Convolutional Neural Network (CNN) Binary Brain Tumor Classification

William W. Winters

Anderson College of Business and Computing, Regis University

MSDS 686 Deep Learning

Dr. Ghulam Majtaba

March 9, 2025

Convolutional Neural Network (CNN) Binary Brain Tumor Classification

Intercranial neoplasms (brain tumors) are among the most devastating diseases that result in a significant reduction of quality of life or life expectancy. Often symptoms such as headaches, changes of personality, irritability and other symptoms are not correctly diagnosed as being brain tumors. In addition, some tumors may be difficult to identify on MRI imagery and are overlooked in the diagnostic process. The golden standard of care brain tumors is 100% resection (removal) to prevent reoccurrence, and being able to accurately interpret and diagnose residual tumor from postoperative MRI brain images is essential to treating them.

Many Kaggle studies have been conducted to classify brain tumors using convolutional neural networks (CNN)s, but most of these focused on classifying the type of tumor into for groups: glioma, meningioma, pituitary, and no tumor (Roy, et al., 2024). In this study the emphasis will be on developing a CNN that can examine MRI images and determine if a tumor exists or not. The goal is to use this model as a diagnostic tool that can be used in the initial examination or postoperative evaluation to determine if any residual tumor remains.

Data

The dataset used in this effort is a combination of two data repositories available on Kaggle.com.   Two Kaggle sources were combined to form the final dataset used in this project and it contains a total of 5,100 images. The original data sources can be found in the links below.

Brain Tumor Classification (MRI) (Bhuvaji, 2020)

Brain Tumor Dataset (Viradiya, 2021)

Most of the data was taken from Viradiya's dataset, and a few images were taken from Bhuvaji's no_tumor class. The following is a breakdown of how the images were used

Class	**Training**	**Testing**	**Total**
Negative	2,065	522	2,587
Positive	2,011	402	2,513
Total	4,076	1,024	5,100

The images from Bhuvaji's dataset were randomly selected and added to this project's dataset to help balance out the classes. A copy of the dataset used in this project can be found on the author's Google Drive.

Methods

A review of current peer-review literature was reviewed to determine current trends in MRI brain tumor classification using convolutional neural networks and sources of brain tumor MRI images. From this research it was determined that many of the public image datasets are available on kaggle.com. After the datasets were procured, each one was evaluated for suitability to this study. Items considered included how well the datasets were documented, labeled and traceable. Sets that were not up to research standards were discarded and the final dataset was constructed.

Since images are involved and this is a classification problem, a convolutional neural network (CNN) is the best choice for this type of deep learning problem (Chollet, 2021). After dataset curation and model architecture selection, a Jupyter-lab notebook was used to develop the model's architecture, clean and process data, perform EDA, train the model and evaluate its performance.

Model Architecture

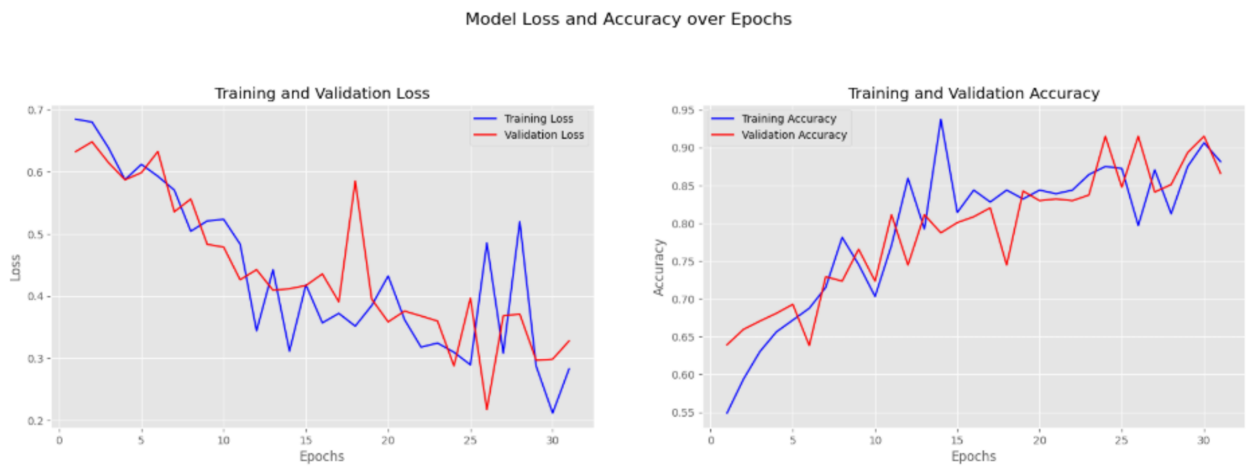
Final CNN architecture of the model used in this project consists of the following key components:

1. **Convolutional Layers**: This model uses four convolutional layers with a combination of max pooling.
2. **Activation Functions**: The convolutional layers are using the ReLU (Rectified Linear Unit) activation function, which is well suited to this type of binary classification problem
3. **Pooling Layers**: Max pooling layers are used to reduce the spatial dimensions of the feature maps while retaining the most important features
4. **Flattening Layer**: Is a crucial component in a CNN and its primary purpose is to transform the output of the convolutional and pooling layers into a one-dimension array that can be fed into a fully connected layer
5. **Dropout**: Is a technique used in CNNs to prevent overfitting and improve generalization
6. **Fully Connected Layers**: Two fully connected layers are being used to perform final classification and produce the results.

