Course Name: Neural Networks Course code: AI303

Lecturer: Dr. Sahar Ghanem TA: Eng. Huda Hemdan



**MRI Tumar Recognition**

Pharos University in Alexandria Faculty of Computer Science & Artificial Intelligent

AI Department Fall 2023 – 2024

1. **Introduction**

The objective of this project is to assist healthcare professionals in efficiently identifying tumor types from Magnetic Resonance Imaging (MRI) images using deep learning techniques. Detecting brain tumors at early stages is critical for improved patient outcomes. Traditionally, manually analyzing the vast number of MRI images and diagnosing tumors is a tedious and error-prone task, which can delay proper medical treatment.

Additionally, the resemblance between normal brain tissue and tumor cells makes it challenging to accurately segment tumor regions. Therefore, there is a pressing need for an accurate and automated tumor detection method.

This project utilizes a dataset of MRI images to classify brain tumors into four categories: glioma, meningioma, pituitary tumors, and healthy brain tissue (no tumor). The dataset consists of a total of 7023 images, which are divided into three subsets: 5688 images for training, 632 for validation, and 703 for testing. The dataset's class distribution is as follows:

* **Glioma:** 1621 images (1321 for training and 300 for testing)
* **Meningioma**: 1645 images (1339 for training and 306 for testing)
* **Pituitary Tumors:** 1757 images (1457 for training and 300 for testing)
* No Tumor: 2000 images (1595 for training and 405 for testing)

By leveraging deep learning models, this project aims to automate the detection and classification of these tumor types, enabling more accurate diagnoses and improving patient care. With the use of

advanced computational intelligence techniques, this system will provide significant support to physicians in identifying brain tumors at early stages, contributing to better clinical outcomes.

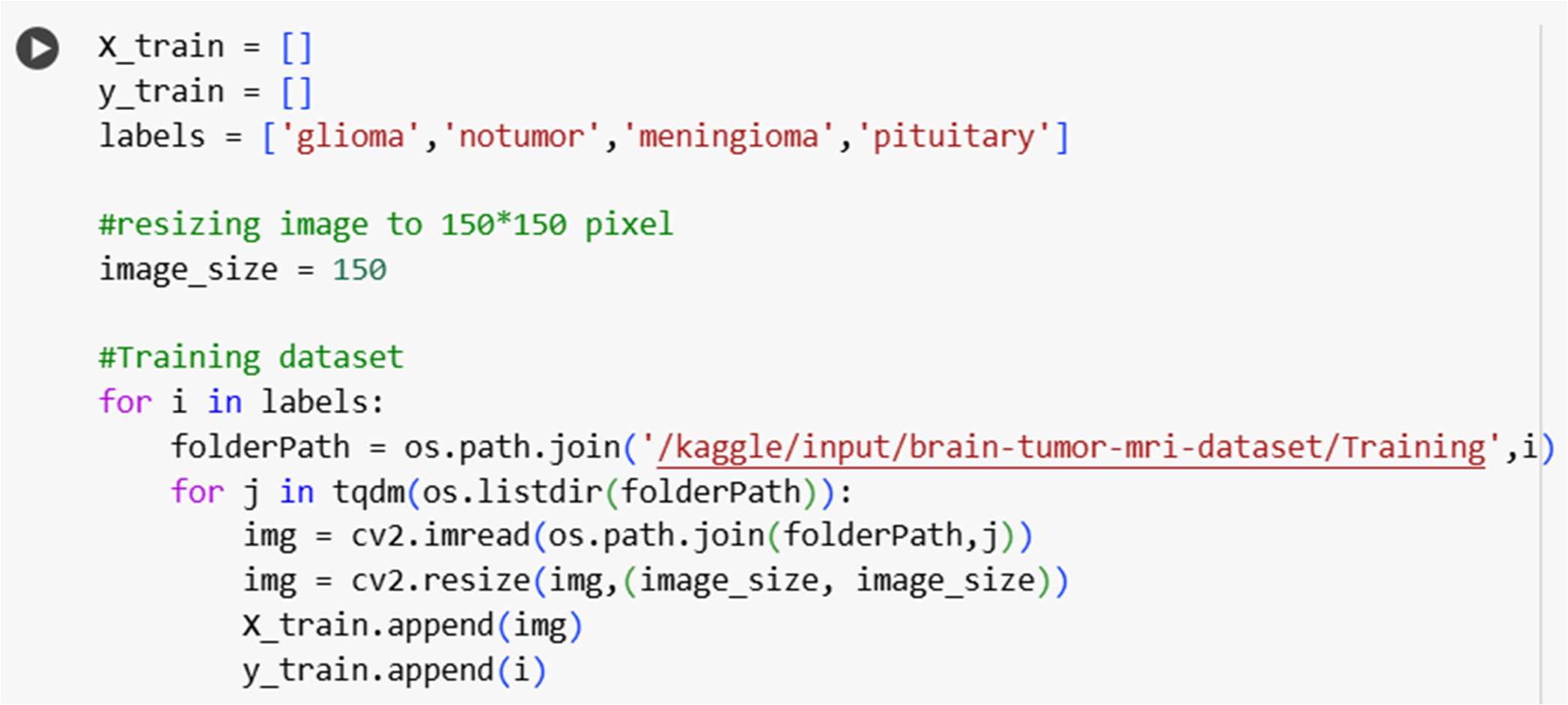
# Challenges in the dataset (The pre-processing phase)

