GEORGE BROWN COLLEGE
SCHOOL OF COMPUTER TECHNOLOGY
APPLIED A.I. SOLUTIONS

DEEP LEARNING-II

Professor: Dr. Mo Fadaee

Group:

Mohammad Ardalanasl

March 25, 2024

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

This project focuses on the development of a robust model for the detection and localization of tumors in brain MRIs. Two distinct approaches were employed: a convolutional neural network (CNN) model trained from scratch and a pre-trained YOLOv8 model. Due to the scarcity of accurate data for this specific task, various techniques were implemented to enhance the model's accuracy and precision.

The first model involved a comprehensive process encompassing data preprocessing, model definition, training, validation, and an in-depth exploration of the fundamental principles of detection and localization in CNN models. In contrast, the second model leveraged the pre-trained YOLO model to enhance precision by capitalizing on its superior feature extraction capabilities. Despite the challenges posed by limited data availability and resource constraints, the final results demonstrate commendable metrics.

1.1 Problem statement

This report addresses the critical issue of detecting and localizing tumors in brain MRIs, which has significant implications for healthcare due to the high stakes involved. The field of AI has seen substantial advancements aimed at expediting and improving the accuracy of diagnosing such diseases. However, the scarcity of accurate data poses a significant challenge. Patient data confidentiality and the high costs associated with medical exams and tests contribute to this challenge.

To mitigate these issues, various numerical approaches were employed throughout the project. The subsequent sections of this report will delve into the specific methods utilized at different stages to tackle these challenges.

1.2 Agile management

Despite being an individual endeavor, this project was managed using agile principles, with tasks and subtasks defined in the Jira platform. This approach aimed to simulate the process of executing the project within predetermined time and budget constraints. The agile methodology proved invaluable in managing uncertainties, including unforeseen issues that arose during the project. Regular communication with other teams, facilitated by acting as scrum masters, enabled the evaluation of the project's progress and each sprint's effectiveness.

1.3 Data

The "Brain tumor object detection datasets" [1] served as the primary dataset for this project, comprising 1100 MRI images along with corresponding bounding boxes of tumors. This dataset is categorized into three subsets based on the direction of scanning in the MRI images.

To address the limited availability of data, we opted to combine all subsets into a single training process rather than treating them as separate learning processes. While the original dataset

allocated approximately 750 images for training and 350 for testing, we made the decision to augment the training set by reducing the size of the test set. Consequently, the test set was reduced to approximately 85 images, while the training set was increased to 1015 images.

The distribution of labels in the training set is reasonably balanced, with 474 labels corresponding to class 0 and 451 labels corresponding to class 1. This balanced distribution ensures that the model is not confronted with unbalanced data, which could affect its performance. Please see Figure 1 for a visual representation of some sample images from the dataset.

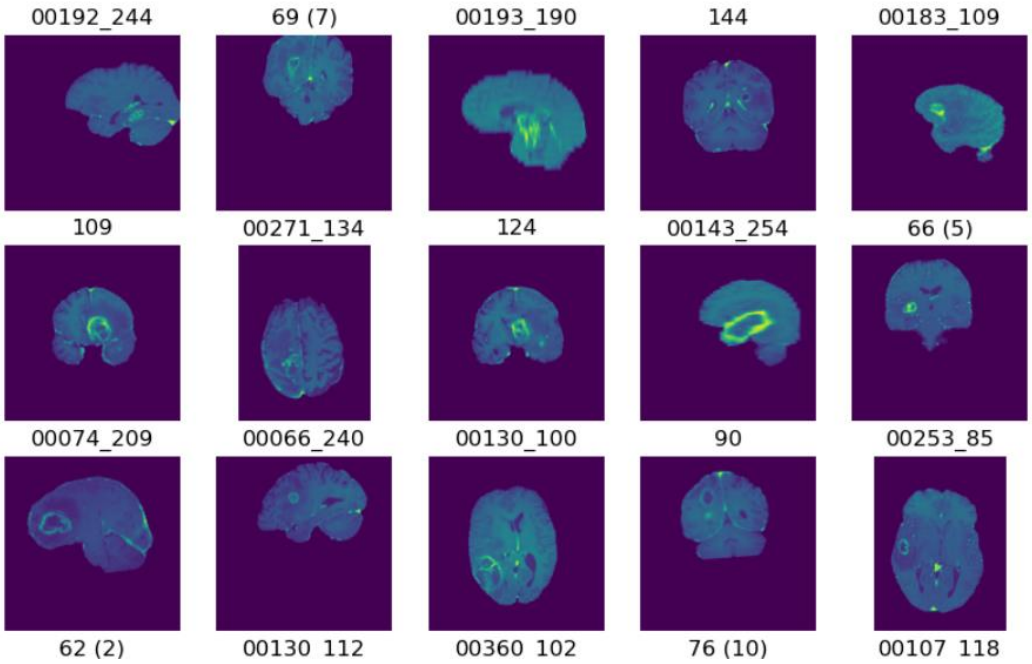

Figure 1. Some sample images of the brain MRIs

Preprocessing Data

In the initial phase of the project, before training the model, it was imperative to preprocess the data to make it suitable for feeding into the model. This involved creating lists of images in both the training and test directories and subsequently reading the contents of the label files, which contained information about the labels and bounding box details for tumors in the MRIs.

The label files contained details about multiple objects detected in the images. While this information would be utilized in an object detection model like YOLO in the second part of the project, the first model focused on classification and localization. Therefore, only a single bounding box per image was used in this model.

