Report

Python Implementation Approach for Preparing Annotated Brats 2020 Dataset for YOLO Model Training

Student :
Celina Bedjou

Supervisor :
Dr Malika Bendechache

August 22, 2024

Contents

1 Acknowledgements

I would like to express my profound gratitude to all those who contributed to the success of this internship.

First and foremost, I wish to extend my sincere thanks to **Dr. Malika Bendechache**, my internship supervisor, for her guidance, insightful advice, and availability throughout this experience. Her expertise and support were essential to the completion of my tasks.

I would also like to thank the team for their welcome, their team spirit and in particular **Dr. Ramin Ranjbarzadeh Kondrood** for his warm welcome and collaboration. His team spirit and professionalism greatly facilitated my integration and made this experience highly enjoyable.

Finally, I thank my professors and the administration of **University de Paris-Saclay** for their support and for providing me with the opportunity to undertake this internship under optimal conditions.

This internship marked a crucial step in my academic and professional career. On the one hand, this is my first internship, and on the other hand, it will offer me professional experiences and future opportunities.

2 Introduction

Currently, an estimated 700,000 people are living with a primary brain tumor, with approximately 85,000 new diagnoses each year. Brain tumors, both benign and malignant, disrupt brain function and significantly affect patients and their loved ones. Malignant tumors, like glioblastoma (GBM) and diffuse astrocytic glioma, are aggressive, with GBM patients having an average prognosis of 14 months with treatment. Genetic analysis of these tumors currently requires surgery and weeks of analysis. Developing imaging techniques to predict tumor genetics could reduce the need for multiple surgeries and refine therapy choices.Therefore , The BraTS 2020 dataset comprises various MRI modalities crucial for brain tumor segmentation. Accurate segmentation is vital for effective diagnosis and treatment planning in oncology. This report outlines the objectives of the internship, detailing the methods employed for data processing and annotation.

3 Presentation of the organization

?? I have to speak about the the dim of the images / speak about the size of the organization the field of study and so on ....

4 Objectives of the internship

The primary objective of this internship was to create a comprehensive dataset consisting of MRI images of brain tumors and their corresponding annotation files in YOLO format. This involved assigning bounding boxes to each tumor to accurately depict their location and dimensions. The goal was to ensure high-quality annotations that would be suitable for training YOLO models, ultimately contributing to advancements in the field of medical image analysis.

5 Description of the Topic

The Brats 2020 dataset, part of the MICCAI Brain Tumor Segmentation (BraTS) challenge, represents a comprehensive collection of magnetic resonance imaging (MRI) scans specifically curated for brain tumor analysis. This dataset includes multiple MRI sequences such as FLAIR (Fluid-Attenuated Inversion Recovery), T1, T2, and T1ce (T1 with contrast enhancement), each providing distinct insights into brain tumor characteristics.

5.1 Dataset Overview

The dataset is organized into individual patient folders, identified by unique numerical codes (e.g., BraTS20_Training_001 to BraTS20_Training_151). Within each patient folder, there are multiple .nii (Neuroimaging Informatics Technology Initiative) files representing different MRI sequences:

1. FLAIR : Highlights lesions and edema within the brain.
2. T1 : Provides detailed anatomical information of brain structures.
3. T2 : Reveals abnormalities and lesions within brain tissues.
4. T1ce : Enhances the visibility of tumors and their boundaries post-contrast injection.
5. SEG : In addition to these MRI sequences, each patient folder contains a segmentation (seg) file, denoted as BraTS20_Training_001_seg.nii, which provides annotated masks outlining specific regions of interest such as tumor cores and edema. These segmentation masks are essential for training machine learning models to automatically identify and classify abnormal tissue regions in the brain.

All images were acquired with different clinical protocols and various scanners from multiple (n=19) institutions, mentioned as data contributors here. All the imaging datasets have been segmented manually, by one to four raters, following the same annotation protocol, and their annotations were approved by experienced neuro-radiologists. Annotations comprise

1. Label0: Unlabeled volume
2. Label1: Necrotic and non-enhancing tumor core(NCR/NET)
3. Label2: Peritumoral edema(ED)
4. Label3: Missing(No Pixels in all the volumes contain label 3)
5. Label4: GD-enhancing tumor (ET)

Let's look into the Constituents of Glioma:

1. Edema: Collection of fluid over water, Best seen in FLAIR and T2 weighted sequence. Fingerlike Projection.
2. Necrosis: Accumulation of dead cells. Best seen in T1 post contrast sequence.
3. Enhancing Tumor: Indicate breakdown of blook brain barrier. Seen in T1c post contrast sequence.
4. Non Enhancing: Regions in region that are neither edema, necrosis or enhancing tumor.