### Image Resizing

Why Used: Convolutional Neural Networks (**CNNs**), require input images to be of a consistent size. If images of varying sizes feeded into a model, it will lead to shape mismatches and errors. By resizing all images to 150x150 pixels, ensure that each input image has the same dimensions, making it compatible with the model. Also, for efficiency that it Reduces the computational load by decreasing the number of pixels the model needs to process, speeding up training and inference. And finally, Helps manage memory usage, making it feasible to process large datasets.

**How Used**: image resizing is performed using the OpenCV function cv2.resize(img, (image\_size, image\_size)), where image\_size is set to 150. This is done for both the training and testing datasets. Each image is loaded and resized to 150x150 pixels before being added to the X\_train (or X\_test for testing) list. This ensures that all images have a consistent size before being fed into the model.

### Code Module:

****

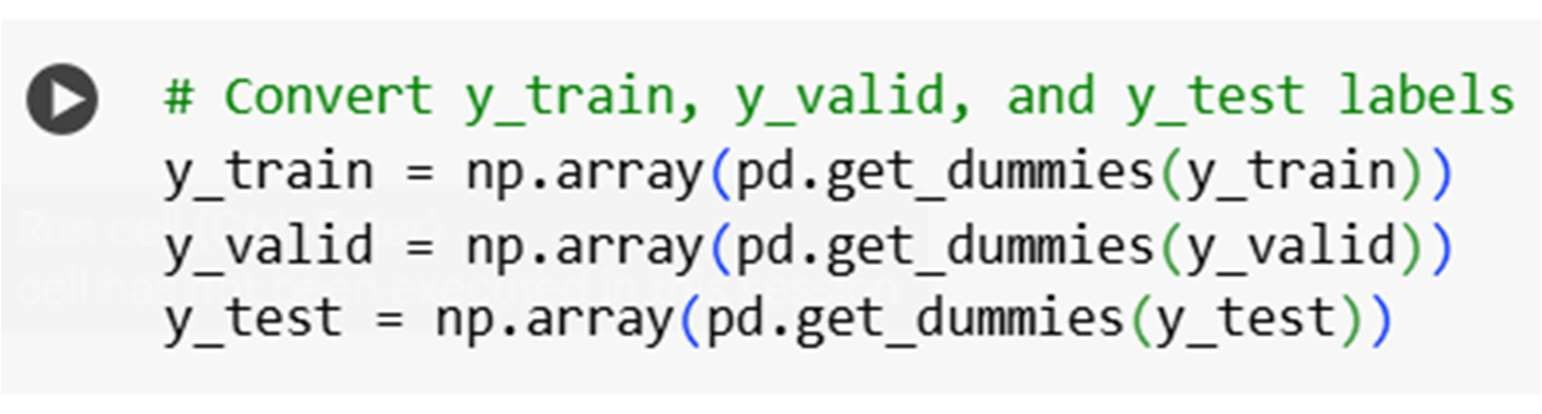
* + **One-Hot Encoding for Labels:**

Why Used: Since this is a multi-class classification task (with 4 classes), the labels need to be one-hot encoded. This ensures that the neural network can handle multiple classes by outputting one probability per class, with the highest probability corresponding to the predicted class.

How Used: The pd.get\_dummies() function was used to one-hot encode the labels. For

instance, the class "2" would be converted into [0, 1,0 , 0] for a 4-class classification.