Upon reading the bounding box information, we consolidated the data into a data frame, which also included the directories of the corresponding images in the database. This process was repeated for all three sub-datasets mentioned earlier to consolidate all relevant information.

Additionally, we augmented the training set by incorporating parts of the test set to enhance the proportion of training data available for the model.

2.0 METHODOLOGY

This project encompasses the exploration of two primary models, each delving into various aspects of training and object detection concepts.

2.1 Model-1: Convolutional Neural Network (CNN) from Scratch

The first model involves a CNN architecture designed for the classification and localization of tumors in brain MRIs. We sought to optimize the model's architecture by incorporating various layers, such as Conv2D, Batch Normalization, ReLU Activation, Max Pooling2D, skip Connection, Dense, and Dropout. This model comprises approximately 2,243,653 parameters, with 2,242,981 parameters being trainable and only 672 parameters being non-trainable.

Furthermore, we investigated the model's sensitivity to the use of data augmentation during the training process. However, due to the nature of bounding box targets, there were limitations to the extent of augmentation that could be applied. Any alteration to the input image would necessitate a corresponding adjustment to the target label, which had to be synchronized carefully. The performance of this model was evaluated and analyzed using various metrics, including F1 score, classification report, and confusion matrix.

Layer (type)	Output Shape	Param #	Connected to
input_8 (InputLayer)	[(None, 128, 128, 3)]	0	[]
conv2d_42 (Conv2D)	(None, 128, 128, 16)	448	['input_8[0][0]']
batch_normalization_42 (BatchNormalization)	(None, 128, 128, 16)	64	['conv2d_42[0][0]']
activation_42 (Activation)	(None, 128, 128, 16)	0	['batch_normalization_42[0][0]']
max_pooling2d_28 (MaxPooling2D)	(None, 64, 64, 16)	0	['activation_42[0][0]']
conv2d_43 (Conv2D)	(None, 64, 64, 32)	4640	['max_pooling2d_28[0][0]']
batch_normalization_43 (BatchNormalization)	(None, 64, 64, 32)	128	['conv2d_43[0][0]']
activation_43 (Activation)	(None, 64, 64, 32)	0	['batch_normalization_43[0][0]']
conv2d_44 (Conv2D)	(None, 64, 64, 32)	9248	['activation_43[0][0]']
batch_normalization_44 (BatchNormalization)	(None, 64, 64, 32)	128	['conv2d_44[0][0]']
activation_44 (Activation)	(None, 64, 64, 32)	0	['batch_normalization_44[0][0]']
add_14 (Add)	(None, 64, 64, 32)	0	['activation_43[0][0]', 'activation_44[0][0]']

max_pooling2d_29 (MaxPooling2D)	(None, 32, 32, 32)	0	['add_14[0][0]']
conv2d_45 (Conv2D)	(None, 32, 32, 64)	18496	['max_pooling2d_29[0][0]']
batch_normalization_45 (BatchNormalization)	(None, 32, 32, 64)	256	['conv2d_45[0][0]']
activation_45 (Activation)	(None, 32, 32, 64)	0	['batch_normalization_45[0][0]']
conv2d_46 (Conv2D)	(None, 32, 32, 64)	36928	['activation_45[0][0]']
batch_normalization_46 (BatchNormalization)	(None, 32, 32, 64)	256	['conv2d_46[0][0]']
activation_46 (Activation)	(None, 32, 32, 64)	0	['batch_normalization_46[0][0]']
add_15 (Add)	(None, 32, 32, 64)	0	['activation_45[0][0]', 'activation_46[0][0]']
max_pooling2d_30 (MaxPooling2D)	(None, 16, 16, 64)	0	['add_15[0][0]']
conv2d_47 (Conv2D)	(None, 16, 16, 128)	73856	['max_pooling2d_30[0][0]']
batch_normalization_47 (BatchNormalization)	(None, 16, 16, 128)	512	['conv2d_47[0][0]']
activation_47 (Activation)	(None, 16, 16, 128)	0	['batch_normalization_47[0][0]']
max_pooling2d_31 (MaxPooling2D)	(None, 8, 8, 128)	0	['activation_47[0][0]']
flatten_7 (Flatten)	(None, 8192)	0	['max_pooling2d_31[0][0]']
dense_7 (Dense)	(None, 256)	2097408	['flatten_7[0][0]']
dropout_7 (Dropout)	(None, 256)	0	['dense_7[0][0]']
class_output (Dense)	(None, 1)	257	['dropout_7[0][0]']
bbox_output (Dense)	(None, 4)	1028	['dropout_7[0][0]']

Total params: 2243653 (8.56 MB)
Trainable params: 2242981 (8.56 MB)
Non-trainable params: 672 (2.62 KB)

Figure 2. The Architecture of Model-1

2.2 Model-2: YOLOv8 for Tumor Detection

The second model employed YOLOv8 for detecting tumors in MRI images. However, due to the limited data availability and the model's large size, overfitting became a significant concern. To mitigate this issue, we employed the auto-augmentation method, L2 penalty factor, and dropout feature of the YOLO model. These measures aided in the model's convergence, but the results indicated that further improvements could be achieved by increasing the input data. The outcomes generated by YOLO are detailed in subsequent sections of this report.

For the implementation of this model, we utilized the Ultralytics library [2], which offers well-organized functions for YOLO v8. The model comprises approximately 3,011,238 parameters,

with approximately 3,011,222 parameters being trainable. The hyperparameters used for fine-tuning the model are presented in Table 1.