The data has been fetch from Brats 2020.

5.2 Importance of the Brats 2020 Dataset

The Brats 2020 dataset plays a crucial role in advancing research and development efforts aimed at enhancing brain tumor diagnosis and treatment planning. Its diverse array of MRI sequences and meticulously annotated segmentation masks provide a valuable resource for developing and evaluating cutting-edge algorithms in medical image analysis.

5.3 Challenges and Opportunities

While the dataset offers significant opportunities, challenges such as variability in tumor appearance across patients and imaging modalities require robust preprocessing and annotation strategies. Addressing these challenges ensures the creation of accurate and reliable datasets essential for training and validating machine learning models.

In conclusion, the Brats 2020 dataset stands as a pivotal resource for driving advancements in medical imaging and artificial intelligence.

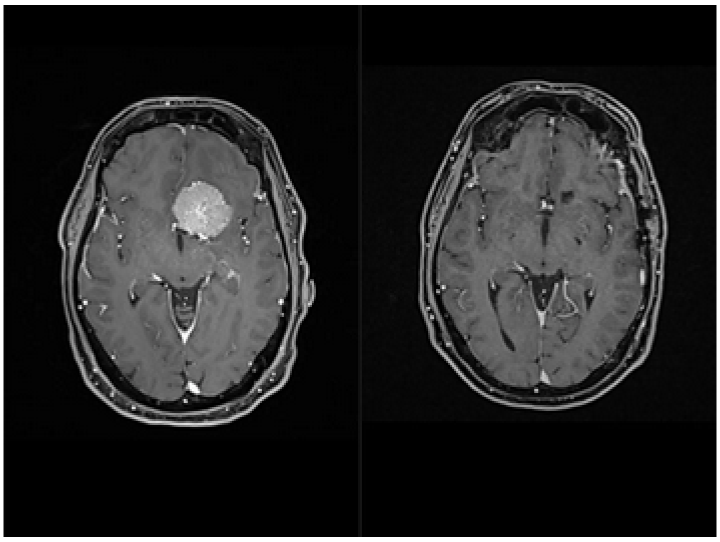


Figure 1: Left side: a brain with a tumor. Right side: a healthy brain

6 Related Work and Materials/Methods

6.1 Description of missions

This report details the steps taken to prepare and annotate the Brats 2020 dataset for training a YOLO model for brain tumor detection using MRI images. The project was conducted over an 8-week internship period.

6.1.1 Assignments given

During my 8-week internship, I undertook the task of preparing annotated data for YOLO model training using the Brats 2020 dataset. This dataset comprises a diverse range of MRI images of brain tumors, which required meticulous preprocessing to convert them into a suitable format for object detection.

The primary objective of this internship was to generate a comprehensive dataset consisting of images and their corresponding annotation files in YOLO format. This involved assigning bounding boxes to each tumor to accurately depict their precise location and dimensions. Throughout the internship, I repeatedly reviewed and refined the dataset to ensure high-quality annotations.

The process was divided into several key phases:

- Introduction to and Configuration of Data (Weeks 1-2): I familiarized myself with the Brats 2020 dataset and configured the necessary Python environment, including essential libraries such as OpenCV, NumPy, and YOLO-specific libraries.
- Preprocessing of Data (Weeks 3-4): I developed Python scripts to preprocess MRI images and their associated segmentation masks, transforming them into bounding box annotations suitable for YOLO.
- Annotating and Validating (Weeks 5-6): I ensured consistency throughout the dataset by automating the annotation process as much as possible and manually validating and improving the annotations to guarantee precision.
- Compilation and Formatting of Data (Week 7): I consolidated the annotated data into a final dataset, ensuring its compliance with the YOLO format, and thoroughly documented the entire process.
- Concluding Analysis and Display (Week 8): I conducted a comprehensive evaluation of the dataset, created a presentation and detailed report summarizing my work, and presented my findings to my supervisor and team.

The outcome of this internship is a high-quality annotated dataset, fully prepared for training YOLO models, which will contribute to advancements in the field of medical image analysis.