### Code Module:



* + **Normalization:**

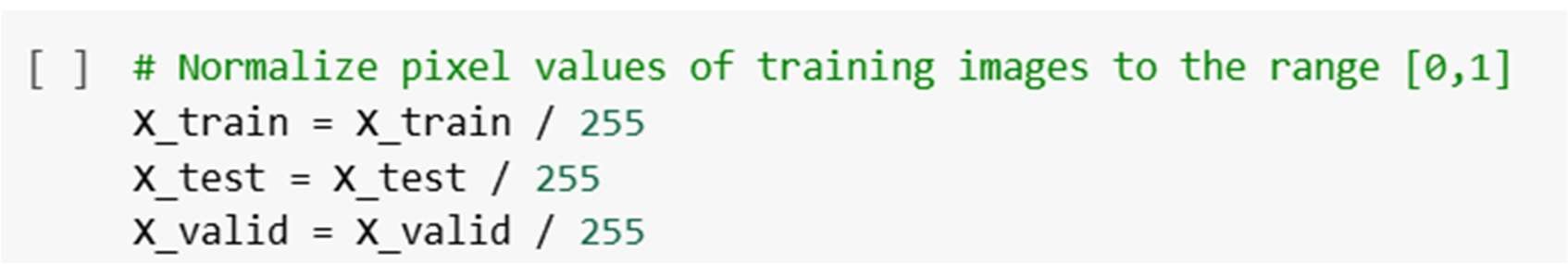
**Why Used:** The images in the dataset were of varying sizes and resolutions, which could negatively affect the performance of the neural network. Therefore, preprocessing steps like resizing and normalization were used to standardize the images and improve training stability.

### How Used:

The images were resized to a fixed size (150x150x3pixels), ensuring consistency in input size.

Pixel values were normalized to the range [0, 1] by dividing by 255.

Code Module:

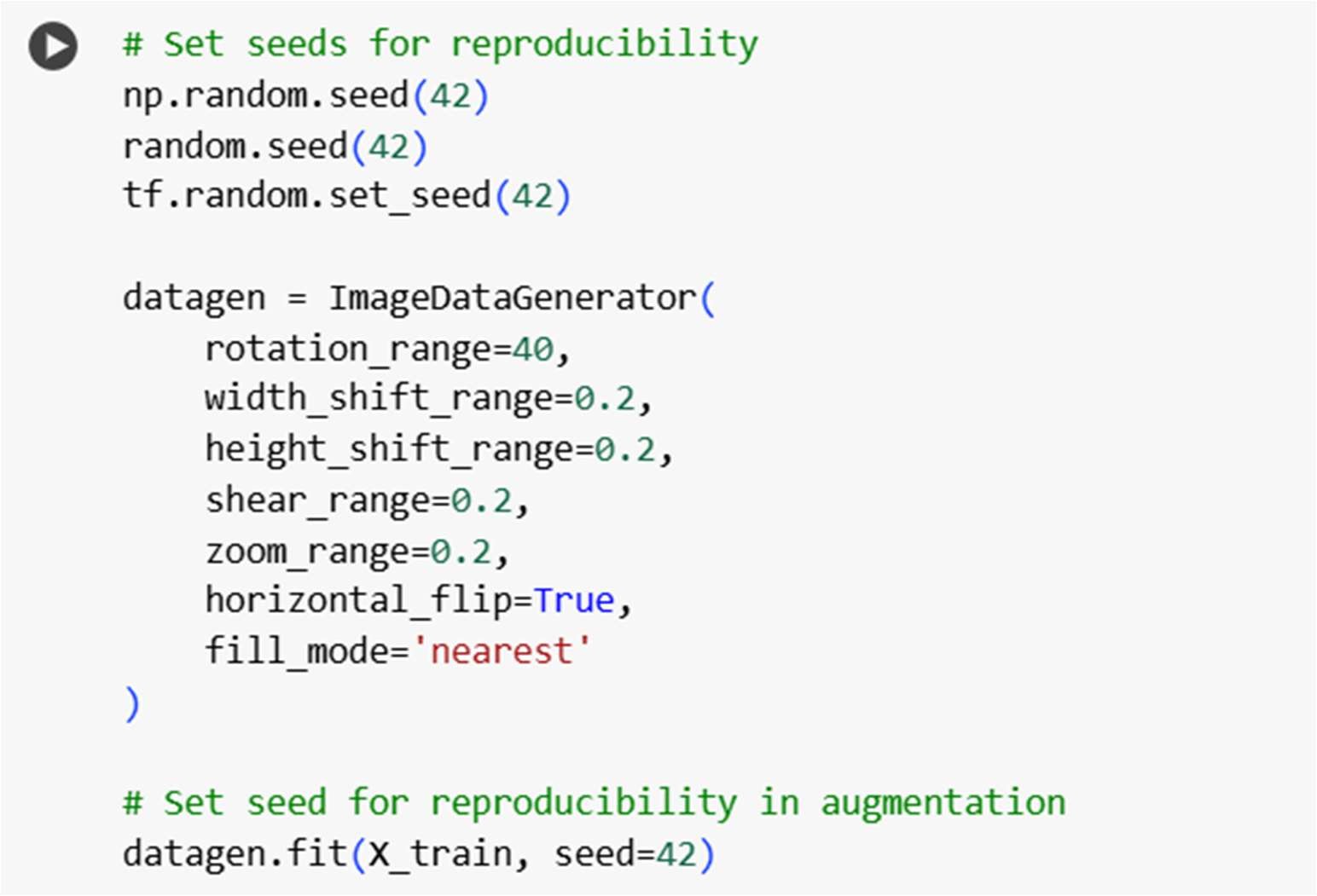


### data Augmentation:

**Why Used:** Data augmentation was employed to artificially increase the size of the training dataset and help the model generalize better, preventing overfitting. Since the original dataset was relatively small, augmentation was essential to improve model performance.

**How Used:** The ImageDataGenerator from **Keras** was used to apply transformations like rotation, zoom, horizontal flipping, and shifting to each image. These transformations increase the diversity of the images and help the model learn to recognize features under different conditions.

**Code module:**



# processing phase

### Convolutional Neural Networks (CNNs) Architecture:

#### Convolutional Layers:

* 1. These layers are the core building blocks of CNNs. A convolutional layer applies a set of learnable filters (kernels) to the input image to extract various features such as edges, textures, and patterns.
  2. Each filter slides over the input image, performing element-wise multiplication and summing the results to produce a feature map.
  3. Convolution preserves the spatial relationship between pixels by learning image features.

#### Activation Function:

* 1. After the convolution operation, an activation function, typically ReLU (Rectified Linear Unit), is applied to introduce non-linearity into the model. This allows the network to learn complex patterns.

#### Pooling Layers:

* 1. Pooling layers are used to reduce the spatial dimensions of feature maps while retaining the most critical information.
  2. Common pooling techniques include Max Pooling (selecting the maximum value from a region) and Average Pooling (computing the average value).
  3. Pooling reduces computational complexity and helps prevent overfitting.

#### Fully Connected Layers:

* 1. Fully connected layers connect every neuron in one layer to every neuron in the next layer.
  2. These layers act as a classifier, mapping the learned features to the output classes.
  3. The final fully connected layer typically uses a softmax activation function for multi-class classification tasks.

#### Dropout Layers (Optional):

* 1. Dropout is a regularization technique used to prevent overfitting by randomly deactivating a subset of neurons during training.
  2. This encourages the network to learn more robust and generalized features.

# Comparisons

### Neural Network Architectures Used: VGG16 and ResNet101

VGG16 and ResNet101 architectures both widely recognized convolutional neural networks (CNNs) known for their effectiveness in image classification tasks. Below is an explanation of these architectures and their application in our project:

## First: VGG16

Visual Geometry Group model is an architecture that use 16 layers in its learning, these layers are:

#### Convolutional Layers:

* 1. VGG16 employs small 3x3 filters throughout the convolutional layers.
  2. The depth of the network increases progressively, starting from 64 filters and doubling after every max-pooling operation (e.g., 64, 128, 256, 512).

#### Max-Pooling Layers:

* 1. Max-pooling with a 2x2 filter and stride of 2 is applied after every two or three convolutional layers.
  2. This reduces the spatial dimensions while retaining the most salient features.

