**NEAR EAST UNIVERSITY**

**Faculty of Engineering**

**Department of Software Engineering**

**AI Brain Tumor Detector**

**Graduation Project**

**SWE492**

**Mudaser Mussa**

**Prof. Dr FADI AL-TURJMAN**

**Nicosia – 2000**

# ACKNOWLEDGEMENT

I would like to sincerely thank, everyone that help to build this project, for their important advice, encouragement, and assistance during the preparation of my graduation project. Their knowledge and experience have been crucial in forming this project and guaranteeing its accomplishment.

I also want to express my sincere gratitude to my family and friends for their constant encouragement and support during this trip. Throughout this journey, I have been inspired by their encouragement.

The Near East University administration and technical staff deserve special recognition for their help and timely support when needed. A comfortable learning atmosphere has been guaranteed by their commitment.

I also want to express my gratitude to my university peers and coworkers for their friendship, ideas, and shared experiences, all of which have contributed to this journey's enrichment and memorability. My learning and development have really benefited from our conversations and teamwork. In addition, I would want to express my gratitude to the outside mentors and business professionals who offered insightful criticism at different points along the project. Their expert viewpoints have substantially improved the calibre and applicability of this work.

Lastly, I would want to express my gratitude to Near East University for providing the atmosphere and resources required to finish this research. Working on a project for the Innovation Center has been an honour, and I hope this website will help further its goals.

# ABSTRACT

Brain tumors impact millions of individuals globally and are among the most serious and potentially fatal neurological disorders. In order to improve patient survival rates and determine treatment choices, early and precise identification is important. Nevertheless, manual MRI scan processing by radiologists is a major component of traditional diagnostic techniques, which may be laborious, prone to human error, and constrained by the availability of medical knowledge.

The main goal of this project is to automatically detect and categorize brain cancers from MRI images by using an AI-driven brain tumor detection model with Convolutional Neural Networks (CNNs). The system uses deep learning algorithms to identify patterns in medical photos that is tested and trained and accurately discriminate between instances that are normal and those that have tumors. Data collection from sources, preprocessing, model training, and performance assessment utilizing important metrics including accuracy, precision, recall, and F1-score are all part of the study.

The project's objective is to develop a dependable and effective tool that may help medical professionals by offering automated preliminary diagnoses, drastically cutting down on analysis time, and lowering the possibility of misdiagnosis. Such AI-powered solutions help close the gap in healthcare services and enhance patient outcomes in areas like Northern Cyprus, where access to specialist medical knowledge may be limited. This experiment shows how deep learning may revolutionize healthcare and advances the expanding area of AI in medical imaging.

**Table of Content**

**Acknowledgement** ............................................................. 1

**Abstract** .............................................................................. 2

**Introduction** ....................................................................... 3  
 3.1 Problem Statement ....................................................... 4  
 3.2 Objectives .................................................................... 4  
 3.3 Importance of the Project ............................................ 5

**Literature Review** ............................................................ 6

**Chapter 1: My Project Setup** ......................................... 9  
 5.1 Project Overview ....................................................... 9  
 5.2 Comparison Table .................................................... 10

**Chapter 2: Methodology** ........................................ 11  
 6.1 Data Collection ....................................................... 12  
 6.2 Tools and Technologies ......................................... 13

**Chapter 3: Development Journey** ......................... 14  
 7.1 Learning Path .......................................................... 14  
 7.2 Why Python? ............................................................ 16

**Chapter 4: Visualization Techniques** ................... 17  
 8.1 Tumor Detection with Contours ......................... 17  
 8.2 Data Augmentation ................................................ 18

**Chapter 5: Model Architecture** .............................. 19  
 9.1 VGG16 Model with Transfer Learning ............... 19  
 9.2 Custom Layers ......................................................... 20  
 9.3 Compiling and Freezing ....................................... 21

**Chapter 6: Model Performance** ............................ 22  
 10.1 Accuracy and Loss Curves ................................. 22  
 10.2 Observations ......................................................... 23

**Chapter 7: Evaluation** ............................................ 24  
 11.1 Validation Results and Confusion Matrix ....... 24  
 11.2 Test Set Evaluation ............................................ 25  
 11.3 Misclassified Image Analysis ......................... 26

**Chapter 8: Model Saving** ...................................... 27

**Conclusion** .............................................................. 28

**References** ............................................................... 29

**INTRODUCTION**

Brain tumors are a serious and sometimes fatal disorder; each year, hundreds of new cases are identified. From benign (non-cancerous) growths to malignant (cancerous and aggressive) tumors, brain tumors can vary in complexity and need prompt medical attention. For the diagnosis of brain malignancies, magnetic resonance imaging (MRI) is one of the best methods available. Expert radiologists are needed for the highly specialized task of manually analyzing MRI images. In addition to taking a long time, this procedure is prone to human error and might result in incorrect diagnosis.

Deep Learning (DL) and Artificial Intelligence (AI) have advanced so quickly that computers can now do things that were previously only possible with human knowledge. Automated medical image analysis is one of the most promising uses of AI in healthcare. Convolutional Neural Networks (CNNs), a type of deep learning model, have demonstrated exceptional performance in identifying anomalies in medical pictures. Large MRI scan datasets may be used to train CNNs, which will enable AI models to accurately distinguish between areas of the brain impacted by tumors and healthy brain tissue.

**1. Problem Statement**

The diagnosis of brain tumors is still difficult despite advances in medical technology for a number of reasons:

* Time-consuming Procedure: Manually analyzing MRI images takes a long time, which postpones diagnosis and care.
* Subjectivity and Human Error: Various radiologists may have differing interpretations of the same MRI scan, which might result in conflicting diagnoses.
* Restricted Access to Specialists: It might be challenging to obtain prompt and precise diagnoses in regions such as Northern Cyprus due to the lack of highly qualified neurologists and radiologists.

An automated method that can help radiologists by promptly and precisely identifying brain tumors in MRI images is desperately needed in light of these difficulties. AI-powered solutions can serve as a second opinion, increasing the effectiveness of diagnoses and assisting medical practitioners in reaching more informed conclusions.

**2. Objectives**

By creating a deep learning-based model that can automatically evaluate MRI images and categorize them as either tumor-positive or tumor-free, this study seeks to overcome the difficulties in brain tumor identification. The main goals are:

* Create a CNN-based deep learning model that can recognize brain cancers from MRI pictures with accuracy.
* To assess the model's performance, train and test it using publically accessible MRI datasets.
* Adjust hyperparameters and apply strategies like data augmentation to maximize the model's accuracy.
* Evaluate the model's performance in comparison to manual diagnosis and conventional machine learning techniques.
* Emphasize how AI-driven tumor identification may be used practically to increase access to healthcare, especially in areas like Northern Cyprus where medical knowledge is scarce.