6.2 Methodology

6.2.1 Preprocessing of Data

Data Loading with Nibabel

- Method Used:

  nibabel for loading NIfTI files.

- Purpose:

  1. Specialized Handling of Medical Imaging : The BraTS 2020 dataset consists of NIfTI files, a common format for storing medical imaging data. The Nibabel library is specifically designed for reading and writing this format, ensuring that we can accurately access the image data without losing any information.

     ```
     def load_patient_data(patient_folder): # Load the NIfTI files
     ```

     This function aims to load and visualize magnetic resonance imaging (MRI) data in NIfTI format for specific patients. It defines the file paths for the images, checks their existence, and loads them using the Nibabel library. The image data is organized as NumPy arrays, enabling efficient manipulation for analysis. Finally, the code extracts and displays the median slices of the different imaging modalities, providing a useful visual representation for clinical evaluation.

  2. Error Handling:

     By implementing checks for the existence of files, we avoid potential runtime errors and can log warnings if certain files are missing. This makes the code more robust and user-friendly.

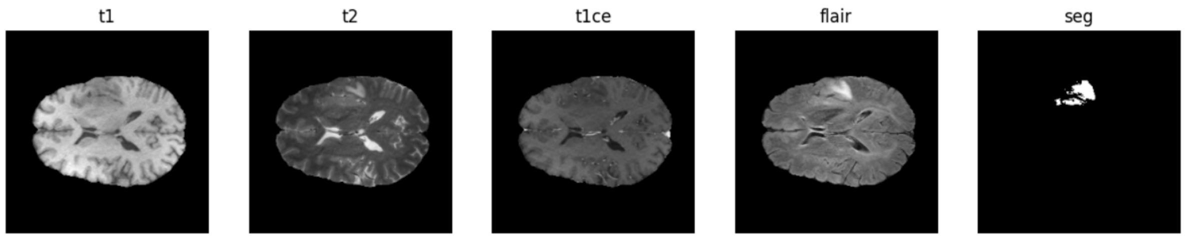


Figure 2: Displaying the 5 different MRI sequences for patient number 001

6.2.2 Generates annotations from the segmentation mask (seg) using the properties of the detected regions

- Method Used:

  regionprops from masks_to_boxes.

- Purpose:

  1. Automated Tumor Detection: The method utilized in this code is to generate bounding box annotations for different regions of a tumor within a segmentation slice. It leverages the **masks_to_boxes** function from **torchvision.ops** to convert binary masks into bounding boxes.. This automation reduces the need for manual annotations, saving time and minimizing human error.
  2. Bounding Box Extraction:

     By extracting bounding boxes for each detected region, we create a format that is compatible with YOLO (You Only Look Once) model training, which requires bounding box annotations for object detection tasks.

- Objective :

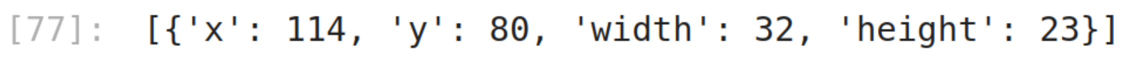
  The objective of this function is to generate bounding box annotations from a segmentation slice of a medical image. Each part of the tumor (necrosis, edema, enhancing tumor) is identified and annotated separately.

```
def generate_annotations(seg_slice): # using the properties of the
detected regions.
```

- Process Steps :

  1. Extract Unique Values: The function starts by extracting the unique values present in the segmentation slice (seg_slice) using np.unique(). A debug statement logs these unique values to help diagnose the content of the slice.
  2. Identify Regions: Labels for each part of the tumor are defined in a dictionary (labels), where each part is associated with a unique integer value (1 for necrosis, 2 for edema, 4 for enhancing tumor). Colors for each part of the tumor are defined in another dictionary (colors) to visually distinguish between the different regions.
  3. Generate Annotations: An empty list annotations is created to store the generated annotations. For each label (necrosis, edema, enhancing tumor), a binary mask is created where pixels corresponding to the label value are set to 1, and others are set to 0. If a binary mask is empty (i.e., contains no pixels corresponding to the label), it is skipped. The binary mask is converted to a PyTorch tensor and a batch dimension is added. The masks_to_boxes function is used to generate bounding boxes from the binary mask tensor. For each bounding box, the coordinates (minimum column, minimum row, maximum column, maximum row) are extracted. An annotation dictionary is created with the part name, color, coordinates (x, y), and dimensions (width, height) of the bounding box. The annotation dictionary is added to the annotations list.

