Brain Cancer Detection and Classification Report

Alex Do, Alice Nguyen, Amber Zhang, Raquib Alam, Vedant Gopal
ECS 171: Machine Learning
Professor Setareh Rafatirad
Spring 2025

I. Introduction and Problem

Brain cancer is one of the deadliest and most life-threatening forms of cancer. Brain cancer happens when irregular cells are formed or found in the brain that may lead to the development of two possible types of tumors: malignant (cancerous) or benign (non-cancerous) (Mustafa et al., 2018). Today, one significant challenge is the lack of specific symptoms in initial stages to be able to detect brain tumors early (Mustafa et al., 2018). Most of the time, high quality scans are obtained using Magnetic Resonance Imaging (MRI) and Computer Tomography (CT). These are widely used tools for diagnostic purposes. However, the analysis of the brain scans depends on the interpretation of radiologists and their opinions. Since human analysis is required to analyze the scans, the results may be biased or inaccurate due to possible issues in timing, accuracy, and natural human error. As a result, brain cancer is often detected and diagnosed after symptoms become severe and apparent which limits treatment options. If treatment options are limited, there may be a reduced chance of survival and recovery.

To address this issue, we proposed the development of a machine learning model to accurately classify brain MRI scans as either cancerous or non-cancerous. The objective is to support and enhance the early diagnosis of brain tumors by automating the image classification process. In turn, reliance on manual interpretation can be reduced which would minimize human error and increase efficiency of clinical processes and workflows. This approach has potential to streamline diagnostic processes, enable faster clinical decision-making for treatment plants, and improve patient outcomes.

The scope of this project involves building a high accuracy classification model that is capable of both detecting the presence of a brain tumor in general, but also classify the type. Our development process consists of key steps including: dataset selection, data preprocessing, exploratory data analysis, model development, model evaluation, and the creation of an interactive front-end. To build the model, a convolutional neural network will be used based on choosing features and hyperparameters to increase accuracy. After evaluating potential datasets, the Brain Tumor MRI Dataset, created by Masoud Nickparvar, was selected. This dataset contains over 7,000 images of human brain MRI categorized into four classes of brain cancer: glioma, meningioma, pituitary, and no tumor present. By utilizing a large, labeled dataset and applying fine-tuning techniques we learned in lecture, we aim to maximize the accuracy of the model, so it can be a reliable tool.

While our project design has the potential to make a direct impact, potential limitations must first be acknowledged. One key challenge, that can be addressed by preprocessing, is variability in the images of the dataset. More specifically, there can be variability in image dimensions, orientation, and resolution. Additionally, although the dataset includes a diverse set of brain MRI images across multiple tumor types, no context is provided on them such as patient age, gender, or medical history. The absence of this context information and clinical data may limit the model's ability to generalize in the future and may reduce its effectiveness when applied to broader or more specific populations.

To provide additional context on the classes of the brain cancer dataset mentioned above, information will be given based on order from most common to least common. First, gliomas are the most common form of brain tumor which originate from the non-neuronal cells in the brain and spinal cord (Mustafa et al., 2018). Next, the second most common type of brain cancer is meningioma which develops in the membrane that covers the brain and spinal cord. Although meningiomas are often benign, non-cancerous, they still have the potential to cause significant health issues. The third most common tumor in the dataset are pituitary tumors which are formed in the pituitary gland. This is well known and is responsible for hormone regulation in the body (Mustafa et al., 2018). The diverse dataset we chose enables our model to be trained on varying data of different forms of brain cancer and will then be accurate in distinguishing between brain cancer types and non-cancerous cases. This illustrates potential for making a real-world impact as decisions and treatment plans can then be formed on a faster timeline and before symptoms worsen beyond repair.

II. Literature Review

Machine learning in real-world situations such as medical imaging has shown considerable progress in recent years, especially in the area of brain cancer detection through MRI scans. Brain cancer typically consists of pituitary, glioma, and meningioma tumors which are dangerous due to their volatility and location. As one of the leading causes of death globally, early and accurate detection is critical to save lives. The research conducted on academic papers regarding this subject have focused on using conventional neural networks (CNN) to assist with careful examination of brain cancer detection. The relevant sources used for this report are primarily taken from Google Scholar.

The majority of studies focused on three major types of tumors–pituitary, glioma, and meningioma tumors–each of which can be classified as either malignant or benign. Each type of tumor possesses different cell types, location, growth pattern, and molecular characteristics. Pituitary tumors originate from the pituitary gland and can result in hormone imbalances for the given individual, glioma tumors develop from glial cells, and meningioma tumors grow on the meninges but may cause neurological symptoms depending on various features. Research consistently shows that tumors vary in color, shape, and location, which are three features we have chosen to focus on in this report. Previous studies involving CNNs have shown promising results with accuracies estimated at around 90% in most datasets (Saba et al., 2020). However, generality remains a key issue as CNNs cannot work the same for every dataset, making it a challenge for researchers.

