Mind the Gap: Bridging Accuracy and Efficiency in Tumor Detection via MRI Imaging and Deep Learning

Jonathan J. Nkangabwa

Northwest Missouri State University, Maryville MO 64468, USA
S556575@nwmissouri.edu and jona.nkangabwa@gmail.com

Abstract. **Keywords:** Tumor detection · MRI imaging · Neural network model · Deep learning · Data analytics

1 Introduction

This is the introduction to my Capstone Data Analytics project and here is a quick synopsis of the work done thus far:

1. I chose the domain of tumor detection through MRI imaging because it plays a crucial role in early diagnosis and prognosis. I also have a family member currently going through a very vicious form of cancer, and wanted this project to mean something more than just a grade/my portfolio.

2. I had thought of this project while Summer I session was winding down. I a large data set from Kaggle.com that seemed to contain the exact data I needed. I will source it down below.

https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection?select=pred

1.1 Goals of this Project

Model The first goal is to develop an efficient neural network model to accurately detect tumors.

Enhancement The second goal would be enhancing the models efficiency through optimization and evaluating the modifications and comparing them across the board with previous model versions.

Therefore, reporting the performance of the final model on a separate (unseen) test set, and ideally on external validation data-sets if applicable, is crucial for verifying the model generalizability.

Understanding Data The last and final goal would be conducting a deeper dive into the results and data with the appropriate knowledge experts (physicians, radiologists, MRI technicians)

My focus is specifically for the automated detection of tumors through MRI imaging because not only does it lead to early detection and action, but is extremely high-resolution and non-invasive in the sense of no radiation being given off. This in turn leads to better patient outcomes and happier families. Giving radiologists and up-and-coming physicians (like my younger sister, who I am extremely proud of) the tools to better detect and diagnose tumors and thus have one less thing they have on their busy plates.

1.2 Project Implementation

Data Collection and Pre-processing: Find quality MRI images and splitting into appropriate training, validation, and testing sets. In order to create a model

that detects tumors, I collected a combination of Kaggle data sets as folders (labeled glioma, meningioma, pituitary and no tumor) full of .jpg and .jpeg files and created a super data set alongside a validation set and a new untouched data set labeled as either yes or no to having a tumor (in order to test the model after the trained testing set)

The only data scraping techniques used were downloading the individual data sets from their respective websites and curating a new super data set for the model to learn from. The only data attributes being focused on are the images themselves and which contain tumors and don't. Before creating and training the model, I must first on normalization to all the images through resizing, normalization, and noise reduction to standardize the images for the model to accurately, efficiently and effectively process the data.

1.3 Data Cleaning:

In order to properly access the data sets, I had to organize, clean and store my data sets in an accessible manner to manipulate with Python. I collected the data set from a reliable source (Kaggle) and determine if I had a sufficient number of tumor and non-tumor images for balanced training and evaluation. I then took the files and placed them in one of 4 folders specified as cancerous or not.

This project required the use of image manipulation and loading, so I had to use several modules. One was the 'os' module, where I could properly navigate my machine and manipulate specific file paths. The Python Imaging Library (PIL) was used for image manipulation and loading, alongside 'cv2' and tensorflow/tensorflow datasets to read and manipulate more loaded images. The last

library would be the 'imghdr' library that helped me identify image files based on the file's contents.

I had no need to cleanse missing values since they were jpeg/jpg files but I removed any images that were too small (anything less than 10kb) would not be as detectable to the model. After this process and uploading my data set file to python through Jupyter notebook, I had a total of 3000 files belonging to 2 classes. I had three sections for my essential data attributes.

Image Data Attributes -

Image pixels: pixel values of MRI images that represent the intensity or color information of each pixel Image dimensions: height and width of the MRI images (necessary for resizing/pre-processing) Image format: the file format of the MRI images (e.g., JPEG,JPG, PNG, etc.)

Label Data Attributes:

Tumor presence: binary label indicating whether the MRI image contains a tumor or not (0 = no tumor, 1 = tumor) Tumor type: my original dataset contained several types of cancers (glioma, meningioma, pituitary, etc), but unfortunately, I did not have an even distribution of data points, therefor I switched to focusing on if the image contained a tumor or not.

