arxiv.org/pdf/1409.1556. [Accessed: 08-Apr-2022].

[9] A. Harrop and A. Lalani, "Andrewharrop/Advanced-ai-Thesis: Thesis for CS4442: Advanced AI," *GitHub*. [Online]. Available: https://github.com/andrewharrop/Advanced-AI-Thesis. [Accessed: 08-Apr-2022].