Current machine learning approaches to brain cancer detection typically involve CNNs for the purpose of image classification and transfer learning (Saba et al., 2020). Approaches even include fusing CNNs with machine learning techniques such as Random Forest, Support Vector Machines (SVMs), and Self-Organizing Maps (SOMs) (Saba et al., 2020). This combination of methods are gaining traction for research in this field as CNNs focus on feature extraction from image and video data, while machine learning methods are better suited for handling structured, numerical data.

While the methods used above are generally accurate, there are certain limitations that must be considered. “Accuracy for each cancer category is far from maturity. Most researchers either did not employ benchmark datasets or used small sets for testing of their proposed techniques” (Saba et al., 2020). This results in a clear imbalance in data, as various types of tumors are misrepresented. Many studies currently deployed use smaller datasets with fewer images, limiting the overall reliability of the model. Moreover, many models lack a clear testing plan as they are not testing in various setting,s furthermore restricting their real-world applicability.

Our work attempts to build on existing research by using a CNN configuration trained on a larger, pre-processed MRI dataset acquired from Kaggle, which includes thousands of images. The overall objective of this research is to find solutions to the challenges listed above, such as a lack of generalization and variability. To accomplish this feat, we aimed to extract and prioritize features based on color contrast, shape, and location to eliminate unwanted features and irrelevant data. “We do not use all the features to train a model” (Chen, Dewi, Huang et al., 2020). Additionally, feature selection helps focus on the features we want to investigate further and adds a new dimension of specificity, furthermore increasing classification accuracy and performance of the model. We can also address generalization issues by regularization techniques and model classification as it can simplify the model, decrease the training time, reduce overfilling, and avoid dimensionality (Chen, Dewi, Huang et al., 2020).

Our model not only focuses on classifying tumors based on various features, but also aims to improve on the accuracy of previous models by using a larger dataset with numerous classification techniques. Although models may vary in accuracy, with certain models performing better than others and vice versa, “A model is correct if it satisfies its purpose no less and no more” (Leitch, R.R., et al., 1999). The primary goal would be to guarantee the model is used appropriately and serves its purpose.

In conclusion, the literature pinpoints specific points of improvement and supports our research for a better model representation. Initial work has made substantial progress in the field of brain cancer detection, but obstacles such as generalization, interpretability, and variability still persist. The work included in our report aims to build on these strengths, but also address these obstacles by targeted feature selection, incorporating a larger dataset, and reducing model complexity–ultimately aiming to improve metric performance in accuracy, precision, and recall for machine learning models in brain cancer diagnosis (Chen, Dewi, Huang et al., 2020).

III. Dataset Description and Exploratory Data Analysis (EDA)

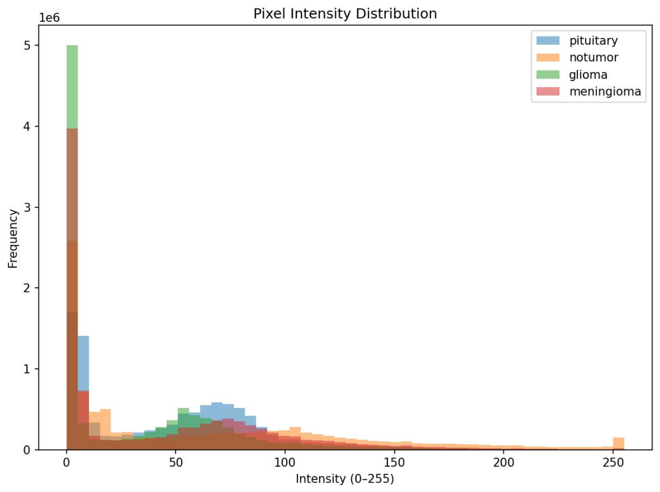

Fig. 1. Pixel Intensity Distribution

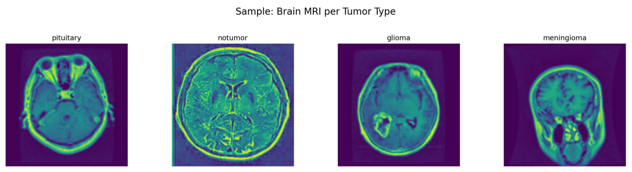

Fig. 2. Images of each class (pituitary, nontumor, glioma, meningioma)

The dataset used in this study is the Brain Tumor MRI Dataset (Nickparvar, 2021) that was acquired from Kaggle. As mentioned in the introduction, this dataset contains over 7,000 images of brain MRIs that are categorized into four distinct classes of brain cancer: glioma, meningioma, pituitary, and no tumor, with training and testing splits for each tumor class. Sample images of each class are provided in Figure 2, These are among the most common forms of brain tumors, and additional background or information can be found in the introduction. The images in this dataset were collected from three different datasets to ensure that the data is as diverse and abundant as possible. There are many potential applications for this dataset, but our main objective is to be able to classify brain cancer in various forms for early diagnosing and to furthermore create treatment plans for patients. The diverse dataset with numerous images of each type of class will allow for better training, testing, and accuracy for the model after analysis and preprocessing.