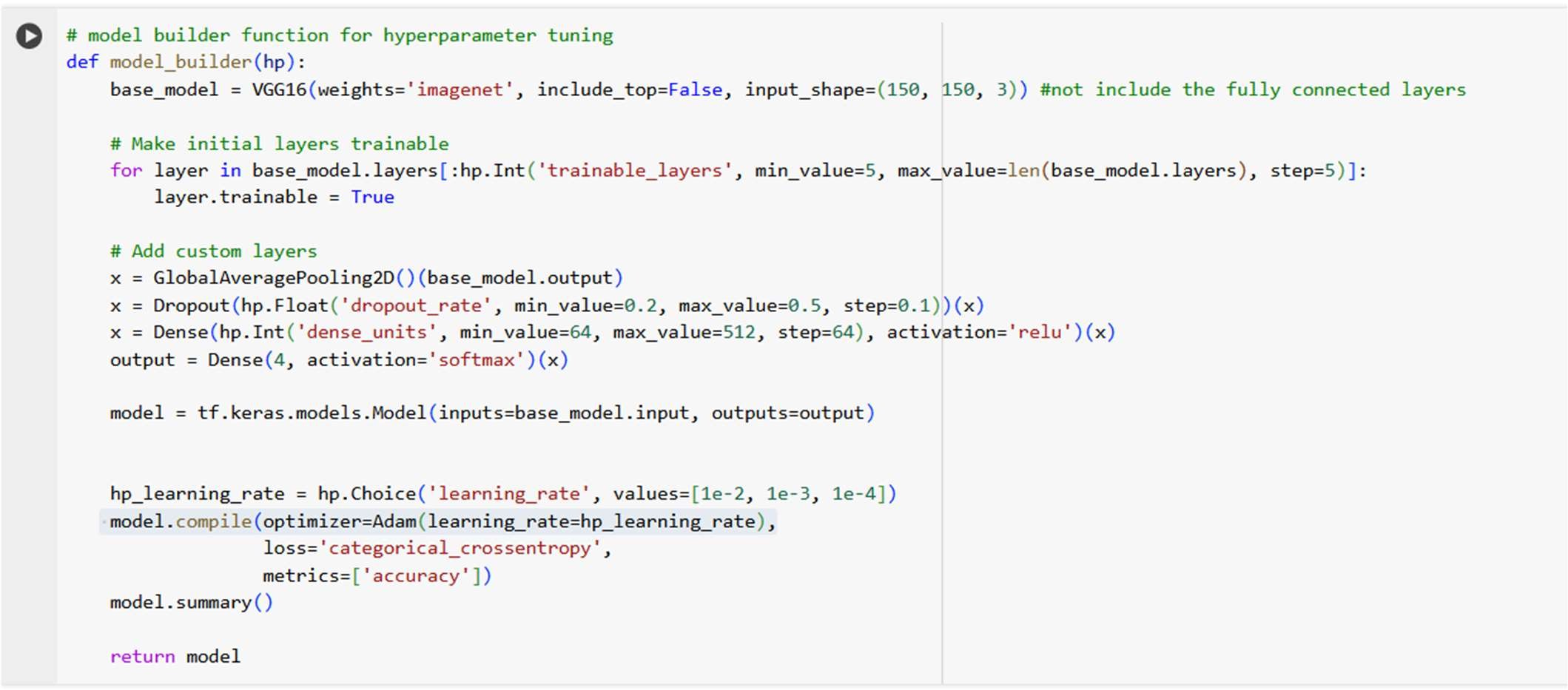
#### Fully Connected Layers:

* 1. At the end of the convolutional stack, four fully connected layers are used.
  2. A three layers, followed by a softmax output layer for classification.

#### Activation Function:

The ReLU (Rectified Linear Unit) activation function is used in all hidden layers, introducing non- linearity and improving learning efficiency.

# Practically:



## Second: ResNet 101:

Residual Network model is an architecture that use 101 layers

1. Residual Connections:
   1. ResNet101 uses skip connections (or shortcuts) to bypass one or more layers, addressing the vanishing gradient problem and enabling deeper network architectures.
2. Bottleneck Design:
   1. Each residual block employs a bottleneck structure with three layers: 1x1, 3x3, and another 1x1 convolution. This design reduces computational complexity while maintaining performance.
3. Global Average Pooling:
   1. Similar to VGG16, ResNet101 replaces fully connected layers with global average pooling to reduce overfitting and improve generalization.

**Practically:**



* *Parameters and Hyper parameter used:*
  1. Parameters:

Both architectures used pre- trained weights of ImageNet dataset, this dataset contain millions of images and weights is trained on these images.