Model Training

To help prevent overfitting two callback functions were developed. The first one is designed to stop the training process when validation loss stops decreasing and the second will reduce the learning rate until validation loss plateaus. The author adjusted several arguments for both functions until satisfactory results with training were obtained. In addition, to the callbacks, the Adam optimizer was utilized for model training and the default learning rate of 0.001 was utilized. The beta_1 and beta_2 arguments were adjusted to improve performance. Furthermore, the model uses the categorical cross entropy loss function with metrics of accuracy, precision, recall, and AUC.

Model training with the Adam optimizer and the two callback functions averaged between 30 to 40 epochs per training session. The training and validation loss and accuracy closely parallel each other indicating over/under fitting is minimal. See training/validation loss and accuracy plots below



Results

Standard evaluation methods were used to determine the accuracy of the model. The following results were observed:

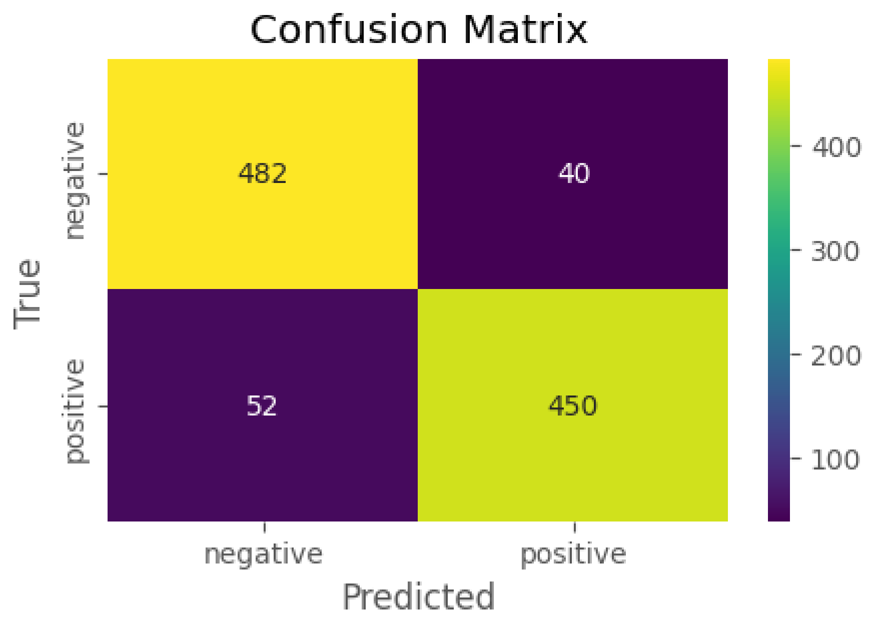
- **Accuracy**: The model consistently scored ***above 0.90 (90%)*** on the test data. This shows the model is generalizing the test data well and is accurately predicting classes
- **AUC**: Area Under Curve is a representation of the classifier's performance and indicates how well classification is taking place. The model used in this project consistently scored ***above 0.90 (90%)*** in this category.

- **F1 Score**: Is a measure of the model's accuracy on a classification problem. This project's model consistently scored ***above 0.90 (90%)*** in this area.
- **TPR**: True Positive Rate is the measure of how well the model predicts true positives. Here again the model consistently performed ***above 0.90 (90%)***.
- **TNR**: True Negative Rate determines how well a model predicts negative classes. Like the TRP reading, this model consistently scored ***above 0.90 (90%)*** in this area.