**3. Importance of the Project**

There are several practical advantages to creating an AI-based brain tumor detection system, such as:

* Faster Diagnosis: AI models shorten the time required for a preliminary diagnosis by processing and analyzing MRI data in a matter of seconds.
* Greater Accuracy & Consistency: AI guarantees consistent identification in every case, in contrast to human radiologists who could make mistakes.
* Support for Medical Professionals: This system will serve as a decision-support tool to assist physicians in making quicker and more precise diagnoses, rather than taking the place of radiologists.
* Possibility of Implementation in Northern Cyprus: Medical specialists are in limited supply in many places, including Northern Cyprus. An AI-based detection system can facilitate early diagnosis and lessen the workload for medical practitioners.

This study shows how AI has the ability to completely transform the healthcare sector by fusing computer vision, deep learning, and medical imaging. AI-powered diagnostic tools have the potential to become commonplace in clinics and hospitals throughout the globe with further development, improving healthcare accessibility and efficiency.

**Literature Review: Detection of Brain Tumors via Artificial Intelligence Models**

My studies in machine learning and software engineering brought me to the field of brain tumor detection. The ability of technology to save lives is fascinating. Early discovery of brain tumors can aid in developing a treatment plan that is appropriate for the patients. A radiologist often completes this stage by manually sorting through MRI scans, which is a very time-consuming and occasionally subjective process. Therefore, there is potential for progress, and doctors are already receiving assistance with these duties from quicker and better methods like deep learning and artificial intelligence (AI).

This review's objective is to look at current AI brain tumor detection methods and assess them in light of a modest project I have in mind. Since I'm still studying, I haven't created the model yet, but my objective is to at least create a simple version that walks me through the principles of picture categorization, particularly as it relates to the medical industry.

**What Already Exists**

In the healthcare industry, research and technical developments have produced advanced technologies and models designed to address this issue. The following are a some of the more noteworthy ones: In the healthcare industry, research and technical developments have produced advanced technologies and models designed to address this issue. The following are a some of the more noteworthy ones:

1. Research on VGG19 I came onto a paper in which the researchers classified brain tumors and even segmented them using a VGG19 model. The BraTS dataset, a well-known collection of MRI pictures with tumors and labels, was used to train the algorithm. Their model performed admirably, achieving about 94%. Furthermore, VGG19 is a big model that requires a lot of data and a good GPU, which is difficult for a beginner like me.

2. ResNet50 for Improved Education ResNet50, a model that includes shortcut connections to aid in the training of deeper networks, was used in another article. They also used the BraTS dataset to train the model, and it is far more stable throughout training. Although this model achieved an accuracy of about 95%, it is still a little bit complex for novices. Setting it up and fine-tuning it would need a significant amount of effort and cumbersome hardware.

3-IBM Watson Health

IBM Watson's healthcare group has created advanced artificial intelligence-powered apps that are focused on evaluating medical pictures to aid in diagnosis. Despite their strength and accuracy, these technologies are nonetheless proprietary and closed-source, which restricts students' and learners' access. Furthermore, there is minimal opportunity for creative learning on a smaller scale because these systems are made for bigger hospital networks.

4-Aidoc’s AI Tools

Aidoc focuses on real-time artificial intelligence systems that automatically detect serious problems in brain scans, such tumors and bleeding. These technologies are exclusive and only available in therapeutic settings, just as IBM's offering, making them unavailable for individual research, experimentation, or discovery.

**CHAPTER 1**

1.1 **My Project: Keeping it simple**

I intend to construct a simple CNN (Convolutional Neural Network) for my project to identify brain cancers in MRI pictures. I just want to learn how the process works and try creating something useful; I'm not trying for really high accuracy.

Here’s what I’m focusing on:

1. A straightforward model with a few pooling and convolutional layers
2. Code that doesn't require specialized hardware to execute on my laptop
3. Utilizing a publicly available dataset, such as a scaled-down version of brats or something from Kaggle

This will assist me in understanding the entire process of machine learning projects, from preprocessing data to training and assessment. In the future, I hope it will also give me the courage to attempt more intricate models.

**1.2 Comparison table**

Below is plain text version of the comparison table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Solution | Model type | Accuracy | dataset | complexity | Training time |
| My Project | Simple CNN | TBD | BraTS/Kaggl | low | short |
| VGG19 Approach | VGG19 CNN | ~94% | BraTS | high | long |
| ResNet50 Model | ResNest50 | ~95% | BraTs | Very high | long |
| IBM Watson Health | Propritary DL | ~95% | Private | Very high | unknown |

### **CHAPTER 2**

### **2.1 Methodology**

I created a project that uses a Convolutional Neural Network (CNN) model to identify brain tumors. My primary objective was to develop a program that could precisely identify brain cancers by analyzing medical photos. This research is classified as a machine learning-based analysis of medical images.

**2.2 Data Collection**

To get to the dataset that I'll be using I went to the web and looked for publicly available medial image datasets, particularly from Kaggle. These images were in the form of MRI or CT images that are necessary to train the CNN model. Given that the dataset was small, I employed the technique of Data Augmentation in order to artificially make it larger – this enhances the generalization of my model. In addition to collecting information online, I had conversations with people who in the trenches of this (medical imaging) to try to get a more nuanced idea of what the issues were." YouTube was another one, as I saw a number of tutorials and educational videos to understand CNN’s in general and how to detect brain tumours.

Tools & Technologies Python was the main programming language I utilized to create the brain tumor detection system. Python was the best option because of its extensive library support, particularly for data science and machine learning. In order to effectively modify and analyze the medical photos, I also made use of OpenCV, a robust image processing package. My preferred development environment was Google Colab due to its ease of use and smooth support for Python and machine learning frameworks in addition to that it has all libraries in it no need to download it.

**CHAPTER 3**

**3.1 Life journy of my project**

Starting this project was quite challenging since it was my fIrst time dealing with AI project. I tried using ChatGPT at first, but I immediately find out that I needed more practical and hands-on assistance. I then spoke with Mr. Ibrahim, who had more experience with AI projects. He introduced me to ideas like model layers and data management while carefully describing how CNN models operate. Before starting the brain tumor identification work, he advised me to start with a more straightforward project to lay a strong foundation.