To better understand our datasets, we conducted Exploratory Data Analysis (EDA) using Python libraries including os, PIL, and matplotlib. We first wanted to ensure that the number of samples per class was balanced. Observing this allowed us to analyze any imbalances or skewness in the dataset that could potentially lead to problems in model development and training. Then, we visualized representative images from each category to assess image quality, tumor visibility, and variation in tumor structure. To further investigate our dataset, we wanted to examine the pixel characteristics and the contrast. We analyzed pixel intensity distributions, as seen in Figure 1, across all classes using the blue channel of 200 randomly sampled images per class. We found that there were notable differences in brightness and contrast. This analysis confirmed that the class distribution was approximately balanced and helped inform our preprocessing strategy, especially our decision to apply the Contrast Limited Adaptive Histogram Equalization from OpenCV.

Since the images in this data are of varying sizes, preprocessing was the first step in cleaning the dataset. This involved resizing the images to standardize the dimensions for better training with our model. For our image preprocessing, we utilized the OpenCV library to help with standardization and enhancement on our dataset. This helped us ensure that our images are consistent throughout to help reduce any variability in our model training. We then resize the images to a consistent resolution of 256 x 256 pixels and convert them to grayscale. As mentioned above, we used the Contrast Limited Adaptive Histogram Equalization (CLAHE) technique from OpenCV in the preprocessing section. This allows us to avoid noise, and increase the contrast in areas that may be harder to see. We also incorporated Gaussian blurring to help reduce any noise and sharpen any critical features. We then enhance the local contrast using Contrast Limited Adaptive Histogram Equalization as mentioned earlier. Images were then normalized to a [0, 1] range and saved to use in our model training.

In conclusion, these preprocessing steps and techniques ensure consistency across the dataset and improve CNN's ability to accurately detect tumors based on localized spatial and intensity features.

IV. Brain Tumor Detection and Classification Methodology

A. *Overview of the Methodology:*

First, we present an overview of our methodology. Utilizing over 7,000 different MRI samples of various types of brain cancer we utilized a CNN-based deep learning algorithmic approach to detect and classify brain tumors using MRI images. In short, our methodology consists of four major stages: data collection, image preprocessing and normalization, feature extraction (color contrast, location, and shape), and model training and evaluation. The goal is to be able to utilize visual features to help capture the spatial and structural characteristics of the tumors shown in the MRI images. The features are then used to train a CNN that will determine each MRI into one of four categories: glioma, meningioma, pituitary tumor, or healthy brain.

B. *Feature Extraction for Classification:*

Our data pre-processing steps helped to guarantee consistency and prepare our dataset for feature selection. During feature extraction, we focused on three aspects to generalize our model: color contrast, location of the tumor, and shape/morphology.

Color Contrast: Color Contrast was selected as a feature due to its ability to capture visual irregularities in the brain within brain imaging scans. Typically, in MRI or CT scans, tumors will exhibit intensity values that are either significantly higher or lower than those of the surrounding healthy tissue. Furthermore, color contrast enhances the edges and boundaries between normal and abnormal tissue in the brain, aiding in differentiation between tumors and healthy tissue.

Location of Tumor: Brain tumors, such as gliomas, meningiomas, and pituitary tumors, display preferences for specific brain regions. Incorporating location-based information assists us in constraining the model's focus to relevant areas of the brain, or regions of interest, to detect possible tumors. Moreover, location helps us in the diagnosis and classification of tumor types.

Shape and Morphology: Shape and Morphology serve as critical features in the characterization of brain tumors. Once we have located a tumor, we use its computed area in pixels and approximate the tumor volume within a 2D slice. The perimeter of the tumor is calculated as the length that it bounds. The solidity of the tumor, or the convex hull area, offers insight into the irregularity of the tumor's shape. The lower this number is, the more irregular the shape of the tumor, and the more severe the tumor likely is. Assessing tumor size also assists in estimating the cancer stage, which is essential for prognosis and treatment planning.

These features listed above were then used to train a Convolutional Neural Network (CNN). We then trained our model on a dataset. Throughout the training process, we kept track of specific model metrics such as accuracy, precision, recall, and how well the model distinguished between glioma, pituitary, meningioma, and healthy brain scans.

