CSCI 1430 Final Project Report: Tumor Detection

Wondybrain & Friends: Brian Cheong, Shafiul Haque, John Rathgeber, Nahum Workalemahu
Project Mentor: Winston Li. Brown University

Abstract

In this project, we implement a Convolutional Neural Network (CNN) capable of making diagnoses on the presence and type of brain tumors. We first explore the medical necessity and social impact of such an innovation. Then, we investigate various architectures and datasets on which to base our project, after which we produce the actual classifier. Then, we implement additional features to improve the interpretability of the program, which leads into a discussion of the limited reliability of our model in a real clinical setting. Additionally, we assess the performance of classification and its implications on the feasibility of our project. Finally, we discuss technical choices that led to the success and failures, finishing with suggestions on points of future study.

1. Introduction

For the final project to culminate this course, we thought it would be most fulfilling to engage with a real-world application that directly deals with computer vision, albeit in a suitably digestible form. As our resident biology scholar, it was Brian who proposed that Wondybrain & Friends consider tumor diagnosis as a potential project avenue. As a group, we figured that among our other options, such an undertaking would be extremely worthwhile as a solution to a pressing, real-world issue.

The underlying consequence is as apparent as its name: appropriate diagnosis for a patient who may suffer from a condition that threatens their life. Misdiagnosis or delays in treatment can severely impact patient outcomes, particularly for conditions as life-threatening as brain tumors. Studies show that upwards to 10-15% of diagnoses are erroneous; further, many hospitals in the developing world face staff shortages. As a result, developing tools to assist medical professionals in identifying tumors is not only worthwhile but also deeply rewarding in its potential to save lives. Furthermore, the project has the incredible feature of varying levels of feasibility, because there exist extensive data and research regarding the study of tumors on the web.

Consequently, we devised a sequence of implementation steps so that we may seek out our ambitious goals while maintaining the ability to retreat to a more humble approach. Our implementation procedure is delineated below:

1. Small-scale binary tumor detection software (tumor/no tumor on small dataset)
2. LIME visualizer
3. Saliency map visualizer
4. Multiclass classification on larger dataset (glioma, meningioma, pituitary, no tumor)
5. Bounding box localization for the tumors
6. 3D reconstruction of the tumor.

We eventually realized that the final two exercises were certainly too ambitious for our liking, so we aim to focus on the first four tasks.

2. Related Work

Since portions of our project required the usage of outside research and understanding how others have approached the issue, we needed to first gain some contextual information on how CNNs interact with brain tumors. Inspired by their approach, we adopted a similar methodology by focusing on CNNs for tumor classification and integrating visual tools like saliency maps to enhance interpretability. While our project operates at a smaller scale, this study offered both a technical blueprint and a benchmark for the goals we aimed to achieve. (Abdusalomov et al.) [1]

The implementation by Raul C. Sîmpetru provided a strong foundation for our binary classification task by using the VGG16 model. The approach addressed the challenges of a limited dataset and imbalanced classes by leveraging pre-trained weights and simulating data variations. The high accuracy (85%) on the test set and the clear visualization of performance metrics helped us understand how his model distinguished between tumor and non-tumor images. This project guided our implementation choices and validated the effectiveness of deep learning for brain tumor detection. [2]

The combination of multiple datasets (figshare, SARTAJ, and Br35H) helped us to enrich the dataset and better cover a diverse set of brain tumor MRI images. By identifying issues with the SARTAJ dataset—where glioma images were misclassified—we were able to refine our dataset, although our attempt at implementing a multiclass classification model initially faced challenges and yielded poor results. [5]

We also used an OpenCV resource to help implement saliency maps to enhance tumor detection. This approach involved applying general techniques to visually emphasize the most significant regions within brain images, which allowed us to pinpoint areas likely harboring tumors. By leveraging these methods, we could effectively highlight potential tumor locations, aiding in more precise detection and classification. [4]

Additionally, in order to train and test the accuracy of our given dataset, we used Google Colab, similar to how we trained our CNN model in homework 5.

3. Method

This outlines the key components of the project aimed at developing tools for brain tumor detection using CNNs and saliency maps. This includes steps for small-scale binary tumor detection, visualizing model outputs through LIME and saliency maps, and implementing multiclass classification.

The project begins with a focus on binary tumor detection, identifying whether an image contains a tumor or not. For this step, a small dataset containing images labeled as either ”tumor” or ”no tumor” is used. The CNN model is trained on this dataset to distinguish between the two classes. This task helps establish a baseline for the model's ability to detect the presence of tumors in images. We implemented data preprocessing, to make sure images are resized to the same size, model training, which we used Google Colab for, and evaluation metrics such as accuracy, precision, and recall.