Table 1. The hyper-parameter of Model-2

epochs	**weight_decay**	**patience**	**auto_augment**
200	0.00065	200	'autoaugment'
optimizer	**lr0**	**lrf**	**dropout**
'Adam'	0.001	0.05	0.2

In the following figure, the architecture of the model 2 is shown:

	from	n	params	module	arguments
0	-1	1	464	ultralytics.nn.modules.conv.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.conv.Conv	[16, 32, 3, 2]
2	-1	1	7360	ultralytics.nn.modules.block.C2f	[32, 32, 1, True]
3	-1	1	18560	ultralytics.nn.modules.conv.Conv	[32, 64, 3, 2]
4	-1	2	49664	ultralytics.nn.modules.block.C2f	[64, 64, 2, True]
5	-1	1	73984	ultralytics.nn.modules.conv.Conv	[64, 128, 3, 2]
6	-1	2	197632	ultralytics.nn.modules.block.C2f	[128, 128, 2, True]
7	-1	1	295424	ultralytics.nn.modules.conv.Conv	[128, 256, 3, 2]
8	-1	1	460288	ultralytics.nn.modules.block.C2f	[256, 256, 1, True]
9	-1	1	164608	ultralytics.nn.modules.block.SPPF	[256, 256, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.conv.Concat	[1]
12	-1	1	148224	ultralytics.nn.modules.block.C2f	[384, 128, 1]
13	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1, 4]	1	0	ultralytics.nn.modules.conv.Concat	[1]
15	-1	1	37248	ultralytics.nn.modules.block.C2f	[192, 64, 1]
16	-1	1	36992	ultralytics.nn.modules.conv.Conv	[64, 64, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.conv.Concat	[1]
18	-1	1	123648	ultralytics.nn.modules.block.C2f	[192, 128, 1]
19	-1	1	147712	ultralytics.nn.modules.conv.Conv	[128, 128, 3, 2]
20	[-1, 9]	1	0	ultralytics.nn.modules.conv.Concat	[1]
21	-1	1	493056	ultralytics.nn.modules.block.C2f	[384, 256, 1]
22	[15, 18, 21]	1	751702	ultralytics.nn.modules.head.Detect	[2, [64, 128, 256]]

Model summary: 225 layers, 3011238 parameters, 3011222 gradients, 8.2 GFLOPs

Figure 3. The Architecture of Model-2

3.0 RESULTS

This section presents the outcomes obtained from each model, along with the corresponding efficiency metrics.

3.1 Model-1: without augmentation

After training, the model achieved an accuracy of approximately 0.4886. The mean square error of bounding boxes was approximately 0.0112, and the F1 score for classification was also around 0.4886. In this model, the image size was set to 128 pixels, the batch size was 32, and the number of epochs was 200. The classification report, depicted in the following figure, confirms the model's performance.

	precision	recall	f1-score	support
0	0.46	0.40	0.43	42
1	0.51	0.57	0.54	46
accuracy			0.49	88
macro avg	0.48	0.48	0.48	88
weighted avg	0.49	0.49	0.49	88

Figure 4. The classification report of Model-1 without augmentation

The confusion matrix, shown in the following figure, highlights the model's poor classification performance across the two classes.

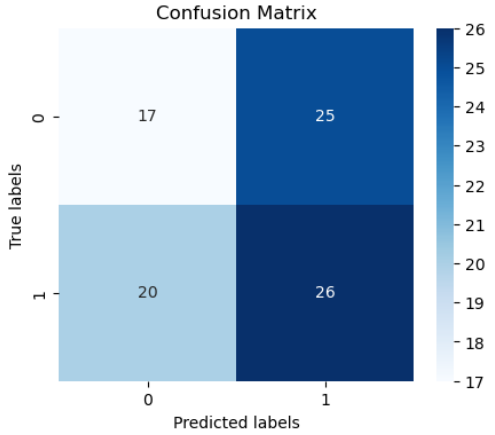

Figure 5. The confusion matrix of Model-1 without augmentation

The loss graph, illustrating the loss values in different epochs for both training and validation sets, indicates a significant gap between the two, indicating an overfitting condition. Thus, increasing the number of input data and employing augmentation techniques could improve the model's performance.

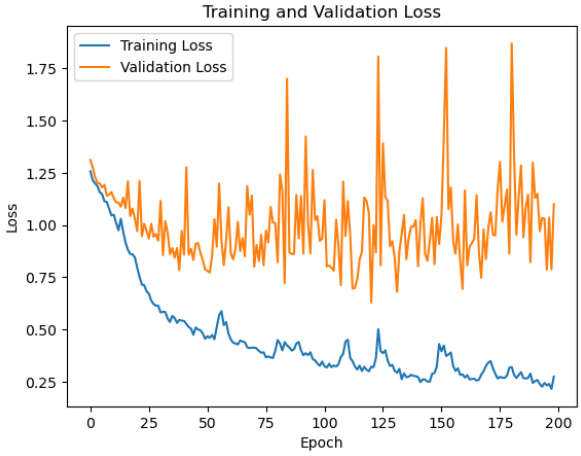

Figure 6. The loss value in each epoch of Model-1 without augmentation

Furthermore, the accuracy graph of Model-1 during the training process demonstrates that while the accuracy of the training set reached approximately 0.9, the validation set remained below 0.8. This suggests a need for additional measures to enhance the model's generalization capability.