Total trained parameter in Vgg16: 14,847,044 Total non\_trainable parameter in Vgg16: 0

Total trained parameter in ResNet101: 43,472,580 Total non\_trainable parameter in ResNet101: 105,344

* 1. Hyper Parameters:

T: number of trials

TL: number of trainable layers D: Dropout units

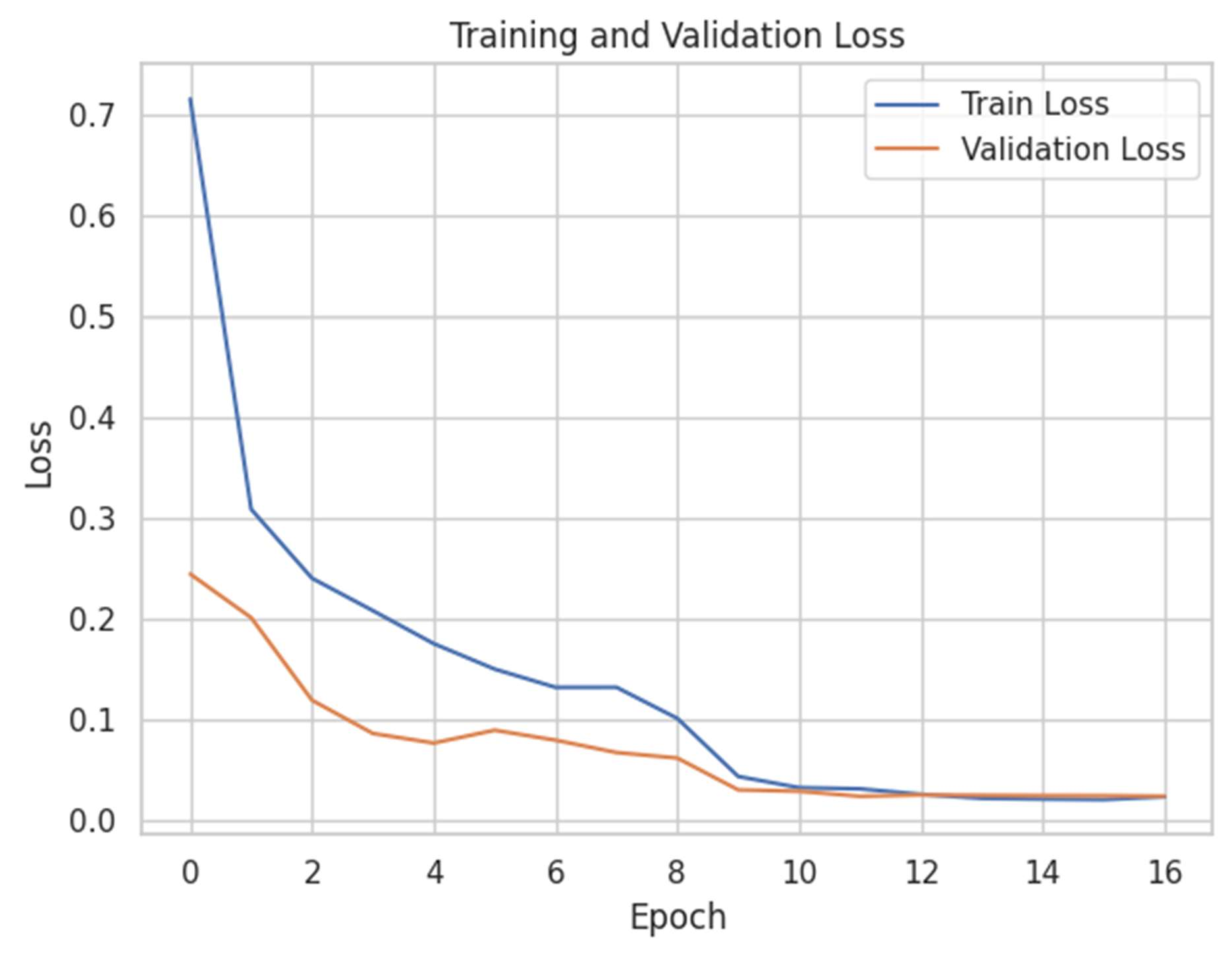
U: Dense Units LR: Learning Rate E: Epochs