4. Return Annotations: The function returns the list of annotations, containing the bounding box information for the main region of interest.



```
[77]:  [{'x': 114, 'y': 80, 'width': 32, 'height': 23}]
```

Figure 3: The annotations for patient 001 with coordinates (x: 114, y: 80) and dimensions (w: 32, h: 23)

6.2.3 Data Visualization

- Method Used:

  Matplotlib for displaying images and annotations.

- Purpose:

  1. Visual Validation: Visualizing the MRI slices along with the generated annotations allows for immediate qualitative assessment of the annotation process. This is crucial in medical imaging, where visual confirmation can help identify any discrepancies or errors in the automated process.
  2. Interactive Analysis: By overlaying bounding boxes on the MRI images, you can visually assess the effectiveness of the annotations in identifying tumor regions, which is critical for subsequent model training and evaluation.

- Objective:

  The objective of the function is to create a visual representation of different MRI modalities and overlay bounding box annotations on the segmentation mask for a specific slice.

- Process Steps: function is designed to visualize MRI images along with their corresponding bounding box annotations for a specific patient. It processes MRI data from multiple modalities (e.g., T1, T2, FLAIR) and generates a comprehensive visual summary by displaying each modality's middle slice. For the segmentation modality, the function also overlays bounding box annotations on the tumor regions. This visual representation helps in understanding the spatial relationships and characteristics of the annotated regions within the MRI slices. The function is essential for evaluating and presenting the results of medical image analysis, especially for identifying and locating tumors.

```
def display_images_and_annotations(image_data,patient_folder, slice_index):

    plt.imshow(...)
```

6.2.4 Writing annotations to text files

- Method Used: Custom function to write annotations to a text file.
- Purpose:
  1. Model Compatibility: YOLO is a popular object detection framework that requires annotations in a specific format :

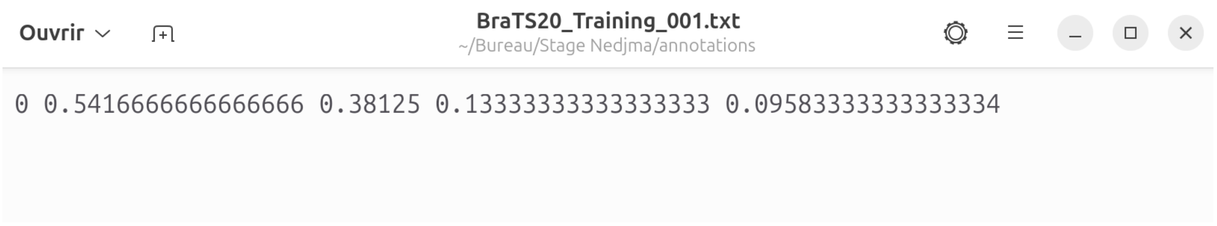
     (class_id, x_centre, y_centre, width, height)

     By converting the annotations into this format, you ensure that they can be directly used for training a YOLO model, facilitating the next steps in your project.
  2. Simplicity and Readability: Writing annotations to a text file keeps the dataset organized and allows for easy access during training. This format is straightforward for both humans and machines to read and parse.

-
  ```
  def write_annotations_to_txt(patient_folder, annotations, image_shape):
  # Write annotations to a text file in YOLO format
  ```

  The write_annotations_to_txt function is responsible for saving bounding box annotations in the YOLO (You Only Look Once) format, which is commonly used for object detection tasks. Here's a breakdown of its functionality:

  1. File Path Definition: The function constructs the output file path by combining the specified annotations_dir directory with the patient's folder name, creating a text file named after the patient (e.g., patient001.txt).
  2. Annotation Processing: It opens the output file in write mode and iterates through the provided annotations. For each annotation: A class ID (set to 0 for 'tumor') is defined. The center coordinates of the bounding box (center_x and center_y) are calculated by adding half the width and height to the top-left corner coordinates (x and y). These values are then normalized by dividing by the width and height of the image, respectively. The width and height of the bounding box are also normalized by dividing by the image dimensions.
  3. Writing to File: The function writes each annotation line in the format required by YOLO: class_id center_x center_y width height, where all values are in normalized form.
  4. Error Handling: If an error occurs during the file writing process, it logs an error message to inform the user about the failure.
  5. Logging: Upon successful writing of annotations, an informational message is logged, confirming that the annotations for the specified patient have been generated and saved.



```
0 0.5416666666666666 0.38125 0.13333333333333333 0.09583333333333334
```

Figure 4: file annotation for patient number 001

6.2.5 Data Validation

- Method Used:

  Custom validation function.