I took his advice to heart and looked through YouTube's beginner-friendly AI projects. I came upon a project that utilized CNN to identify fruits and veggies, and it looked practical and instructive. But while working on that project, I stumbled across a big problem: the model mistook apples for oranges and had an accuracy of just around 55% which i remember Mr. Ibrahim mention that 50`s percent is a bad model. I looked into the issue and found that the poor accuracy and inaccurate predictions were caused by an unintentional random file I had included to the training dataset. Resolving this problem showed me the value of properly arranging data and significantly enhanced the model's performance.

From this experience, I gained knowledge about the importance of appropriate data management and preparation from this experience, which I then used in the brain tumor detection project. I came to see how important it is to thoroughly review and clean the dataset before using it to train the model. I also gained knowledge on image processing contours. I was able to remove extraneous portions of the MRI pictures by using contours to identify the extreme spots. The model's ability to detect brain cancers was improved by this preprocessing step.

But I soon realized that the brain tumor detection model was very different from the fruit and vegetable study when I first started working on it. Because of their greater complexity, the medical pictures needed more sophisticated preprocessing methods. In addition to looking through GitHub to research related projects, I had to get in touch with a buddy who had more medical image processing knowledge. I gained fresh perspectives on how to organize my model and address certain issues by examining the code of other engineers.

Keras was one essential tool that helped me organize my project. By supplying layers and functions that made the design and training of the CNN model easier, Keras offered a simple method for building the model. I was able to effectively test out various topologies and optimization strategies thanks to its connection with TensorFlow. I was able to concentrate on increasing model correctness and drastically cut down on development time after learning how to utilize Keras.

I learned a lot from this practical experience, including how to generate datasets, divide them into training, validation, and testing sets, and efficiently train a model. Important ideas like epochs, testing, and model validation were also taught to me. My confidence increased after finishing this assignment, and I now have the abilities I need to take on the more challenging brain tumor detection model.

**3.2 Why did I choose Python?**

Python was a logical choice because of its vast ecosystem of libraries, including TensorFlow and OpenCV, which are essential for creating and refining CNN models. In particular, OpenCV's powerful image processing features proved helpful. For someone who is new to deep learning in particular, Kera's made model development quicker and easier. Google Colab was an obvious choice since it provides a welcoming environment and computing capabilities for the development of AI using Python. The foundation for the brain tumor detection system was laid when I spoke with Ibrahim and took his advice to start small in order to gain a practical grasp of CNNs.