Data Split Attributes:

Train, Validation, Test split: indicators to identify what portions of the data set belong to the training set, validation set and test set.

The main goal of this project is to be able to utilize a model that can differentiate whether an MRI image contains a tumor or does not. The independent variable is the image data itself, where the MRI image will contain specific features used to classify it. The Dependent variable is the class label, indicating if it will be classified as a cancer or not.

1.4 Exploratory Data Analysis:

In the exploratory data analysis (EDA) phase, I utilized various techniques to gain insights and understand the structure of the MRI image data for tumor detection. EDA allowed me to make assumptions, identify anomalies, and determine the steps needed for model building. Data visualization played a crucial role in this process, with histograms, box plots, and heatmaps being key tools.

For the image data attributes, I focused on pixel values representing intensity, image dimensions, and image format. The label data attributes included tumor presence (binary) and tumor type, although I eventually focused on a binary classification of tumor presence.

In the EDA process, I first used histograms to visualize pixel intensity distributions separately for tumor and non-tumor images. Next, I compared the distributions using box plots. To gain a deeper understanding, I created heatmaps to display normalized pixel intensity distributions, revealing variations across intensity levels for both classes. Additionally, I computed correlation matrices to identify relationships between different pixel intensities, providing insights into potential feature importance for tumor detection.

Overall, this phase taught me the importance of exploring data thoroughly and leveraging various techniques to prepare for the model-building process. Balancing the dataset and ensuring the quality of images significantly impacted the potential success of my tumor classification model. With these insights, I am optimistic about the project's outcome and the model's ability to produce accurate tumor classifications.

2 Imports and Tools Used in MRI CNN Image Classification Model

The following imports and tools were utilized in the MRI CNN image classification model:

2.1 Data Handling and Preprocessing

- `numpy`: Employed for numerical array operations and data handling.
- `os`: Utilized for file and directory operations.
- `PIL`: Python Imaging Library, used for image processing tasks.
- `tensorflow`: The core library used for building and training machine learning models.
- `tensorflow_datasets`: Used for loading datasets from TensorFlow datasets library.

2.2 Data Splitting and Metrics

- `shuffle` from `sklearn.utils`: For shuffling the data during splitting.
- `train_test_split` from `sklearn.model_selection`: For splitting data into training and testing sets.
- `classification_report` from `sklearn.metrics`: For generating a classification report.

2.3 Model Building

- `tensorflow.keras`: The high-level Keras API for building deep learning models.
- `tensorflow.keras.layers`: For defining different layers in the model architecture.

- `tensorflow.keras.losses`: For specifying loss functions during model training.
- `tensorflow.keras.models`: For building and loading models.
- `tensorflow.keras.metrics`: For specifying evaluation metrics during model training.
- `tensorflow.keras.optimizers`: For specifying optimization algorithms during model training.
- `tensorflow.keras.applications`: For using pre-trained models for transfer learning.
- `tensorflow.keras.preprocessing.image.load_img`: For loading images.

2.4 Data Visualization

- `matplotlib.pyplot`: For creating visualizations such as plots and charts.
- `seaborn`: For enhanced data visualization using statistical plots.

2.5 Miscellaneous

- `tqdm`: For adding progress bars to loops to monitor their progress.
- `random`: For generating random numbers.
- `tensorflow.keras.models.Sequential`: For building a sequential model, where layers are stacked sequentially.
- `tensorflow.keras.layers`: For defining layers in the model, such as Conv2D, MaxPooling2D, Dense, Flatten, Dropout.

Certainly! Below is the LaTeX Overleaf paragraph post, broken up into sections as requested:

3 Model Training and Evaluation

For my CNN model for tumor detection in MRI images, the pipeline was straightforward. It involved data collection of MRI images with binary labels, data preprocessing such as resizing, normalization, and augmentation, and splitting into training, validation, and test sets. The CNN model architecture was designed with convolutional layers, activation functions, pooling layers, etc. The model was then trained iteratively on the training dataset, and its performance was evaluated on the validation dataset to determine its accuracy in detecting tumors.