C. *CNN-Based Model Architecture and Training:*

To classify the brain MRI images into the four different categories (glioma, meningioma, pituitary tumor, and healthy), we design a hybrid model that combines CNN with the hand-crafted features representing the tumor shape, color contrast, and location. The hybrid architecture allows the model to learn both high level spatial features from raw images and domain-specific characteristics.

The handcrafted features include: the area of the largest contour detected via edge detection, representing tumor size, the mean pixel intensity of the equalized grayscale image, representing color contrast, and the normalized (x, y) coordinates of the tumor's center, representing spatial location. These features were extracted from each input image and standardized before training.

The CNN architecture consists of two convolutional blocks where they each contain a Conv2D layer. First with 32 filters and the second with 64 filters. It is then followed by a max pooling layer to reduce spatial dimensions and a dropout layer to mitigate overfitting. The output of the CNN is flattened and concatenated with the handcrafted feature vector. This combined representation is then passed through a fully connected dense layer with 128 neurons, followed by a softmax output layer for multi-class classification.

We trained the model using 85% of the dataset for training and 15% for validation. The model was compiled with the Adam optimizer and categorical cross-entropy as the loss function. Early stopping was applied with a patience of three epochs based on validation loss. Training was conducted for 10 epochs with a batch size of 32. The model was subsequently evaluated on a separate test set, achieving a test accuracy of about 93%, demonstrating its effectiveness in classifying different brain tumor types using both visual and structural cues.

D. Training Configuration and Hyperparameter Tuning

After our initial training phase where we were able to establish a baseline performance of our model, we began to tune key hyperparameters. Hyperparameters are variables that determine the structure of a Deep Neural Network (DNN). These variables are valuable in a model's performance as it can help increase the accuracy and control of our network. To optimize which hyperparameters to use in our model, we applied manual hyperparameter tuning. We manually tested different hyperparameters to find the most optimal configuration for our model. The hyperparameters we focused on during model training included batch size, dropout rate, number of filters, dense layer size, learning rate, number of epochs, and early stopping patience.

Batch Size: This hyperparameter is the number of samples processed before the model updates its weight. A smaller batch size typically leads to a more stable learning but a slower training. On the other hand, a larger batch size could increase the speed of training but risk having poorer generalization. For our model, we observed that when we manually tuned the batch size, decreasing it led to more stable training but it took a significant amount of time. When we compared this with increasing the batch size, we noticed that the time sped up but validation accuracy was slightly reduced due to less frequent updates. We found that for our model's best performance, a batch size of 32 allowed the model to train efficiently while maintaining stable learning.

Dropout Rate: The dropout rate is the fraction of neurons that are randomly "dropped" during training to prevent overfitting. When a model has a higher dropout rate, this typically adds more regularization, however, too much can lead to underfitting. Too small of a dropout rate can result in overfitting. We found that having a dropout rate of 0.5 performed the best. It was the perfect value such that it wasn't too low to lead to overfitting and wasn't too high to reduce training accuracy.

Number of Filters: The number of filters in each convolutional layer helps determine how many feature maps are learned. Having more filters allows the model to capture a more complex pattern, but increases the computational cost and risk of overfitting. When we increased the number of filters from 16 to 32, we saw that this improved our early performance; however, after over 64 filters, we began to observe overfitting. Overall, for best performance, we kept the number of filters between 32 and 64 in the convolutional layers.

Dense Layer Size: This is the number of neurons in the fully connected dense layer before the output. Having a larger dense layer increases the learning capacity but could cause overfitting. Smaller dense layers may reduce the model complexity and training time, helping to prevent overfitting, but a value too small could underfit the data and fail to capture important patterns. We found that a dense layer of 128 neurons perfectly balances performance and generalization.

Learning Rate: This hyperparameter is important as it controls how much the weight is updated during training. Having too high of a learning rate can lead to unstable training and divergence, while a value too small can result in slow learning or getting stuck at the local minima. We utilized the Adam optimizer, which stands for adaptive moment estimation. It is a full optimization algorithm that uses learning rate internally and defaults the value at 0.001. When using the Adam optimizer, we notice a stable and effective learning throughout our training process.

Number of Epochs: This hyperparameter is the number of times the model sees the entire training dataset. Having more epochs allows the model to learn more but could potentially overfit to where our model performs well on training data but poorly on unseen data. Having too few epochs may result in underfitting where it has not learned enough to generalize effectively. For our model, we observed that setting our epoch to 10 was the perfect balance.

Early Stopping Patience: The early stopping patience hyperparameter stops the training when loss has been minimized. This is to prevent overfitting as well as unnecessarily continue training. At the end of every epoch, the validation loss is monitored. For our model, we set our "patience" to be three, meaning that if the validation loss has not improved over three consecutive epochs, training will terminate early. Three seemed to be the most optimal patience level for our model, as it wouldn't prematurely stop training, nor would it continue training past the necessary point and risk overfitting. Additionally, once training has terminated, the model will restore the weights to the epoch that had the smallest validation loss.