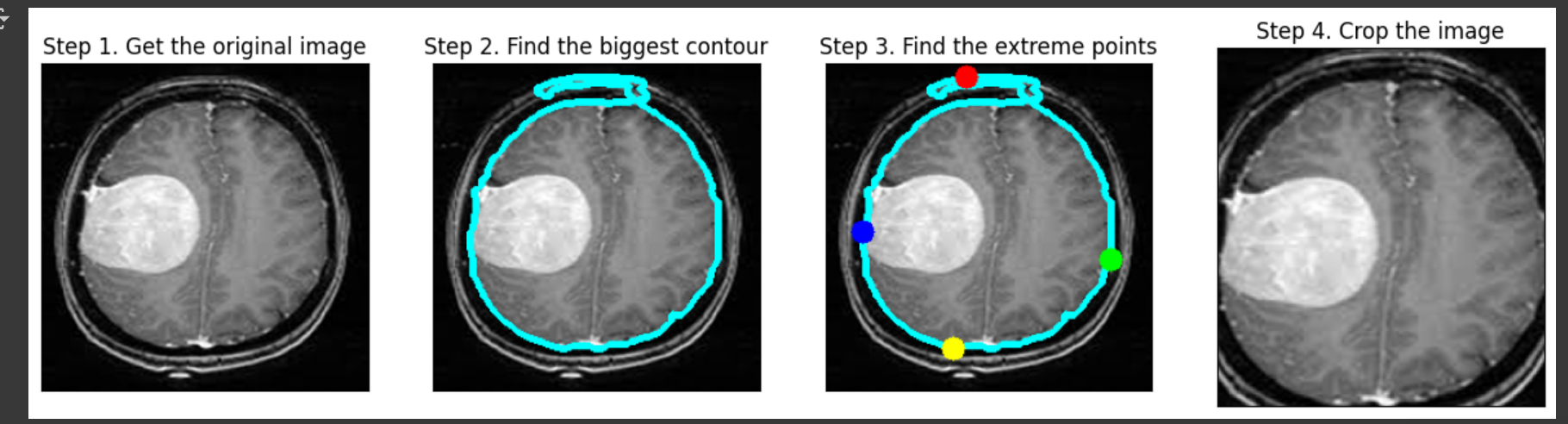
**CHAPTER 4**

**4.1 Visualization and interpretation of Tumour Detection Results**

So, as you can see in this picture I used contours

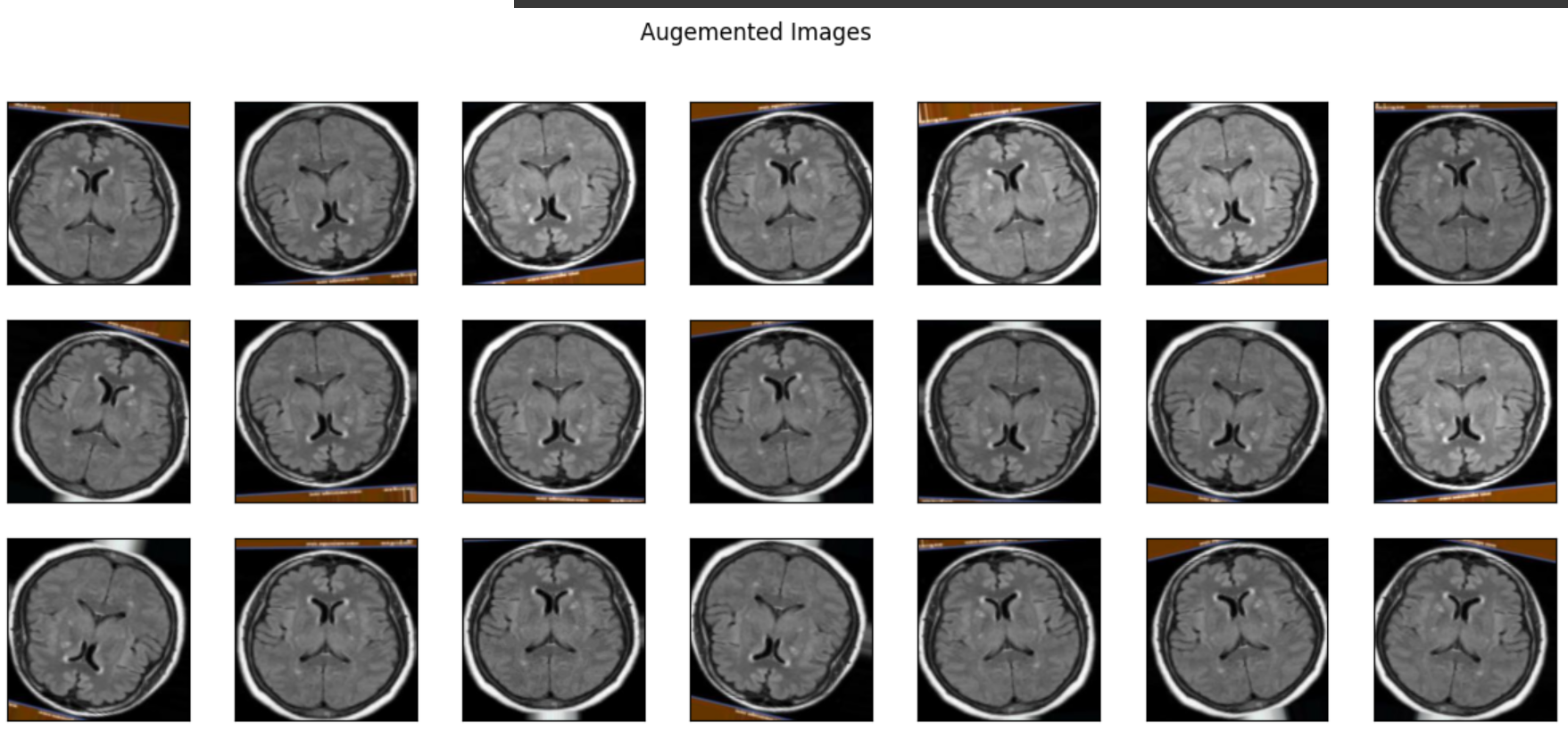
First, I convert the image to grayscale using OpenCV {cv2.cvtColor()}

That to simplify processing. Then we apply the gaussian blur to reduce noise and improve edge clarity. After that we use the canny edge detection to identify the edges of tumor region. that will help me to extract using contour on orginal image to highlight the detected boundries , making the tumer region more clear.



**4.2 Augmentation**

To enhance the unseen images, we try to generate more picture form the original images the model becomes more accurate. It will train one image in different rotation, shifting, rescaling so that we have more image to train from different sides. It will be better in recognizing tumours.



**CHAPTER 5**

**5.1 Result and Discussion**

5.1.1 Model Architecture and Explanation

In this project, we used the VGG16 pretrained convolutional neural network with transfer learning to create a brain tumor detection model. The model's construction and the function of each component are explained in detail below.

5.1.2 Pretrained Model VGG16

As a basis feature extractor, we used VGG16, a deep convolutional neural network that was first trained on the ImageNet dataset. The VGG16 model has a reputation for being simple to use and having strong feature extraction capabilities. In our execution:



5.1.3 Custom layer on Top of VGG16

After loading the VGG16 we add layers to perform classfication:

model = Sequential()

model.add(base\_model)

model.add(layers.Flatten())

model.add(layers.Dropout(0.5))

model.add(layers.Dense(NUM\_CLASSES, activation='sigmoid'))

Flatten(): Creates a one-dimensional vector from the VGG16 base's multidimensional output.

Dropout(0.5): Randomly turning off half of the neurons during training helps prevent overfitting.

Dense(NUM\_CLASSES, activation='sigmoid'): An output layer that is fully connected for binary classification (tumor or no tumor). The likelihood that a tumor is present is represented by the sigmoid activation's output, which ranges from 0 to 1.

5.2 Freezing the Base Model

To avoid retraining the entire VGG16, we pause the layers:

model.layers[0].trainable = False

This implies that the VGG16 will be trained using the only layers we introduced. This has the advantage of cutting down on training time and preventing needless pretrained modifications that could include helpful image feature representation.

5.3 Compiling the Model

Finally, the model is compiled using:

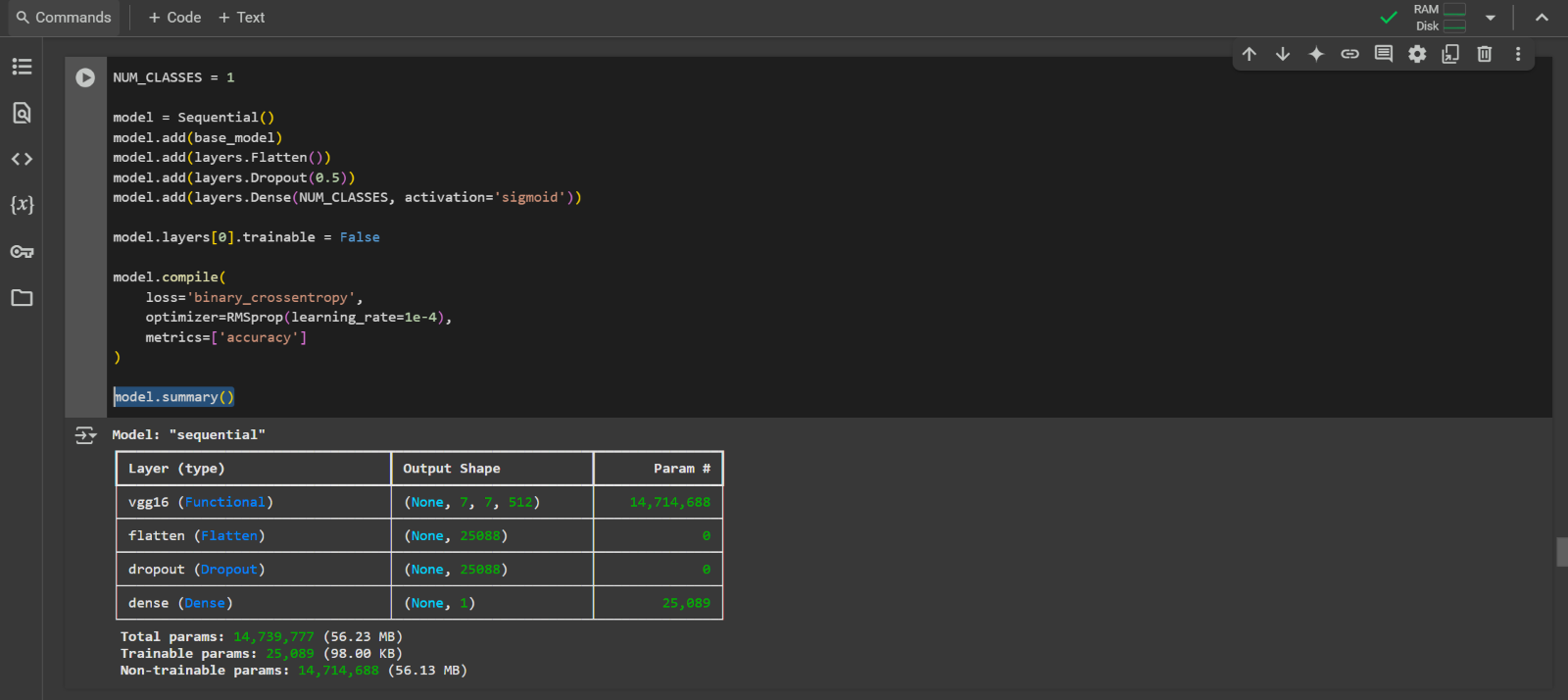
model.compile(

loss='binary\_crossentropy',

optimizer=RMSprop(learning\_rate=1e-4),

metrics=['accuracy']

)



**CHAPTER 6**

**6.1. Model Performance**

We plotted the accuracy and loss for the training and validation sets over each epoch to assess how well the model learned during training. The model's performance during training is demonstrated clearly by these charts.

**6.2 Curve of Accuracy**

The accuracy of the model throughout 30 epochs on both the training and validation datasets is displayed in the figure below.

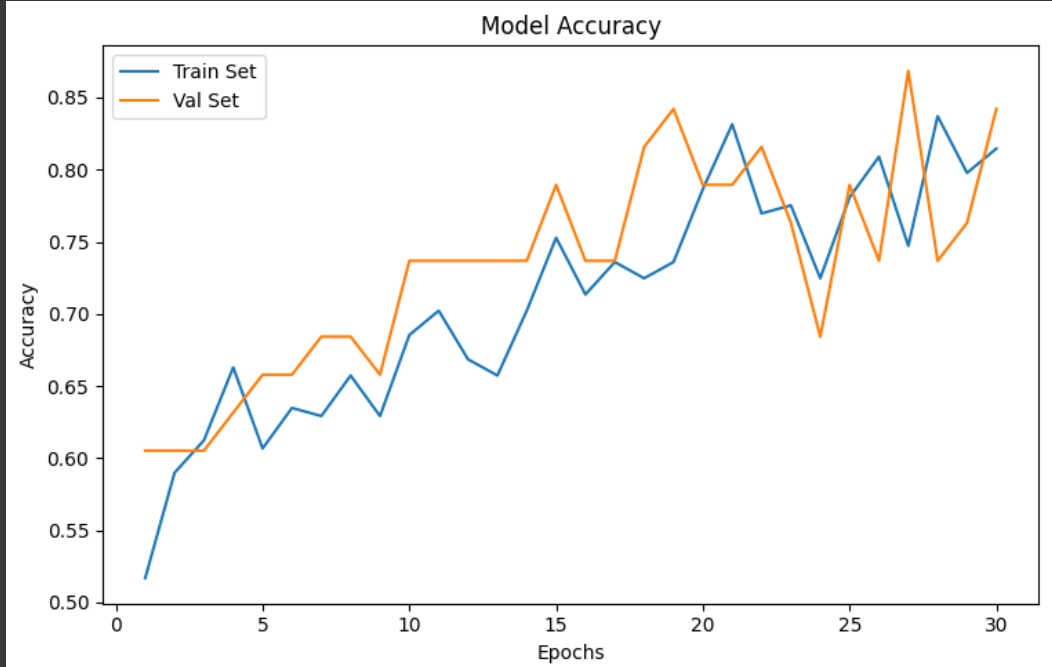
![Model Accuracy Graph]

**6.2.1 Graph Explanation:**

* The accuracy of the model on the training set is shown by **the blue line**.
* The accuracy on the validation set, which is used to assess the model's generalization on unseen data, is shown by **the orange line.**

**6.2.2 Key Observations:**

* The training accuracy and validation accuracy were approximately **52%** and **61%**, respectively, at the beginning of the first period.
* By the end of instruction, both accuracies had steadily increased to above **80%**.
* A good indicator that the model is not overfitting is that the validation accuracy frequently stayed somewhat higher than the training accuracy.
* Both curves exhibit some variations, particularly in the latter epochs. The scale and diversity of the dataset may be the cause of this, which is typical in real-world training.



**6.3 Loss Results**

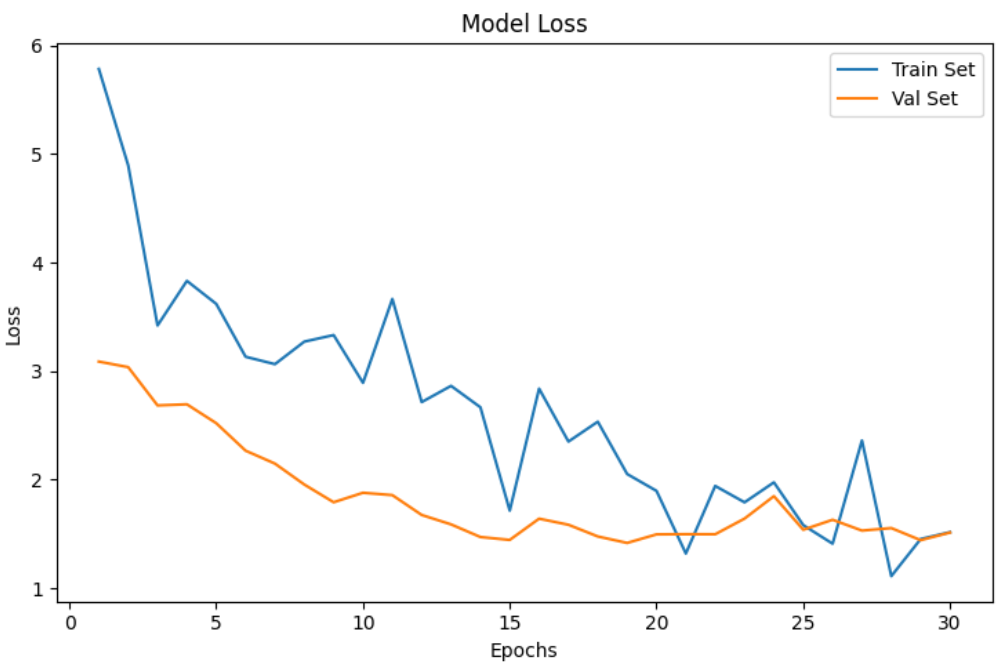
The graph below represents the model's loss for both the training and validation sets over the 30 training epochs.