4 Fine-tuning and Optimization

The machine learning algorithm used in this project is Convolutional Neural Networks (CNNs). CNNs are well-suited for image-based tasks like tumor detection due to their ability to learn spatial features effectively from images. CNNs are a class of deep learning algorithms specially designed for image recognition tasks.

5 Testing and Validation

During the training process, the model was fed batches of MRI images, and the model's updates were done through backpropagation to minimize the difference between predicted and true labels. The model was trained for multiple epochs (20) to gradually improve its performance. The validation data was used during training to tune hyperparameters and avoid overfitting. The testing data was used to evaluate the final model's performance and assess how it generalizes to new, unseen data.

6 Implementation and Evaluation Process

The CNN model was implemented using deep learning frameworks such as TensorFlow and Keras. Python was used for data cleaning, preprocessing, and analysis/evaluation. The model was equipped with an optimizer (Adam) and a loss function to ensure efficient learning during training.

7 Analysis and Results

The MRI CNN image classification model was trained over 20 epochs, and various evaluation metrics were used to assess its performance. The metrics included loss, accuracy, sensitivity (recall), specificity, precision, and F1 score. The model's performance showed improvement with each epoch, evident through decreasing loss and increasing accuracy and other metrics on the validation set. This suggests effective tumor detection in MRI images.

References: GitHub repository containing the source code: https://github.com/jonagitsdata/Capstone-Final-Project-.

Future Directions: Finally concluding if there are any other directions or suggestions to improve the models ability to detect tumors.

7.1 Key Components

a. Utilizing and optimizing code for a Neural Network to detect tumors from MRI images

b. Implementing the appropriate pre-processing techniques for data augmentation, noise reduction, and image enhancement to not jeopardize the quality of the models performance.

c. Training the neural network using specifically labeled MRI images for high accuracy and minimal false negatives/positives.

d. Evaluation of the model's performance using confusion matrix (accuracy, precision, recall, and F1-score)

7.2 Limitations

Availability: : the labeled MRI data sets I've found may be large, but they are data sets that come from 2-3 years ago, on top of the methods I chose to enhance image resolution, reduce noise and better help my neural net model.

Generalization: : the model may train well on specific data set data used for training and testing, but may struggle with unseen data from 2022 or newer depending on the quality of the images.

Computational Restrictions/Domain Expertise: : I am working on a computer that does not have a robust GPU and processor to efficiently provide processing power to enable the machine learning model to classify to its highest ability. Account for differences in picture quality and tagging in images. I've researched several articles that I am not super equipped with breaking down and may want to collaborate closely with medical professionals/radiologists to ensure I understand terminology and interpret the results accurately.

Ethical Considerations: : Dealing with medical data always involves a level of privacy that we must keep. It is important to ensure the data being accessed and used is in compliance with HIPAA regulations.

False Positives/Negatives: Neural Networks can still misclassify a tumor as positive and fail to see one, so it's important to maintain a balance of sensitivity and specificity