Overall, hyperparameter tuning is important when developing and training our model. It allows us to optimize our model's performance, and through manual tuning, we were able to find the most optimized hyperparameter values. With a batch size of 32, a dropout rate of 0.5, 32 and 64 filters in the convolutional layers, 128 neurons for the dense layer size , a learning rate of 0.001, training for up to 10 epochs, and applying early stopping with a patience of 3, this configuration help provide a strong balance between learning capacity and generalization while achieving high accuracy.

E. User Interface and Front-End Deployment

Our simple web application's frontend is built on React to match the style and formatting of our Figma mockups alongside having it hosted on the localhost with the Flask backend. When the user navigates to http://localhost:3000, they will see a clean 1280X1028 px screen with a constantly moving gradient background from pink to white representing the frontend of our project. The frontend is broken up to three parts in with we have the title of our website ”Brain Cancer Detection Website” on the left side header and on the right side header we have ”ECS 171 Project” Right below this is the InfoCard component that we made that represents a little bit about our project, where to find the paper and who we are. Alongside having an InfoCard component, we also made sure to have a TeamMember card component in which it represents the team that we had to build out the website. Now, within the InfoCard that represents the purpose of the website, we created an upload icon that will redirect you to your file system in which the user can upload a picture in the restricted formats of a .jpg, .jpeg, or a .png. When clicking on the upload button, it will trigger a HTML tag of `<input type="file">` and once the user has selected their desired file, the frontend will send a fetch POST request to the Flask /predict endpoint in our backend. This allows it so that the image that is being sent to our backend is being sent in as FormData. We also integrated CORS in which this allows Flask to communicate between ports 3000 (the React frontend) and port 5001 (the flask server backend).

After the backend runs the given picture by the user into our model.py and app.py backend, the backend will output the tumor type of which the image was predicted to be classified in. The frontend will display an ”Output” button and when this button is clicked, it will show an OutputCard that shows the user's uploaded image and what the model predicted the tumor type to be. For the OutputCard we also created this as a separate component in our frontend so that depending on which type of tumor the image is classified in, it will be able to give the right and accurate information back to the user. This Output card is also designed in figma and the styling was made in a way where the output was cleanly formatted and presentable to the user using the same fonts as the entirety of the website - sticking to one coherent theme.

When running this on your localhost, the user would have to run npm install within the frontend folder to make sure that all of the necessary packages are downloaded. Then, when in the frontend folder directory you will want to run npm start to make sure that the server starts on your localhost port 3000. Furthermore, you will also want to do this for the backend as well, as this is the only way the model will run and give a predicted output back to the user. For this, you will have to make sure that you are in the python backend folder and run python app.py. When this command is run it will start up the server on port 5001 and this will be connected with the frontend. When these are both running, you will have the localhost website up, running and working smoothly.

V. Experimental Results and Evaluation

This section will discuss the results that we obtained through our training and testing. We generated various graphs that represent different comparisons and metrics to gain a comprehensive understanding of our model's strengths and weaknesses. These graphs include: Accuracy of Training vs Validation Data, Loss of Training vs Validation Data, Confusion Matrix, Heatmap of Tumor Centroids, Distribution of Tumor Color Contrast, and Tumor Shape Area Distribution.

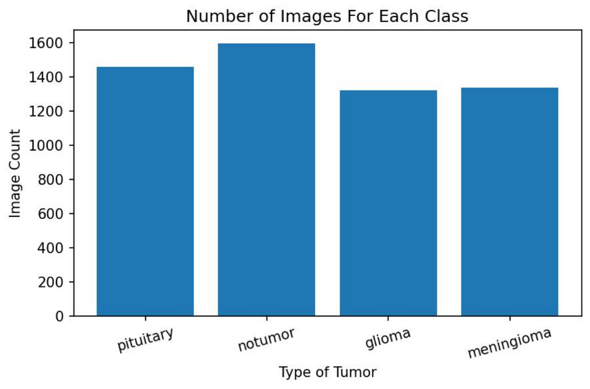

Fig. 3. Images of each class (pituitary, nontumor, glioma, meningioma)

Figure 3 simply indicates the number of total images contained in each class.