This code snippet highlights the head we used.

```
        self.head = [
            Flatten(),
            Dropout(rate=0.5),
            Dense(units=1, activation='sigmoid')
        ]
```

We implemented a binary classification task using VGG16 architecture, which is known for its strong performance in image classification tasks. The code was tailored specifically for binary tumor detection, and we adapted the pre-trained weights and modifying the fully connected layers to classify images into tumor or non-tumor categories. We fine-tuned the model's parameters, including learning rate and number of epochs, to optimize accuracy.

The LIME visualizer is used to understand the decision-making process of the model, especially when working with CNNs that can be seen as ”black boxes”. LIME takes a trained CNN and explains its predictions by approximating the decision boundaries locally with interpretable models, such as linear or decision trees, for individual predictions. The LIME visualizer then provides an output that highlights regions of the input image that are positively or negatively correlated with the tumor prediction.

Saliency maps are another method to visualize which parts of the image the CNN focuses on to make its predictions. They provide insights into how the model identifies regions that are critical for the tumor/no tumor decision.

This code snippet highlights our saliency map implementation.

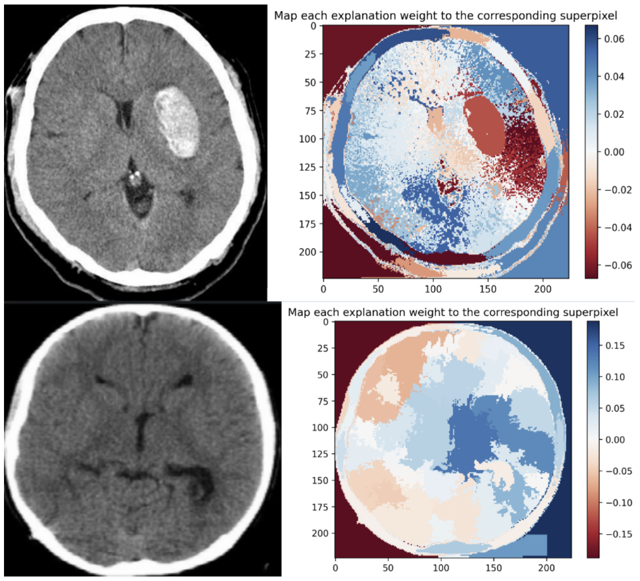
```
    with tf.GradientTape() as tape:
        tape.watch(input_image)
        predictions = model(input_image)
        class_index = tf.argmax(predictions[0])
        class_score = predictions[:, class_index]

    saliency = tape.gradient(class_score, input_image)
    saliency = tf.abs(saliency)
    saliency = tf.reduce_max(saliency, axis=-1).numpy()[0]

    saliency = (saliency - np.min(saliency)) / (np.max(saliency) - np.min(saliency) + 1e-7)
    saliency = saliency * 3

    return raw_image, saliency
```

We implemented this segment of the code by doing research on existing saliency maps for different usages. The implementation involves backpropagating the error through the network to determine which parts of the input image contribute the most to the final output. The output is a heatmap that indicates the importance of different regions in the image for the final classification.

For multiclass classification, the objective is to classify brain MRI images into one of four classes: glioma, meningioma, pituitary, or no tumor. We included a larger, balanced dataset is used to ensure that each class has sufficient samples for effective training. The dataset is split into training, validation, and test sets to evaluate the model performance. The model is trained with an appropriate optimizer like Adam and regularization techniques to prevent overfitting.

Model	Accuracy
VGG-16 (binary)	90%
Multi-class	N/A
Human Doctors	63.33%

Table 1. Our binary classifier performed at an excellent accuracy well above the industry low. Our multi-class classifier fell short.


Map each explanation weight to the corresponding superpixel
Map each explanation weight to the corresponding superpixel
Figure 1. MRI scans of a tumor-containing (top left) and healthy (bottom left) brain and the LIME images from their classification by VGG-16 (to their right). Both classifications were true. Note the clear boundaries of the tumor present in the true positive classification's LIME (top right).

4. Results

Upon testing the completed models, we found that our classifiers performed at a satisfactory rate, though there were considerable shortcomings in accuracy as well as in interpretability. The binary classifier (VGG-16) performed at an excellent accuracy of 90% on the testing dataset retrieved from Kaggle (Table 1). This marks a notable improvement from industry lows in diagnostic accuracy, which in certain cases such as pediatric, mixed-neuronal glial and histiocytic tumors, are as low as 63.33% (Table 1) [3]. However, it is important to note that testing for our binary model was on a limited dataset of MRI scans that did not contain many classes of tumors encountered by practicing clinicians. Acquisition of a more comprehensive dataset containing these more elusive tumors could be a productive next step.