8 Literature Review

9 Data Collection

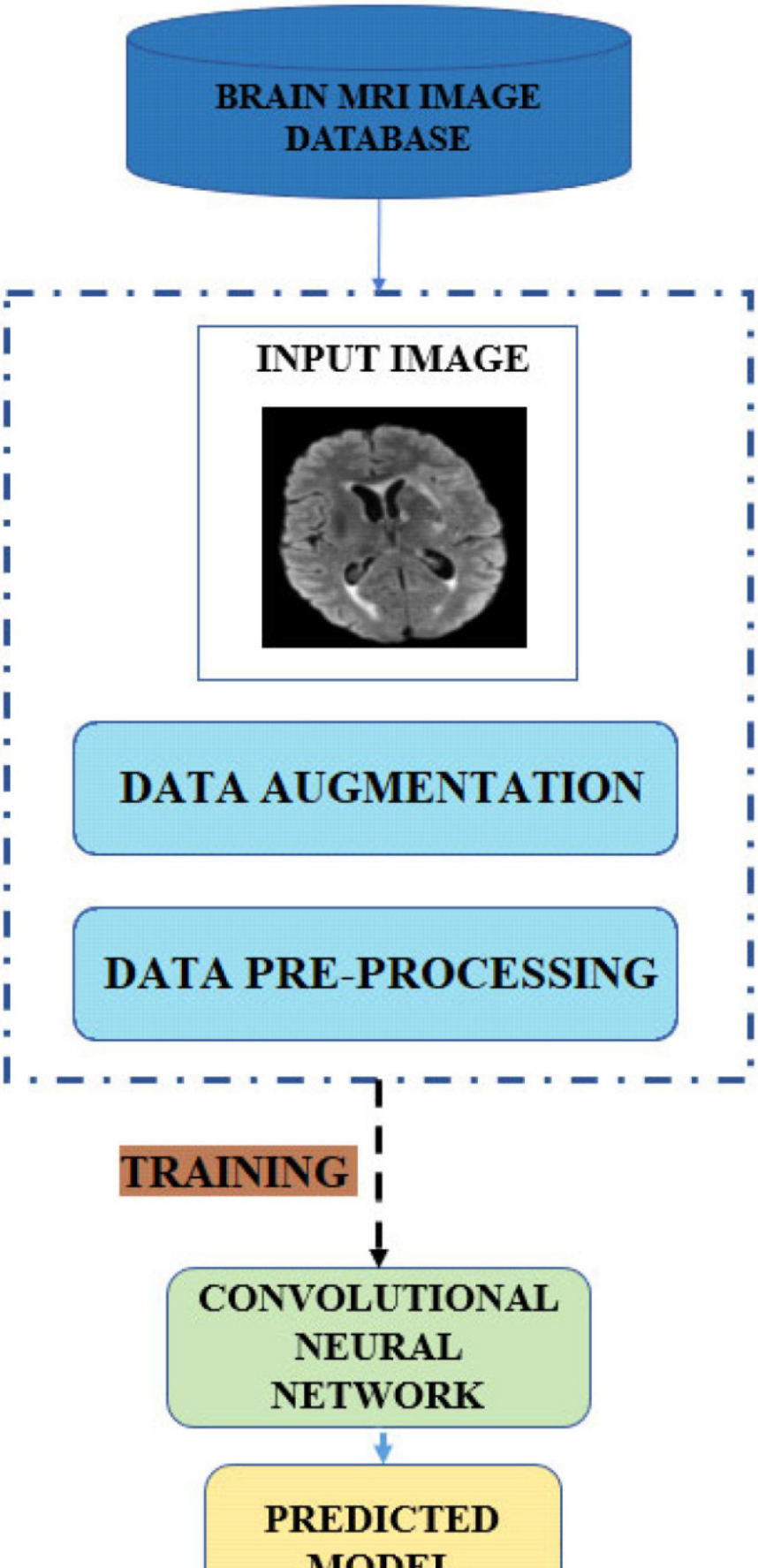
Pre-processing

10 Methodology

11 Limitations

12 Experimental Results

Quantitative data


BRAIN MRI IMAGE DATABASE
INPUT IMAGE
DATA AUGMENTATION
DATA PRE-PROCESSING
TRAINING
CONVOLUTIONAL NEURAL NETWORK
PREDICTED

Qualitative data

13 Analysis and Discussion

14 Ethical Considerations

15 Conclusions and Future Directions

16 References

https://link.springer.com/article/10.1007/s10278-013-9622-7

https://aacrjournals.org/cancerres/article/81/4/1171/649750/Interactive-Classification-of-Whole-Slide-Imaging https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9468505/

https://www.nature.com/articles/s43856-022-00199-0 https://medium.com/miccai-

educational-initiative/how-to-get-started-with-deep-learning-using-mri-data-5d6a41dbc417
https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9 https://www.ncbi.nlm.nih.gov/

diagnostics-13-01562title