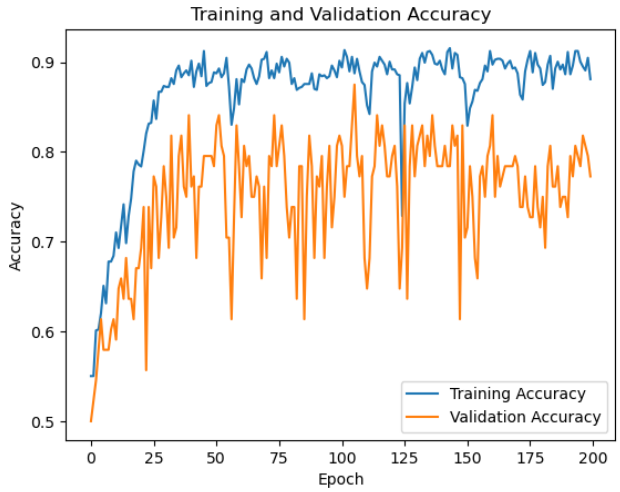

Figure 7. The accuracy in each epoch of Model-1 without augmentation

3.2 Model-1: with augmentation

This model is similar to the previous model but using an augmentation function in input data. In this model the size of data is 128 pixels, batch size is 32, and Number of epochs equals to 125.

After training and validation, the model, the output accuracy is about 0.5455 and mean square error of bounding boxes equals to 0.0116 and finally the F1 score of the classification is equal to 0.5454. Although these results show the improvement by using augmentation in the model, the metrics are not acceptable yet to use as an application. This is because of about 2242981 parameters of this model that are not trained with enough data, so we need much more data to train this model in a proper way. Following, the classification report of the model is shown:

	precision	recall	f1-score	support
0	0.56	0.21	0.31	42
1	0.54	0.85	0.66	46
accuracy			0.55	88
macro avg	0.55	0.53	0.49	88
weighted avg	0.55	0.55	0.49	88

Figure 8. The classification report of Model-1 with augmentation

The confusion matrix of the model 1 with augmentation shows the miss classification in label 0 in the model that indicated the overfitting condition is still exist in the process because the model learned to predict the labels mostly as label 1 rather than 0.

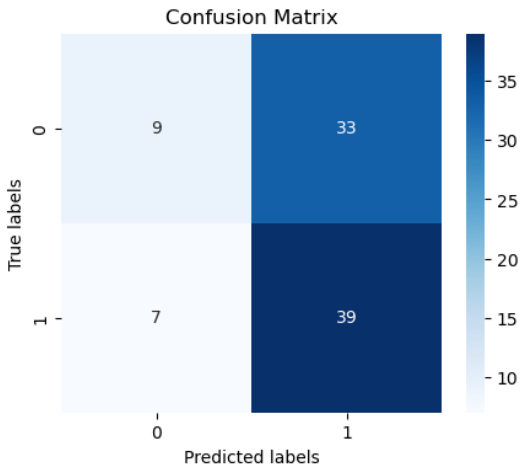

Figure 9. The confusion matrix of Model-1 with augmentation

In the loss value graph, we can see a gap between training and validation results, confirming the overfitting condition of the model.

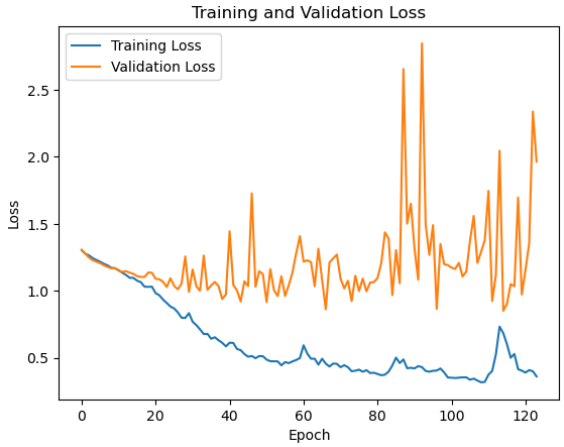

Figure 10. The loss graph in Model-1 with augmentation

Besides, the accuracy of the model is in a lower level than model 1 without augmentation. It shows that although by applying augmentation the f1 score of the model is increased, the pattern of overfitting in this case, predicting most labels as 1, caused less values in the accuracy. Therefore, this index is not a reasonable metrics for analyzing the results.

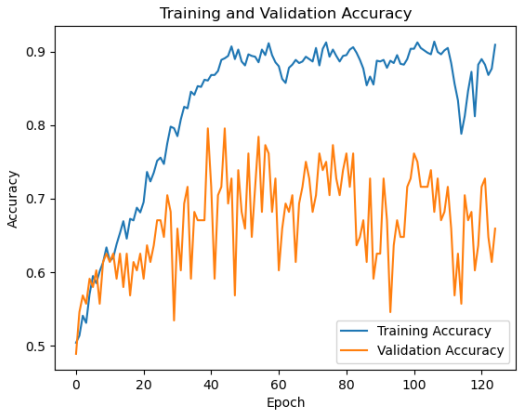

Figure 11. The accuracy graph of Model-1 with augmentation

3.3 Model-2: YOLO v8 model

Model-2 utilizes YOLO as a sophisticated model that automates the training and validation processes, yielding a multitude of results. The model benefits from being pre-trained on a vast dataset, enabling it to converge to superior results compared to Model-1, which was trained from scratch. Advanced features such as auto-augmentation, which introduces various augmentations including adjustments to labels corresponding to random changes in input images, were employed. Additionally, regularization techniques like L2 and dropout with high values were applied to mitigate overfitting.