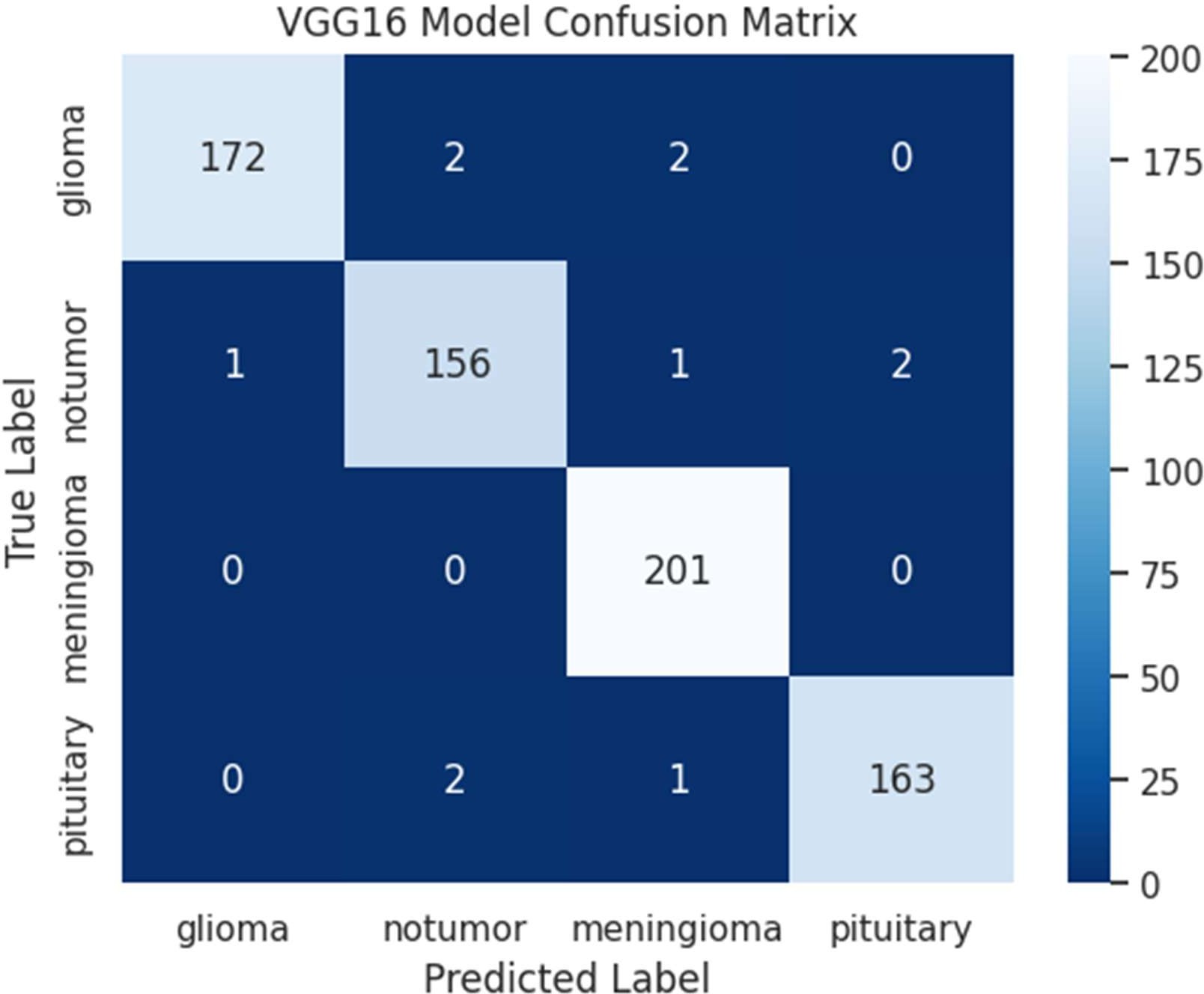
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **T** | **TL** | **D** | **U** | **LR** | **E** |
| 1 | 15 | 0.4 | 192 | 0.001 | 2 |
| 2 | 10 | 0.4 | 192 | 0.001 | 2 |
| 3 | 10 | 0.3 | 320 | 0.01 | 2 |
| 4 | 10 | 0.2 | 64 | 0.0001 | 2 |
| 5 | 10 | 0.2 | 64 | 0.0001 | 2 |
| 6 | 10 | 0.2 | 64 | 0.0001 | 2 |
| 7 | 10 | 0.2 | 64 | 0.01 | 2 |
| 8 | 10 | 0.2 | 64 | 0.0001 | 2 |
| 9 | 15 | 0.3 | 384 | 0.0001 | 2 |
| 10 | 15 | 0.3 | 384 | 0.0001 | 2 |
| 11 | 15 | 0.3 | 384 | 0.0001 | 2 |
| 12 | 15 | 0.3 | 384 | 0.0001 | 2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 13 | 15 | 0.3 | 384 | 0.0001 | 4 |
| 14 | 15 | 0.3 | 384 | 0.0001 | 4 |
| 15 | 15 | 0.3 | 384 | 0.0001 | 4 |
| 16 | 15 | 0.3 | 384 | 0.0001 | 4 |
| 17 | 15 | 0.3 | 384 | 0.0001 | 4 |
| 18 | 10 | 0.2 | 64 | 0.0001 | 10 |
| 19 | 10 | 0.2 | 64 | 0.0001 | 10 |
| 20 | 10 | 0.2 | 64 | 0.0001 | 10 |
| 21 | 10 | 0.2 | 64 | 0.0001 | 10 |
| 22 | 10 | 0.2 | 64 | 0.0001 | 10 |
| 23 | 10 | 0.2 | 64 | 0.0001 | 10 |
| 24 | 10 | 0.2 | 64 | 0.0001 | 10 |
| 25 | 10 | 0.2 | 64 | 0.0001 | 10 |
| 26 | 15 | 0.2 | 128 | 0.0001 | 10 |
| 27 | 15 | 0.2 | 128 | 0.0001 | 10 |
| 28 | 15 | 0.2 | 128 | 0.0001 | 10 |
| 29 | 15 | 0.3 | 128 | 0.0001 | 10 |
| 30 | 15 | 0.3 | 128 | 0.0001 | 10 |