**6.3.1 Graph Explanation:**

* The **blue line** represents the training set's loss, and the validation set's loss is indicated by the orange line.
* **Loss** is a measure that indicates how far the model's predictions differ from the actual values; the smaller the difference, the better.

**6.3.2 Key Observations:**

* The model had a **high loss** on the training set (about 5.8) at its beginning of training (epoch 1), while the validation loss was approximately 3.1.
* Training and validation loss both sharply declined as training went on, demonstrating the model's **successful learning**.
* The model had converged (reached a steady learning state) when both curves began to settle at about **epoch 20**.
* Throughout, the validation loss was less than the training loss, indicating that the model was not overfitting and that it can effectively generalize to unseen data.



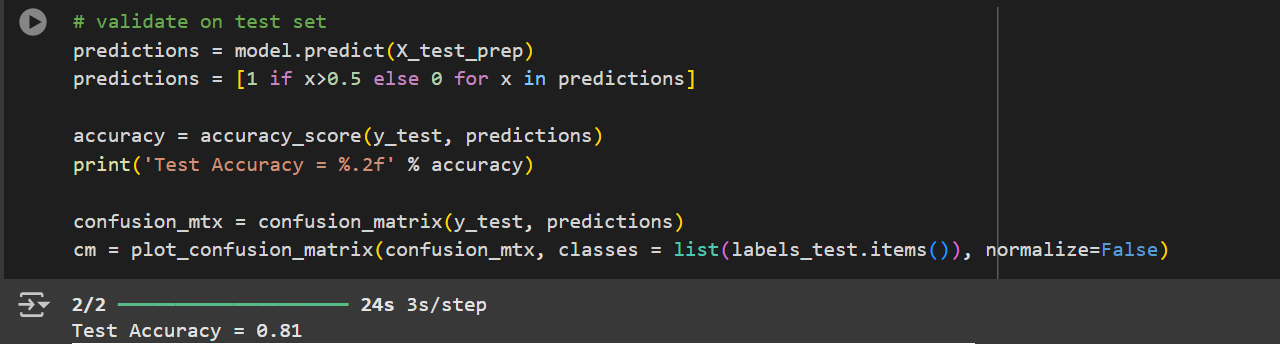
**CHAPTER 7**

**7. Model Evaluation on Validation Set**

After training the model, we evaluated its performance using the **validation dataset** to test how well it generalizes to new, unseen data.

**7.1.1 Accuracy**

Using a threshold of **0.5** to categorize outputs as either 0 or 1 (binary classification), the model generated predictions on the validation set. The results revealed:



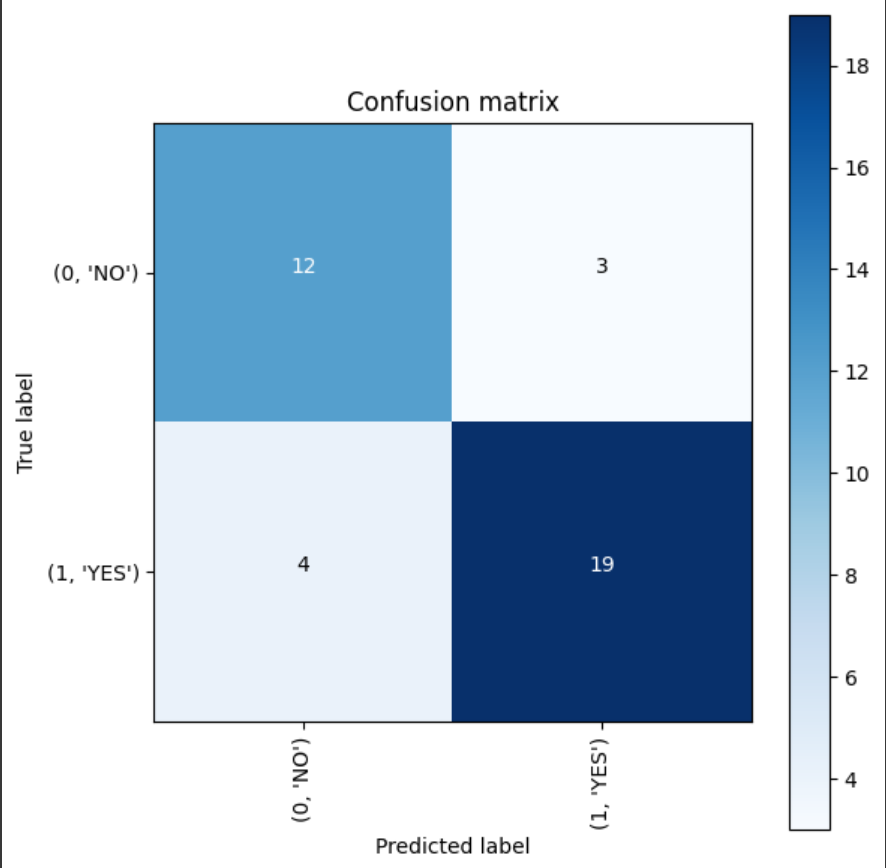
This indicates that the model has successfully trained to recognize patterns in the data and performs well on unseen examples, as seen by the fact that **82% of the validation samples were correctly classified**.

**7.1.2 Confusion Matrix**

The model's performance was further examined by creating a **confusion matrix**. It offers insights into the model's mistake areas and the proportion of accurate predictions:

* **True Positives (TP):** Correctly predicted positive class.
* **True Negatives (TN):** Correctly predicted negative class.
* **False Positives (FP):** Incorrectly predicted as positive.
* **False Negatives (FN):** Incorrectly predicted as negative.

The confusion matrix is visualized using a plotting function to clearly show the distribution of predictions across the actual classes.