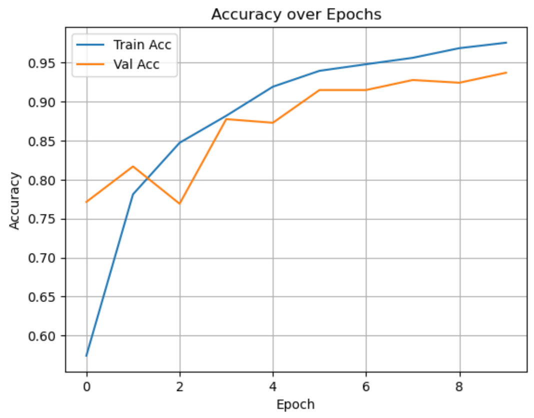

Fig. 4. Change in accuracy over epochs

Figure 4 illustrates the evolution of training and validation accuracy across successive epochs. Validation accuracy is computed by comparing the results obtained by running the model on the validation dataset with the ground truth. Likewise, the training accuracy is determined by comparing the results from the training dataset with the ground truth. Over time, we can see how the training accuracy consistently exhibits an upwards trajectory while validation accuracy initially oscillates before gradually converging at a similar level to the training accuracy. Additionally, we can deduce that this model is not overfitted because training and validation accuracy stabilize at a similar point. In contrast, if training accuracy continued to increase while validation accuracy suddenly dropped, we would need to investigate for potential overfitting.

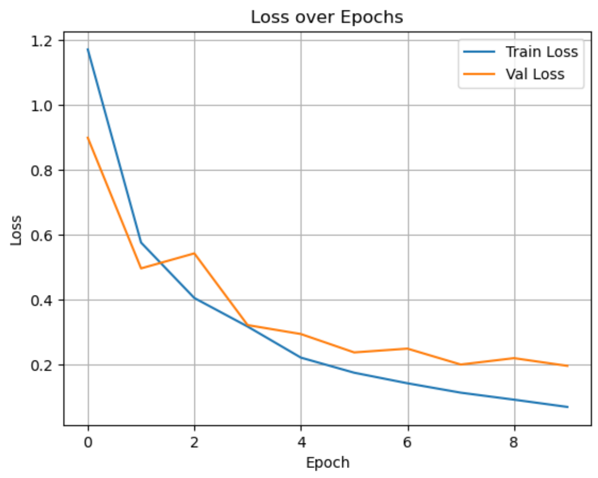

Fig. 5. Change in loss over epochs

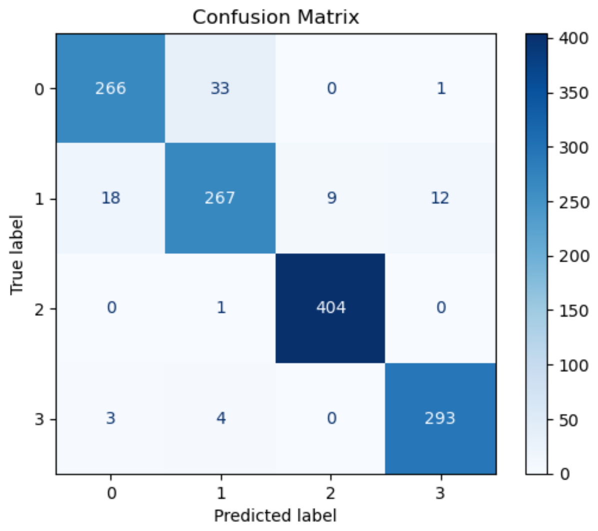

Fig. 6. Confusion matrix comparing predicted and truth labels

Figure 5 illustrates the evolution of training and validation loss across successive epochs. Validation loss is the error on the validation data, while training loss is the error on the training data. Calculation for loss is done by averaging the loss across all samples in a batch and then averaging all the batches to obtain the loss within the epoch. As we can see in the results, the training loss decreases over time, while the validation loss fluctuates during the beginning epochs but eventually levels off near the training loss line plot. This is the ideal outcome. Additionally, we can tell by looking at the graphs that we are not overfitting as both lines level off close to each other; if training loss continued decreasing while validation loss suddenly increased, we may have overfitted.

Figure 6 illustrates the Confusion Matrix for our model. It compares the number of instances where the model predicted the correct output (predicted label) compared to the ground truth (true label), and the instances where it made incorrect predictions. Each of the four classes in our dataset, glioma, meningioma, non-tumor, and pituitary, is assigned a code (0, 1, 2, and 3, respectively). The main diagonal displays the number of correct predictions. The surrounding cells within each true label's row represent the number of instances it was incorrectly classified as the class represented in the x-axis. For example, cell (1,1) represents the number of meningioma tumors that were correctly classified as meningioma tumors. Cell (0,1), Cell (2, 1), and Cell (3,1) represent the amount of meningioma tumors incorrectly classified as glioma, non-tumor, and pituitary tumors, respectively. As we can see, the model did the best job of correctly predicting non-tumors, with only one incorrect classification as a meningioma tumor. Using this matrix, we can determine where our model may need improvement, such as how it had a high number of occurrences classifying true glioma tumors as meningioma tumors. We can also observe that all other classes are most frequently incorrectly classified as meningioma tumors, indicating that we may need to improve training in that area.