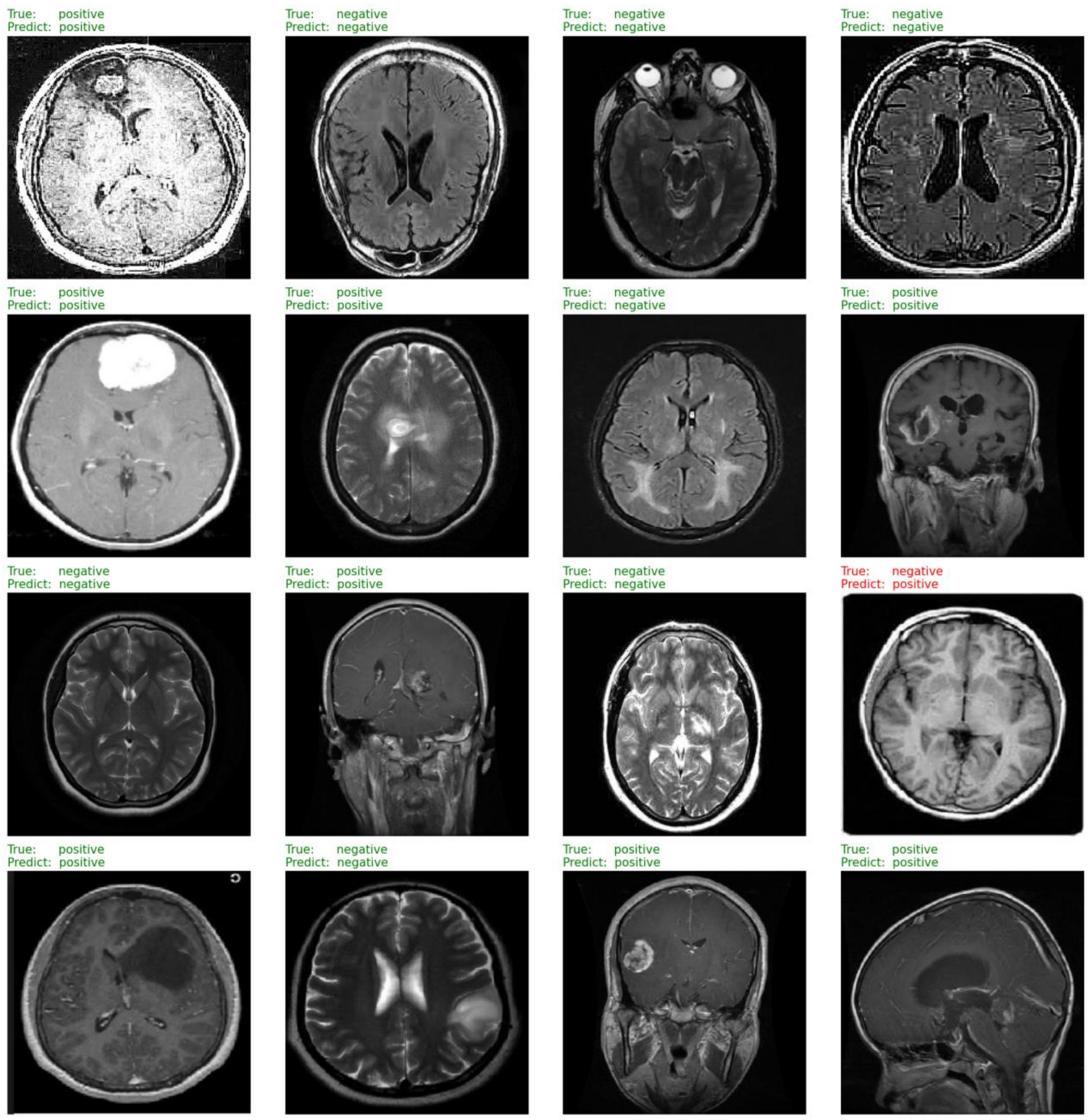
Based on the training plots and the results of evaluating the model against the test dataset, the model is consistently making accurate predictions over 90% of the time and does well with predicting the positive and negative classes. It is generalizing the test data very well. Teh AUC and F1 scores are in the 90s which indicate the model is correctly predicting the classes over 90% of the time. The table below summarizes the results.

Metric	**Score**
Loss	0.2208
Accuracy	0.9102
Precision	0.9102
Recall	0.9696
AUC	0.9102
F1 Score	0.9389

Furthermore, a comparison of the True Positive Rate (TPR) and True Negative Rate (TNR) and the confusion matrix confirm the model is correctly predicting on the test data close to 90% of the time.   The TPR and TNR can be summarized as follows: TPR 0.8964 and TNR 0.9234. The confusion matrix below provides a visualization of the true and predicted true positive (TP), true negative (TN), false positive (FP) and false negative FN).



To help visualize how well the model is performing a sample of test images was selected and displayed with their true and predicted labels.



Conclusion

The process of developing a study of this type can be time consuming. Curating useful public images proved to be problematic; however, a useful dataset was collected. Furthermore, the

CNN model training process selects weights differently for each training run causing different results each time. I developed the habit of saving each training run's weights for future use.

The model developed and tested in this study performed well, although the training dataset was rather small. Image augmentation helped make up for this deficiency but was not ideal. I also found that keras' *ImageDataGenerator()* method is depreciated but could not find an alternative method to use; however, it still works. Performance of the model is acceptable by more standards but for medical diagnosis, the accuracy rate should be in the high 90s and as close to 100% as possible.

References

Bhuvaji, S. (2020, May 24). *Brain tumor classification (MRI)*. Kaggle. https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri

Chollet, F. (2022). *Deep learning with python, second edition*. Manning Publications.

Viradiya, P. (2021, May 16). *Brian tumor dataset*. Kaggle. https://www.kaggle.com/datasets/preetviradiya/brian-tumor-dataset