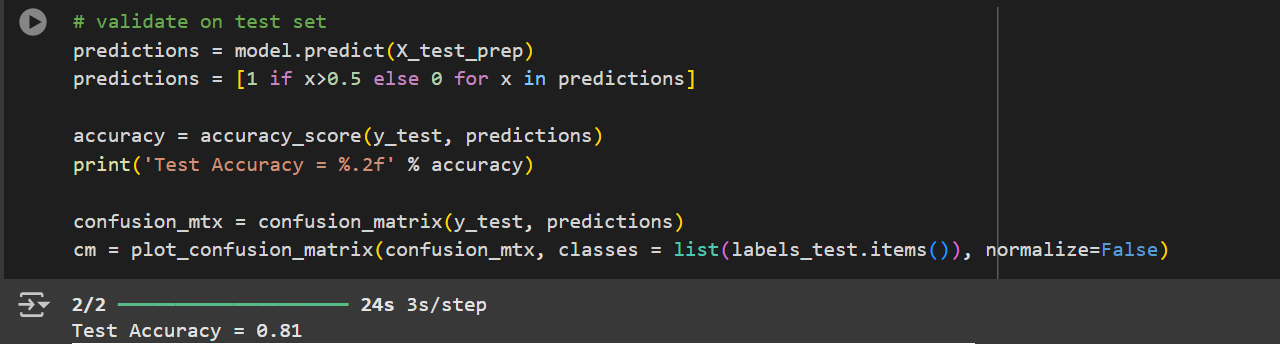


7.2. Model Evaluation on Test Set

Following performance validation, we performed a final assessment by simulating real-world performance on entirely unseen data using the **test dataset**.

7.2.1 Accuracy on Test Set

Using the pre-processed test dataset (X\_test\_prep), the model was used to predict results. Probabilities were transformed into binary classification results by thresholding predictions at **0.5**. The accuracy metric was then used to measure the performance:

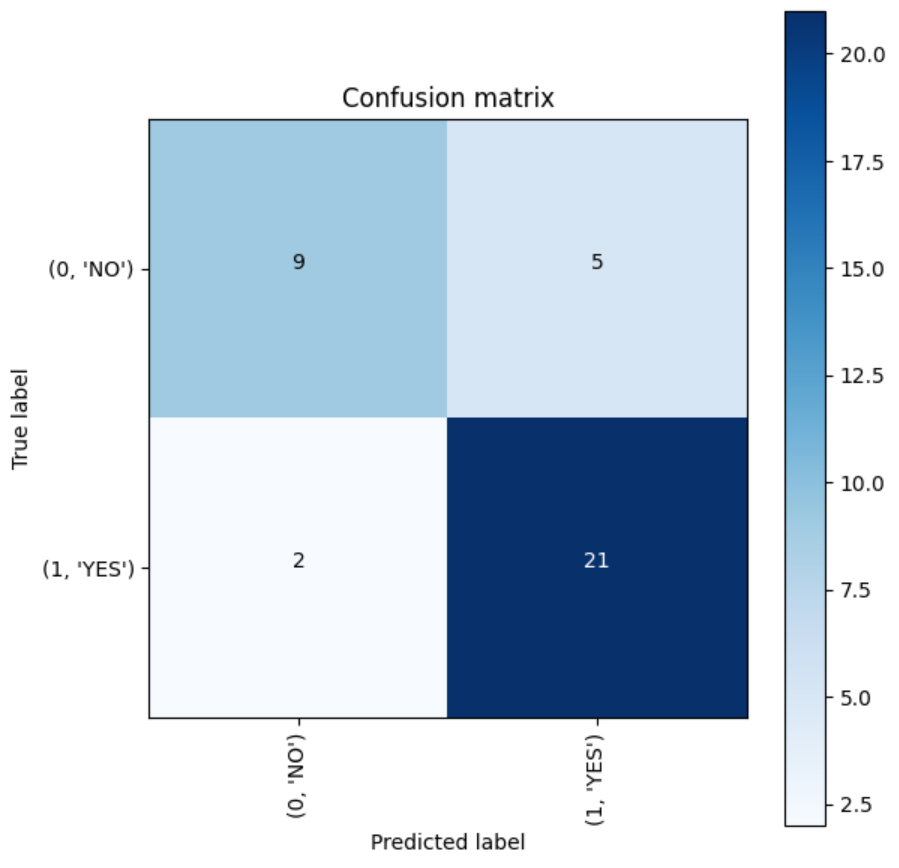


This indicates that the model performed well outside of the training and validation datasets, as evidenced by its 81% accuracy on the test set. When faced with new input, it performs consistently and reliably.