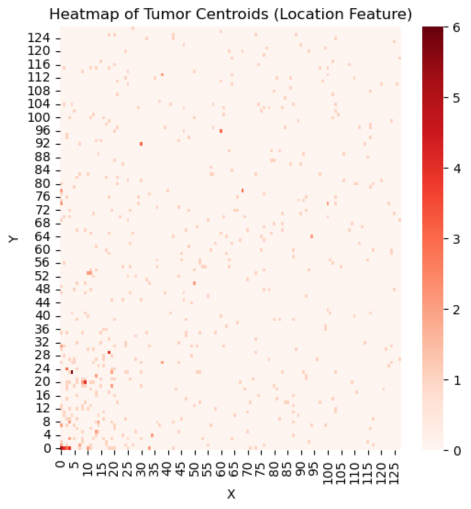

Fig. 7. Heatmap depicting tumor centroids

Figure 7 illustrates a heat map that indicates the spatial distribution of tumors that were in our training dataset. Regions shaded in darker red correspond to areas with a higher frequency of tumor occurrences, while lighter shades indicate lower incidence. As we can observe from the figure, the tumors are concentrated near the origin, where the red is more dense and intense.

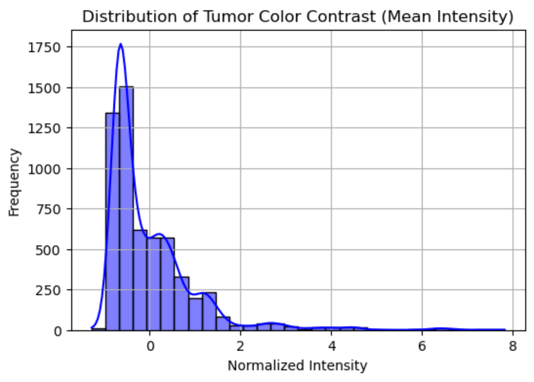

Fig. 8. Distribution of tumor color contrast by normalized intensity and frequency

Figure 8 illustrates the mean intensity of tumor color contrast. The closer to zero the normalized intensity is, the darker the tumor's color is. As the value increases, the tumor gets lighter in color. As we can observe in the figure, the distribution of the tumors we evaluated is skewed towards the left, indicating that more tumors are darker in color rather than lighter.

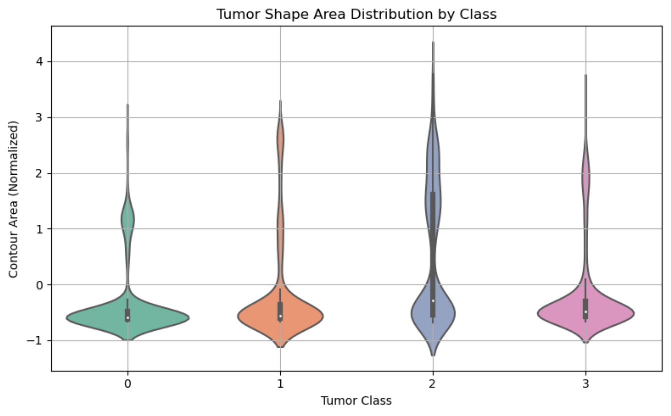

Fig. 9. Tumor Shape Area Distribution by Class by Tumor Class and Normalized Contour Area

Figure 9 is a violin plot that shows the density and range of tumors within a shape area. Specifically, the width of the "violin" plot at any Y-axis value shows the density of tumors within that shape area, while the height represents the range of shape areas within the class. These 4 (0, 1, 2, 3) classes correspond to glioma, meningioma, non-tumor, and pituitary, respectively. This graph helped us to determine if certain classes may have larger or smaller contour areas and whether classes are more consistent in shape.

VI. Conclusion and Discussion

Today, brain cancer remains a critical issue in the medical field, with existing early detection methods lacking reliability and precision. Our study on brain cancer detection aims to address these challenges and more by ultimately developing a more efficient and accurate model capable of brain tumor classification. To achieve this, we built upon former research conducted by experts listed in our references and supplemented their findings with our personal research and key observations, focusing on specific obstacles that can affect detection accuracy and model generalization. Due to the research we conducted, we found out that we can use a larger dataset for generalizability and filtering techniques such as feature selection to fix the shortcomings of past research.

Through the use of a Conventional Neural Network (CNN), we were able to successfully train a model using a large dataset acquired from Kaggle. By engaging in feature selection, emphasizing most notably accuracy and generalizability, we significantly improved our model's classification. By including features such as tumor color contrast, location of tumor, and morphological shape of the tumor, we were able to achieve higher accuracy for detecting brain cancer and furthermore distinguish between the different brain tumor types.