Furthermore, we found reassuring evidence against any confounding in the binary classifier's diagnoses. As mentioned earlier, we implemented two modes of interpretability to assess this: LIME and saliency maps. In the case of the binary classifier, the former suggests the models operates on reliable, well-grounded evidence (fig. 1). The top two images in figure one are representative of the MRI and LIME image of most positive diagnoses by our binary classifier. Inspecting them, it is clear that the binary classifier is indeed making diagnoses based on the tumor, and not on an irrelevant, confounding feature common in tumor-containing MRI's. This result is important because it confirms that its assessment is grounded in the actual subject of diagnosis and can thus respond to novel tumors independent of other features in the MRI; this makes it a more confident option to consult in a clinical setting.

However, the second of the two modes of interpretability, saliency maps, suggests that the model may not be as tumor-focused as described. In contrast to the LIME images, the saliency map of the true positive diagnosis does not indicate any sort of focus on the tumor, or any particular feature, at all (fig. 2). Such a lack of focus can be expected of true negative diagnoses, but it is concerning to see in true positives because it conceals the reasoning behind classification, reducing our confidence in the model. It is interesting to note that despite this, the accuracy of binary classification was still very high as mentioned. This anomaly suggests that our model may be advanced to the point that it recognizes and makes judgements based off of features that humans would typically disregard; this is certainly a point deserving of further investigation as it could uncover powerful tools for diagnoses.

As mentioned in our introduction, we had intended to extend our work towards multi-class classification on a larger dataset. This dataset included images of the brain with glioma, meningioma, and pituitary tumors, or no tumor. Each of these sections contained significantly more scene data with which our CNN could train, and test its model efficacy. Unfortunately however, we were not able to get this done within the time constraint for a couple of reasons. First off, we wanted to ensure that the smaller-scale of our project worked, with both the LIME and saliency map visualizers. In the meantime, noting that the deadline was quickly approaching, we settled with utilizing the weights of a previous design with VGG16 model architecture, and simply mutating a trainable head. The largest issue that was unfortunately unreconcilable so late in the project process was the proper use of git's large file storage system. For some reason, the weights were not being properly tracked, and despite a lot of research on the internet we could not find the proper way in which to push changes to github. Ultimately, we reasoned that we could deal with those issues eventually so long as we are able to test our work at all on colab, so unlike our streamlined process from homework 5, we manually uploaded our project folder to colab so that we could at least test the multi-class model in the limited remaining time. However new

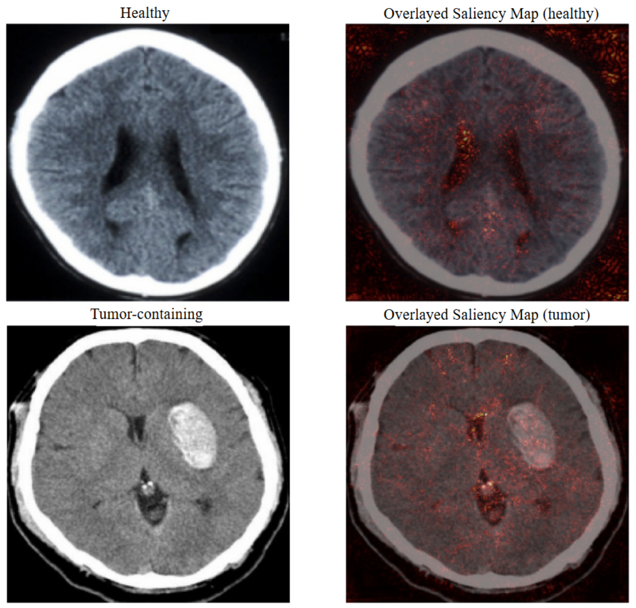

Healthy
Overlayed Saliency Map (healthy)
Tumor-containing
Overlayed Saliency Map (tumor)

Figure 2. MRI scans of a healthy (top left) and tumor-containing (bottom left) brain. The saliency maps from their binary classification overlayed on top is shown to their right. Both classifications were true. Note the the random distribution of saliency across the true positive diagnosis (bottom right).

problems arose when our testing performance was extremely low and stagnant, suggesting some significant underfitting. At this point we were not sure if this was the result of an overly simplified model, or something more severe within the underlying code (that may have been a result of the divergent branches by way of some simultaneous pushing/pulling from github), and we concluded that because any more effort in this avenue might not be so meaningful because we were unlikely to finish at that point, we settled with ensuring that our visualization worked as desired.

4.1. Technical Discussion

In implementing the various classifiers and features of our project, we had to make a number of design decisions which affected the quality and performance of our final product.

One, we found that implementing both binary and multi-class classification for our project brought challenges that raised the question of balancing complexity and feasibility. Indeed, the most significant challenge–and failure–we faced in this project is the multi-class classification. The difficulty of creating a sufficiently accurate multi-class model made the choice of settling for the binary classifier with superior accuracy a desirable choice. We saw that pursuing a more complex model capable of more specific classifications comes at the cost of labor as well as accuracy.