7.2.2 Confusion Matrix on test Set

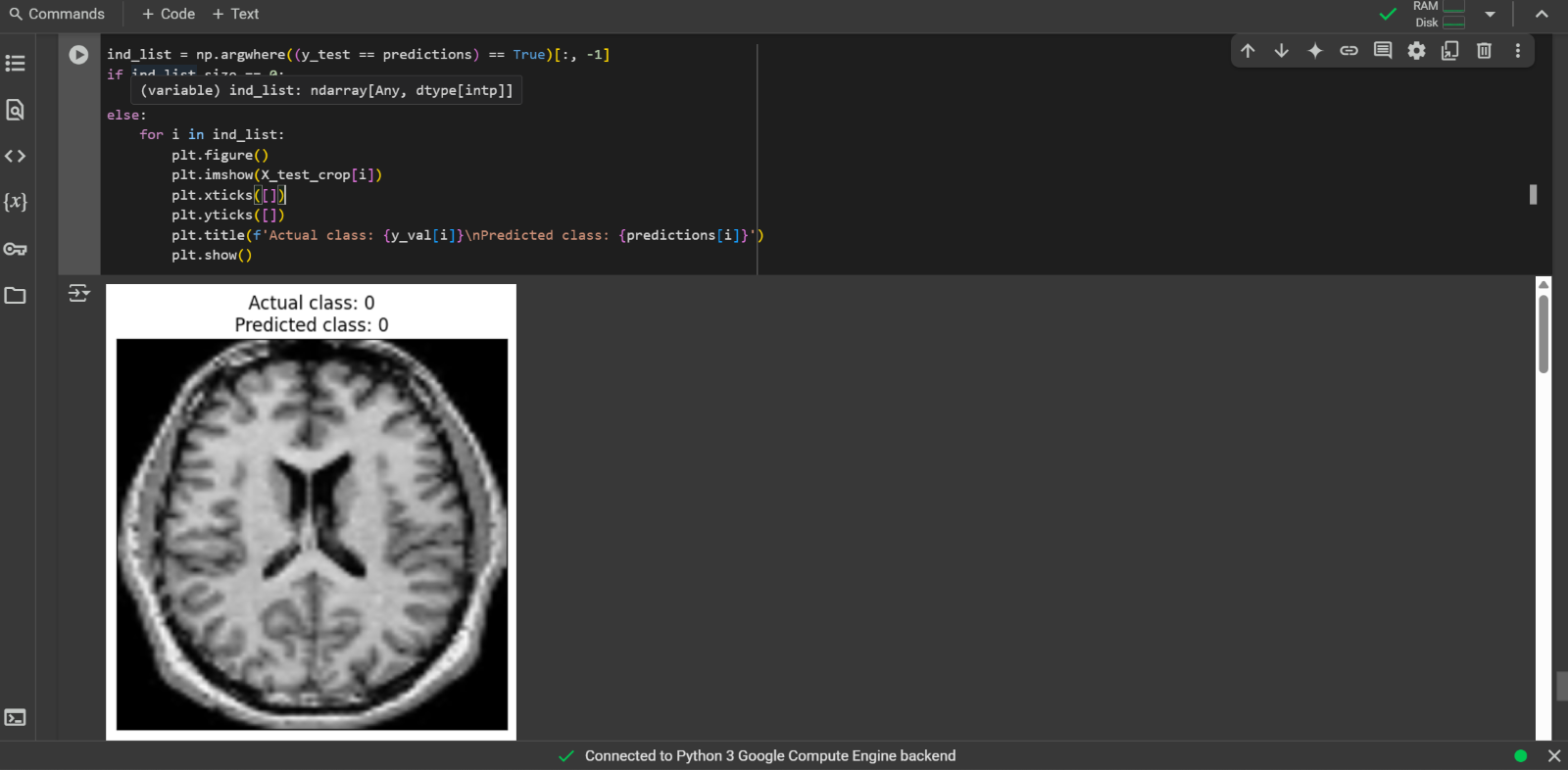
A **confusion matrix** was created to provide a detailed breakdown of prediction performance:

* **True Positives (TP):** Correctly predicted positive class
* **True Negatives (TN):** Correctly predicted negative class
* **False Positives (FP):** Incorrectly predicted positive class
* **False Negatives (FN):** Incorrectly predicted negative class



With the help of this matrix representation, one can evaluate the model's prediction balance, identify any bias toward a certain class, and highlight areas that might require improvement.

7.3 Misclassified Images Analysis



To gain deeper insight into the model’s predictions, we examined the test dataset's misclassified photos. This enables us to visually examine the instances in which the model produced inaccurate predictions and potentially comprehend the reasons behind them.

All test images where the predicted class differs from the actual class are identified by this code:

* This line locates the indices of **correctly classified** images (where prediction == ground truth).
* If ind\_list is empty, it means **all predictions were incorrect**—which is unlikely given the high

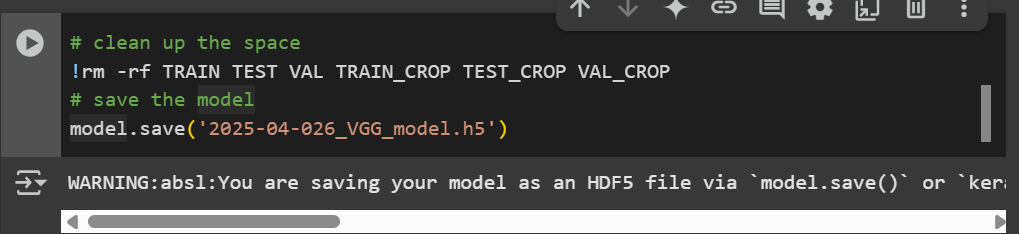
**CHAPTER 8**

**8.1 Saving the Trained model**

This line saves the trained model into a file named **2025-04-026\_VGG\_model.h5**. The model is stored in HDF5 format (.h5), which allows it to be:

* Reused later for testing or deployment
* Shared or loaded back for making predictions without retraining

This step is crucial for **model deployment** or continuing work in the future.



### **Conclusion**

The present project provided a preliminary brain tumor detection system with a pretrained VGG16 convolutional neural network architecture and achieved a test accuracy of about 82%. The dataset contained approximately 200 MRI images, and we preprocessed and augmented the MRI dataset with rotation, flipping and shifting to maximize model ability to generalize given the smaller dataset.

The VGG16 model's impactful deep ConvNet layers enabled the model to learn and extract many relevant hierarchical features appropriate for tumor classification, the necessary building blocks for separating tumor versus non-tumor images. A transfer learning model with pretrained VGG16 weights also showed faster convergence and better performance compared to training from scratch. The results of the testing phase were promising, but the limitations of dataset size and diversity of scans, lack of overfitting indicators, and the absence of clinical validation took clinical reliability off the table for some time. We suspect that data augmentation may have reduced some biases associated with overfitting, but additional efforts will be needed to understand how robust the model could be in such relatively atypical clinically use cases.

Future work therefore needs to expand the dataset, notably with larger volumes of diverse MRI scans, optimize hyper-parameters, and assess using a suite of regularization methods for overfit. In addition, extensive clinical testing and validation will be required to provide evidence for the model’s reliability as a diagnostic tool prior to introduction into healthcare workflow.

In conclusion, the results of this study validate the feasibility and potential of an automated brain tumor detection model using a VGG16-based CNN and data augmentation. It sets an important first step for creating AI-assisted decision tools with respect to neuro-oncology and ultimately offers a potential avenue to improve early detection rates, and patient outcomes.

### **References**

1. Kaggle. *Brain MRI Images for Brain Tumor Detection* <https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection>
2. Chollet, F. (2015). *Keras*. GitHub repository. <https://github.com/keras-team/keras>
3. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 265–283. [https://www.tensorflow.org](https://www.tensorflow.org/)
4. OpenCV. (n.d.). *Open Source Computer Vision Library*. Retrieved from <https://opencv.org/>
5. Brownlee, J. (2019). *How to Train a Convolutional Neural Network to Classify Photos*. Machine Learning Mastery. Retrieved from [https://machinelearningmastery.com](https://machinelearningmastery.com/)
6. Spotless Tech. (2023). *Brain Tumor Detection using CNN | Deep Learning Project* [Video]. YouTube. <https://youtu.be/90ElPbbhVWA?si=7z5hdTb4hp4Jv7Hr>