Despite the impressive performance of our model, especially in terms of generalization and accuracy, certain limitations still remain. Namely, our model is tailor-made for MRI scans, but cannot generalize to other imaging techniques such as Computed Tomography (CT) scans, Positron Emission Tomography (PET) scans, and even X-rays. Additionally, model performance may vary based on image positioning and angling. While these limitations highlight certain areas for model improvement in order to construct a more refined model, our findings can prove to be a real step forward in terms of brain cancer detection and classification, while also being a valuable asset to use in medical settings.

VII. Github Link, Demo Video, Project Roadmap, and Member Contributions

A. Github Link: https://github.com/alicetnguyen/ECS171_BrainCancerDetection

B. Demo Video Link: https://www.youtube.com/watch?v=KZQKmXKMf9s

C. Project Roadmap:

Week 4: At the start of week 4, we were able to form our team and also come up with the topic that we finally decided on. We brainstormed, gathered lots of information on what we thought would best fit the project goals and outline and we found a complex problem that we all deeply wanted to solve. We all had an interest in biology and prevention of diseases. As a team, we went onto Kaggle where there are many datasets on various diseases and decided that brain cancer was the right topic for this project. With having so much training and testing data and images, we were confident enough to make a strong model that could help predict brain cancer based on CT scan images.

Week 5: Our group had our first in-person meetup to discuss the final report and which group member would contribute to which section of the report. Using the mid-quarter progress report as a guideline, we started working on the code and developing our model.

Week 6: We stuck to having weekly in person meetings where we would give our insights and updates to our part of the project and what we were going to work on for the following week. By the end of week 6 we had a full report on the exploratory data analysis which gave us more insight into what the dataset has and entails.

Week 7: By the end of week 7 we had flushed out pre-processed data with what certain features we thought were the best for helping with detecting the different types of Brain Cancer. We decided to focus on color, location and shape.

Week 8: Starting week 8, based on our initial EDA and pre-processing, we started working on our CNN model which is the model we decided to use in the end. We used features such as shape/morphology, color contrast, and location of tumor for our model.

Week 9: We again met in-person a week earlier to finish the majority of the report. By now, we finished the EDA, data-preprocessing, and constructed our model. The major goal would be to finish the writing aspects of the report and adding onto the work we completed in the mid-quarter progress report.

Week 10: We meet up for the final time this week to record the demo video for our project and make sure that the frontend was working and connected to the model. We created a Figma mock up to help us organize and see which exact components we needed in order for the user to seamlessly use our website and so the user can easily follow and upload an MRI image to then see what type of brain tumor it is classified as. We used Flask for our backend and React for the frontend. We made sure that in our demo we highlighted how to clone the code and make sure that the user can run this on their localhost and start using our website.

D. Team:

Alice Nguyen: Team Leader
Alexander Do: Team Member
Amber Zhang: Team Member
Raquib Alam: Team Member
Vedant Gopal: Team Member

E. Roles Assigned and Member Contributions:

Alexander Do: Alex worked on the early plan for modeling development, the actual model development, as well as the methodology section of the report.

Alice Nguyen: As team leader, Alice was responsible for organization of team communication and group meetings, the planning of the road map for the project, and created our Github repository. She also did dataset research, data preprocessing, exploratory data analysis, the report, and demonstration video.

Amber Zhang: Amber worked on dataset research, feature selection research, model development, results evaluation, and demonstration video.

Raquib Alam: Raquib worked on dataset research, exploratory data analysis, feature selection research, the frontend of the interface, and demonstration video.

Vedant Gopal: Vedant focused on Literature review, model evaluation, conclusion, discussion, and demonstration video.

References

[1] C. Bettegowda, “Metastatic Brain Tumors,” Johns Hopkins Medicine, 2025. [Online]. Available: https://www.hopkinsmedicine.org/health/conditions-and-diseases/metastatic-brain-tumors

[2] R. C. Chen, C. Dewi, S. W. Huang, et al., “Selecting critical features for data classification based on machine learning methods,” *J Big Data*, vol. 7, no. 52, 2020. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00327-4

[3] R. R. Leitch, et al., “Choosing the Right Model,” *The Institution of Engineering and Technology*, Sep. 1999. [Online]. Available: https://digital-library.theiet.org/doi/epdf/10.1049/ip-cta%3A19990503

[4] T. Saba, “Recent advancement in cancer detection using machine learning: Systematic survey of decades, comparisons and challenges,” *ScienceDirect*, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1876034120305633

[5] M. Nickparvar, “Brain Tumor MRI Dataset,” Kaggle, 2021. [Online]. Available: https://doi.org/10.34740/KAGGLE/DSV/2645886

[6] M. Mustafa, A. F. Sali, E. M. Illzam, A. M. Sharifa, and M. K. Nang, “Brain cancer: Current concepts, diagnosis and prognosis,” *IOSR Journal of Dental and Medical Sciences (IOSR-JDMS)*, vol. 17, no. 3 (Ver. 8), pp. 41–46, 2018. [Online]. Available: https://www.iosrjournals.org