The confusion matrix for Model-2 on the validation dataset demonstrates improved classification results compared to Model-1. However, further enhancements are necessary for the model to be suitable for practical application, which hinges on the availability of more training data.

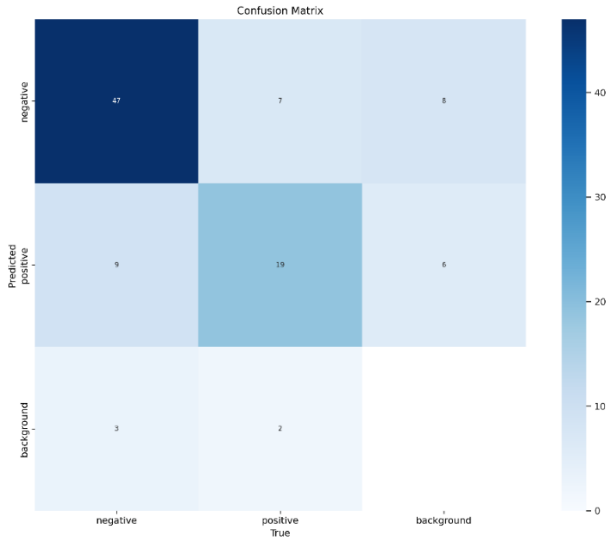

Figure 12. The confusion matrix of Model-2

The F1 vs confidence curve indicates an average F1 score of approximately 0.74 across a wide range of confidence levels, representing a significant improvement of more than 37% over Model-1.

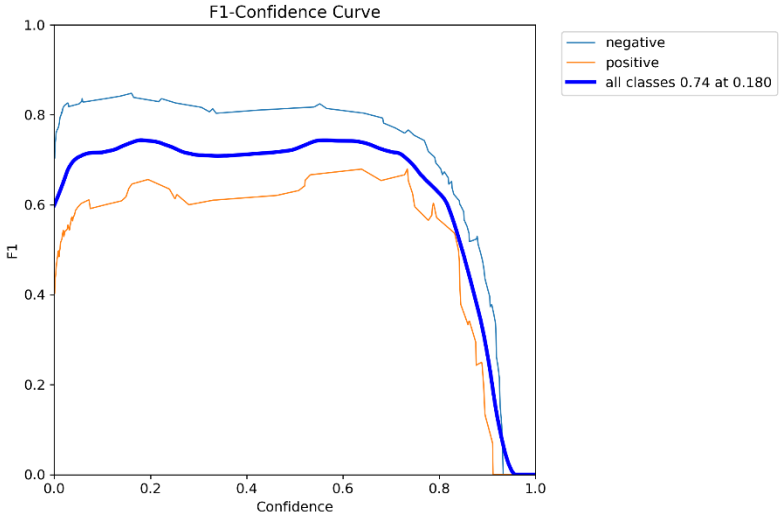

Figure 13. the F1 score value of model-2 vs the confidence value

The precision vs confidence curve shows an average precision of about 0.75, a 36% improvement over Model-1, with precision reaching 1 at a confidence level of about 0.944.

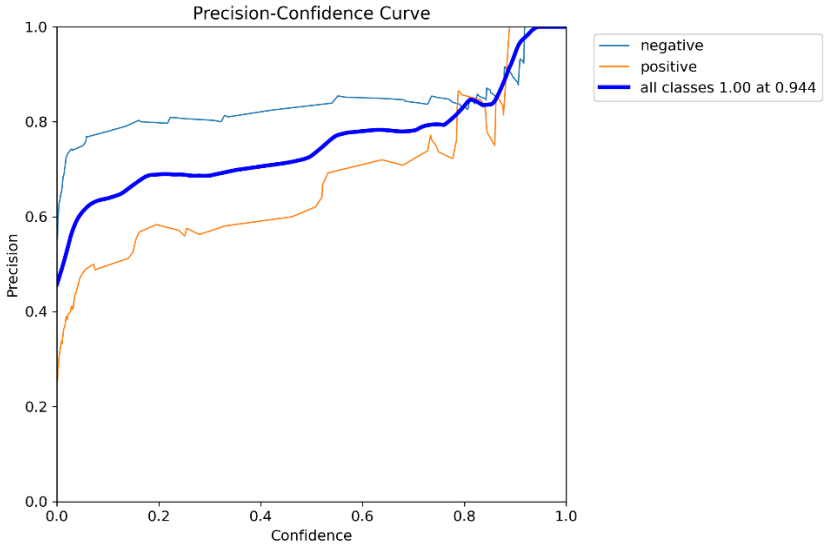

Figure 14. The precision value of Model-2 vs the confidence value

Similarly, the recall vs confidence curve indicates an average recall of about 0.75, a 35% improvement over Model-1, with recall reaching 0 at a confidence level of 0.96.