- Purpose:

  1. Quality Assurance: Validating annotations is critical to ensure that the generated bounding boxes do not contain negative values, which could lead to errors during model training. This step is essential for maintaining the integrity of your dataset.
  2. Reducing False Positives: By implementing strict validation rules, you minimize the risk of including incorrect annotations in the training data, which can adversely affect model performance.

- def validate_annotations(annotations): # Check for negative values in annotations

  In summary, the validate_annotations function is a crucial step in ensuring the integrity of the generated tumor annotations by verifying that all dimensions and coordinates are non-negative, thus preventing potential errors in subsequent processing stages

6.2.6 Overview of the main Function

The main function serves as the primary control point for processing MRI data across multiple patients. It systematically performs the following steps:

1. Patient Data Loading: It iterates through each patient folder, calling load_patient_data to load MRI images and segmentation masks. If any data fails to load, it skips to the next patient.

2. Segmentation Validation: The function checks for the presence and correct dimensions of the segmentation data. This ensures that only valid data is processed.

3. Annotation Generation: It selects a central slice of the MRI data and generates bounding box annotations from the segmentation mask using generate_annotations.

4. Annotation Validation: The generated annotations are validated using validate_annotations. If they are valid, it visualizes the images with annotations and saves them in the YOLO format. If not, it logs a warning.

5. Error Handling: Throughout the process, logging is implemented to track issues, providing insights into the status of the processing pipeline.

Conclusion

In summary, the main function efficiently orchestrates the workflow for MRI data processing and annotation generation, ensuring that the pipeline is robust and that only valid data is utilized.

7 Analysis and results

7.1 Results obtained

Here is the result of my work for visualizing the bounding boxes overlaying the first patient's image:

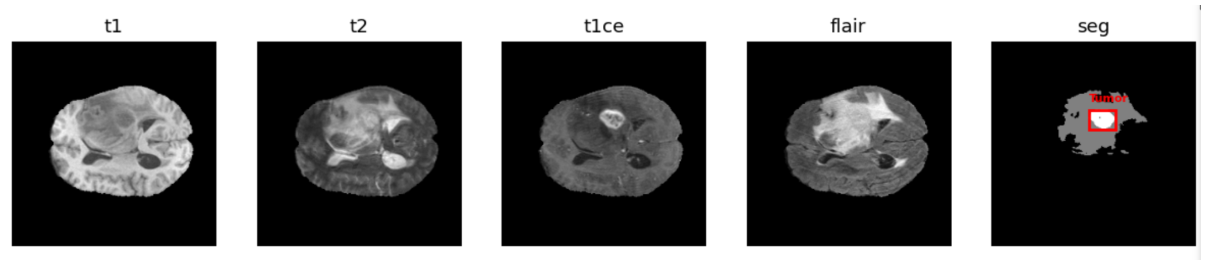


Figure 5: Result of the work on patient number 001

7.2 Difficulties encountered

The project was not without its challenges. One significant issue was ensuring that the dimensions of the images and annotations aligned correctly to avoid runtime errors. To address this, I implemented robust error handling and logging mechanisms, allowing for better tracking of the script's execution and easier debugging.

Moreover, I recognized the potential for further enhancement through data augmentation techniques, which could improve the robustness of the dataset by increasing its variability. This is a potential area for future work.

7.2.1 Challenges

7.2.2 Solutions

7.3 Human and professional achievements

8 Conclusions

My internship provided invaluable experience in processing medical imaging data and implementing practical solutions for real-world challenges in brain tumor segmentation. The skills I developed in Python programming, data manipulation, and validation will serve me well in future endeavors within the field of medical imaging and machine learning.

As I reflect on this experience, I am excited about the possibilities for further research and application of the techniques I have learned. The processed BraTS 2020 dataset stands as a testament to the progress made during my internship and paves the way for future advancements in the automated analysis of brain tumors.

9 Appendices

10 References

1. Report Structure : Springer
2. Code Structure : Git
3. Informations of brain Tumor : Science Direct
4. Images : Hirslanden
5. Data and Availability -Dataset: Kaggle