Furthermore, the challenge led us to explore other approaches to what is otherwise a procedural method of simply selecting an architecture and tweaking parameters. Although we ultimately settled with the sub-optimal VGG-16 architecture for the multi-class classifier out of necessity, we suggest the exploration of other architectures as potential avenues for future improvement.

Second, the decision to implement multiple modes of interpretability shed light on the complexity of interpretability, which further enforces its importance in machine learning models. Specifically, the apparent disagreement between the saliency maps and LIME images suggests that interpretability may not be a simple, binary characteristic of a classifier. Rather, it is a measureable quantity which, like accuracy, likely comes at a cost. Indeed, we found significantly poorer interpretability with our multi-class classification model; again, we see that complexity of classification comes at the cost of other important features. It continues to probe at the question of which features should be prioritized. This is question is as social as it is technical. Indeed, in developing or updating any given classifier, changes are often shifts in priority, not necessarily strict improvements.

5. Conclusion

Ultimately, we implemented a resource that can be utilized to properly diagnose a patient. Its impact is extensive, but among them the most significant is certainly its potential to save human life. There are certainly several models out there that have made contributions to solving this worldly plight, our project imparts a unique underlying impact that cannot be denied. Most significantly, this project provides preventative measures within the early stages of a treatment (via diagnosis/prognosis). Proper classification of the tumor would allow the doctor to assign the corresponding treatment plan, which only serves to better the quality of life of the patient, as well as their individual chance of surviving a potentially life threatening condition. Beyond the more compelling social implications of this implementation, tumor detection software may also provide a cost benefit to hospitals, which can potentially extend to patients by way of quality of care, direct payment plans, etc. There remains to be seen a ’perfect’ model for this issue in modernity, which is why so many programmers continue to add to the plethora of preexisting designs involving this specific subset of scene recognition. As such, we believe that we have contributed to the solution of this looming health threat, and have a model that can only continue to improve with further investment.

References

[1] Akmalbek Bobomirzaevich Abdusalomov, Mukhriddin Mukhiddinov, and Taeg Keun Whangbo. Brain tumor detection based on deep learning approaches and magnetic resonance imaging. *National Library of Medicine*, 2024. Accessed: 2024-12-17. 1

[2] Raul Csimpeanu. Vgg16 binary classification, 2024. Accessed: 2024-12-17. 1

[3] Sidpra J Mankad K. Dixon L, Jandu GK. Diagnostic accuracy of qualitative mri in 550 paediatric brain tumours: evaluating current practice in the computational era. *National Library of Medicine*, 2022. Accessed: 2024-12-17. 3

[4] Adrian Rosebrock. Opencv saliency detection. *Py Image Search*, 2018. Accessed: 2024-12-17. 2

[5] Fathy Sahlool. Brain tumors classification cnn, 2024. Accessed: 2024-12-17. 2

Appendix

Powerpoint Presentation

Slides

Team contributions

John I implemented the Binary Classification(classifying whether or not an image has a tumor) with 90% accuracy(using VGG 16). This involved making the github repository, downloading the dataset from Kaggle, reproducing the Homework 5 code, changing up parameters to accommodate for binary classification, and fine-tuning the head to produce the most desirable results. I also created a few LIME images to put in this report and our slides using the model's best checkpoint.

Brian I was responsible for research in the project's early stages, finding the sources and outlining the ideas that we have ultimately decided to finish the project on. Furthermore, I implemented LIME visualizations for our classifier and used them to guide improvements (parameter tweaks and changes to architecture) on the binary classifier, which ultimately pushed its accuracy to its current state. I was also the one responsible for the production of the slide deck presentation and drafting the final project report.

Shafiul I focused on implementing custom saliency maps to highlight the important features that the model was using to make predictions, particularly in distinguishing between tumor and non-tumor cases. I fine-tuned the parameters of the saliency map visualizer to ensure that it accurately reflected the regions contributing most to the model's decision. I also tested LIME on various images to validate its effectiveness in providing interpretable explanations for the model's predictions, ensuring that it accurately represented the key features considered by the model in its decision-making process.

Nahum My main focus for this project was to attempt the implementation of the multi-class classification portion of this project. As evidenced by the report, this section of the project ultimately fell through as I realized quickly that there were a couple of conflicting issues that I simply could not reconcile (see results section for further details). Although this segment of the project unfortunately did not pan out as we had hoped, I was in charge of setting up the colab workspaces, attempting to resolve the git lfs issues, merging divergent branches via github, and all of the individual aspects of the multi-class implementation (model architecture, retrieving online datasets and weights as needed, testing and debugging, etc.).