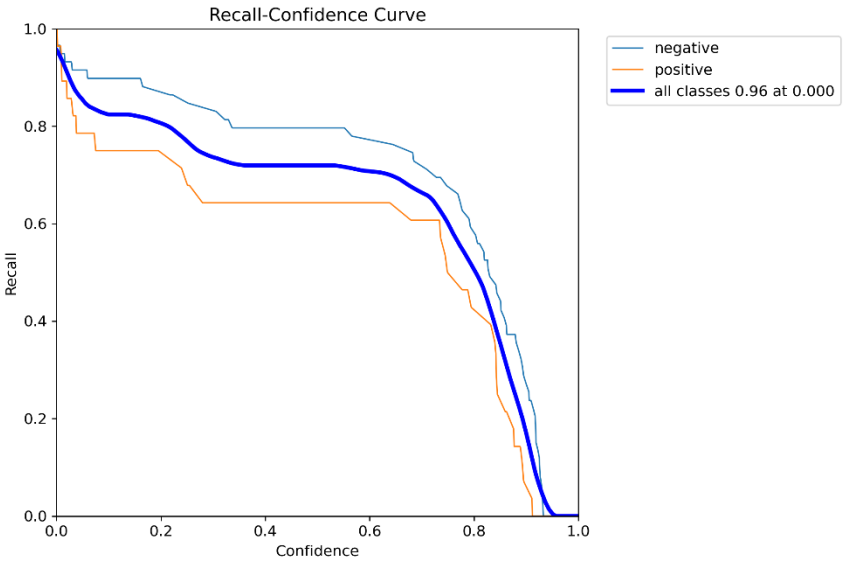

Figure 15. The recall of Model-2 vs confidence value

The precision vs recall curve reveals a mean Average Precision (mAP) at 50% overlap (mAP@50) of 0.789, indicating that the results are satisfactory despite the limited input data.

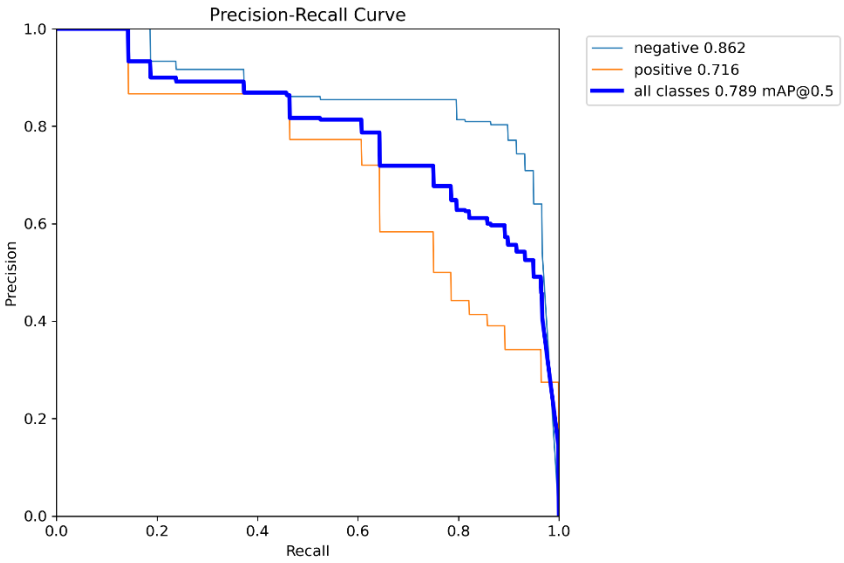

Figure 16. The precision vs recall of Model-2

The summary of classification results in Model-2 compares the losses, precision, and recall of the training and validation sets, highlighting the need for more data to improve predictions related to bounding boxes. The mAP@50 value of approximately 0.79 underscores the model's effectiveness.

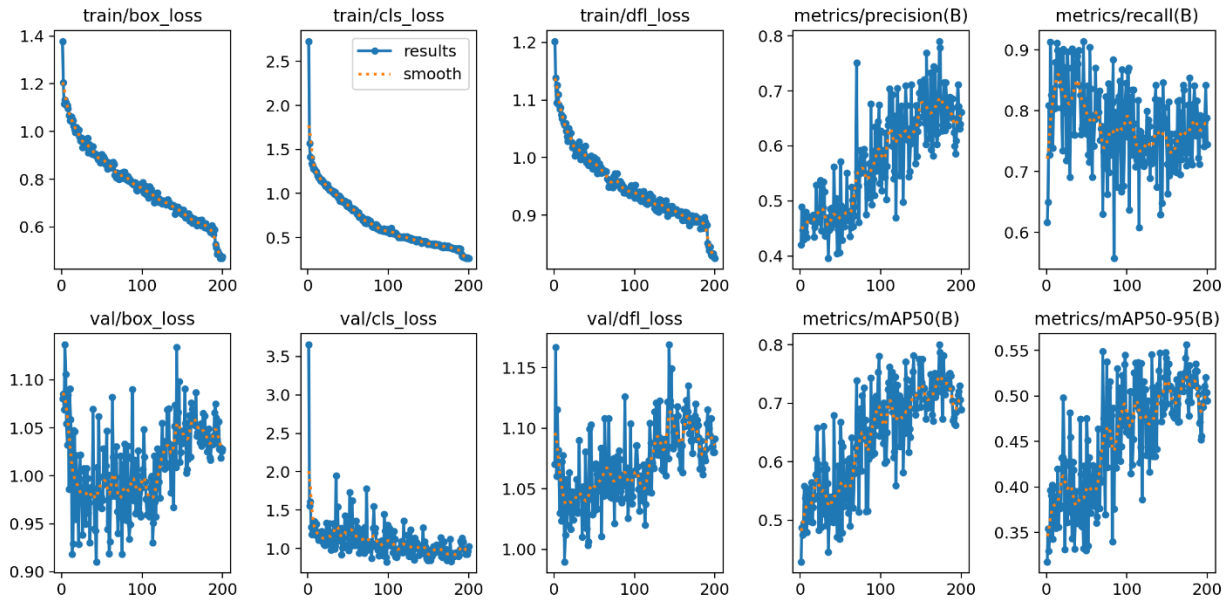

Figure 17. The summary of classification results in Model-2

Samples of real targets and predictions on the validation set demonstrate the model's capability to detect issues, despite limitations in input data and resources. While the model shows promise, it requires additional data to enhance its predictions regarding tumor size and location.

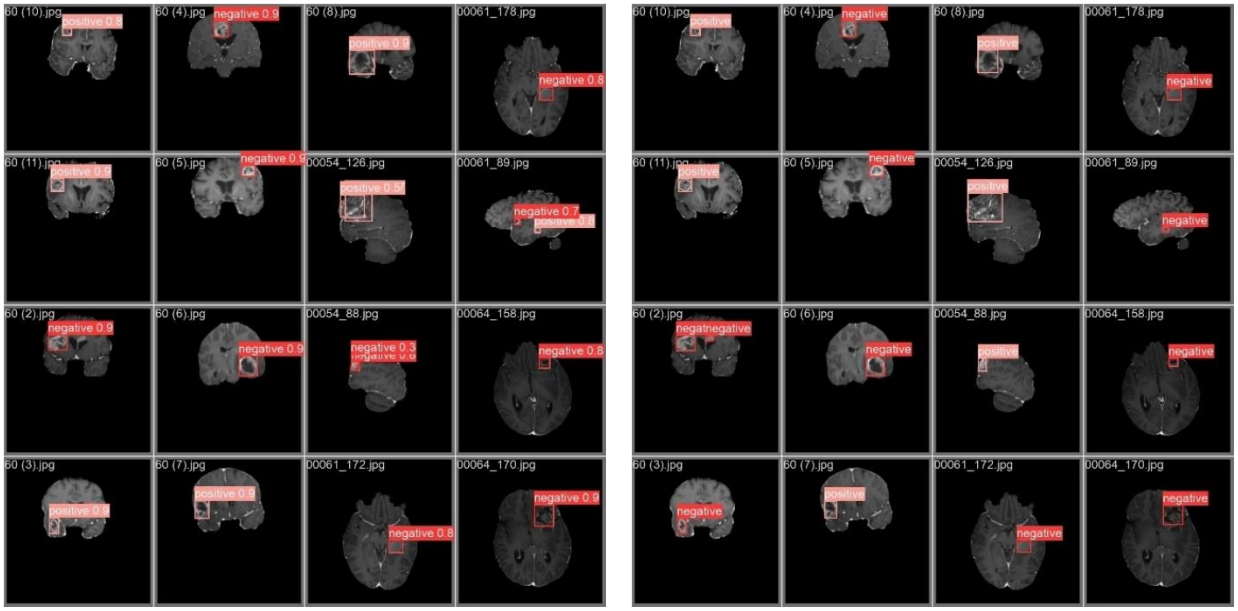

Figure 18. Some samples of real and predicted results of Model-2

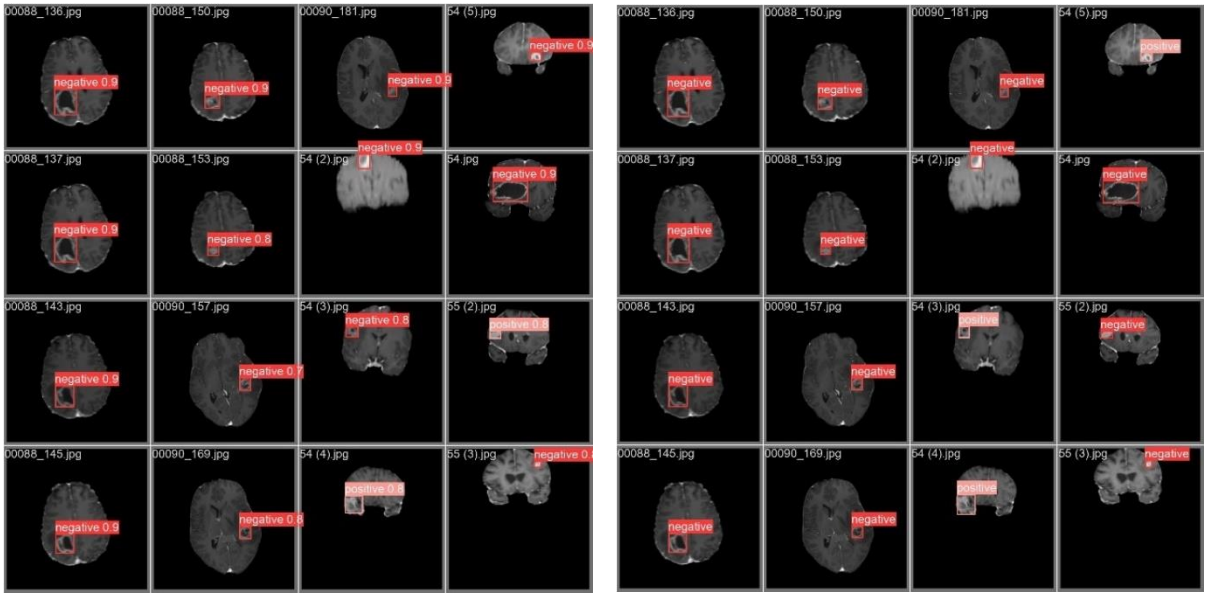

Figure 19. Some samples of real and predicted results of Model-2

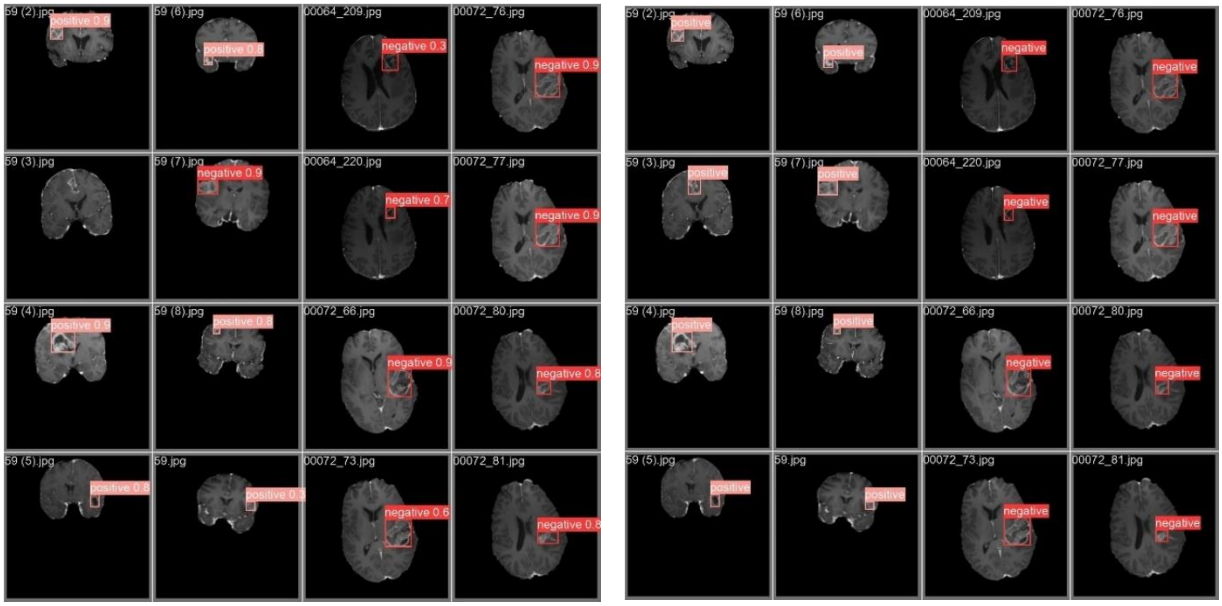

Figure 20. Some samples of real and predicted results of Model-2

Finally, two sample images are presented to illustrate the model's performance in a testing scenario, showcasing its potential utility as an application for investigating MRI results.

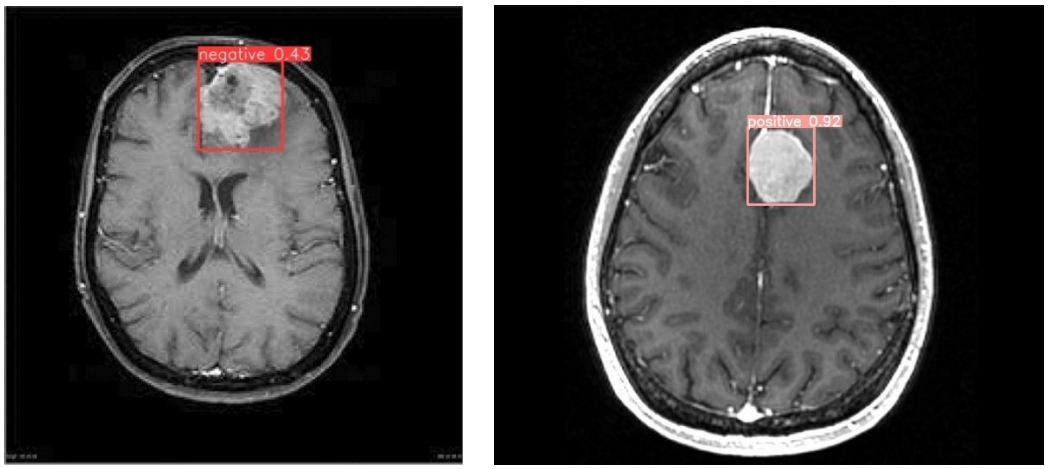

Figure 21. Two sample images for testing Model-2

4.0 CONCLUSIONS

This project aimed to develop effective models for the detection and localization of brain tumors in MRI images. Two main models were explored: a CNN model trained from scratch and a YOLOv8 model. The CNN model, despite achieving an accuracy of around 0.4886, exhibited overfitting and limited performance due to data scarcity. Introducing data augmentation improved the accuracy to approximately 0.5455 but highlighted the need for more data to enhance model training.

In contrast, the YOLOv8 model, leveraging its pre-trained weights and advanced features like auto-augmentation, demonstrated significant improvements. It achieved an accuracy of around 0.74 and displayed superior performance in classification and localization compared to the CNN model. The model's precision, recall, and mAP@50 values further validated its efficiency. Although limitations in data and resources were evident, the results indicate the potential of the YOLOv8 model for accurate tumor detection in MRI images. However, further improvements and additional data are essential for practical application and enhanced predictive capabilities.

5.0 REFERENCES

[1] David Roberts, Brain tumor object detection datasets, Kaggle, https://www.kaggle.com/datasets/davidbroberts/brain-tumor-object-detection-datasets

[2] Ultralytics, https://docs.ultralytics.com/usage/python/