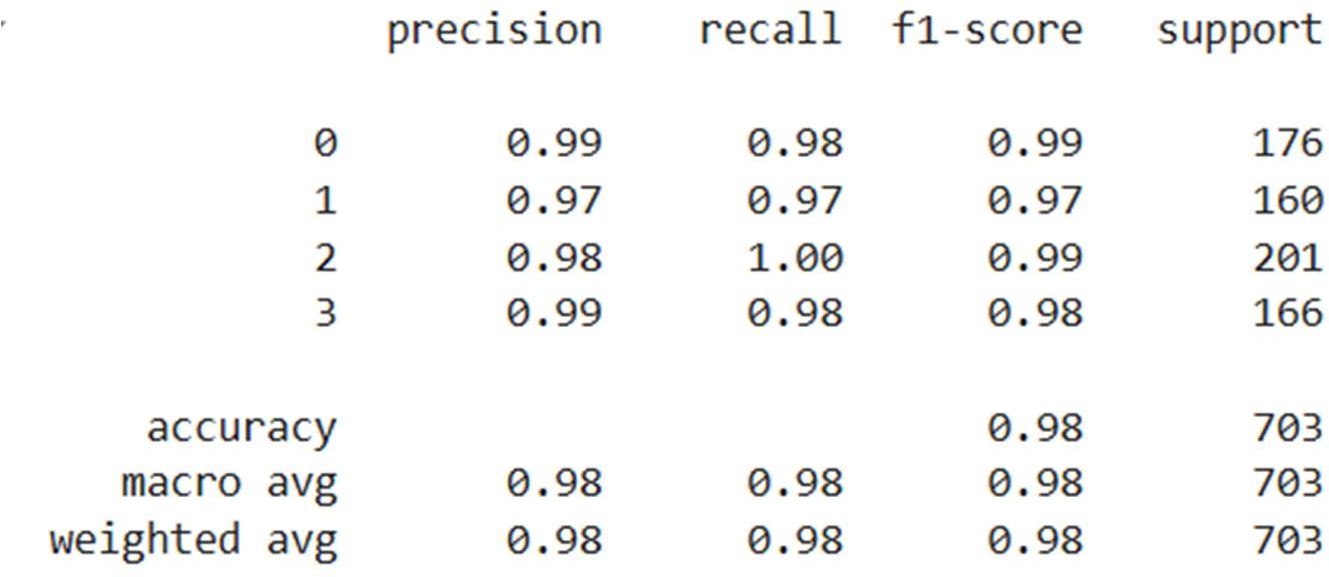
# Results:

## Results of VGG16:

* ***Loss:***
  + ***Confusion Matrix:***

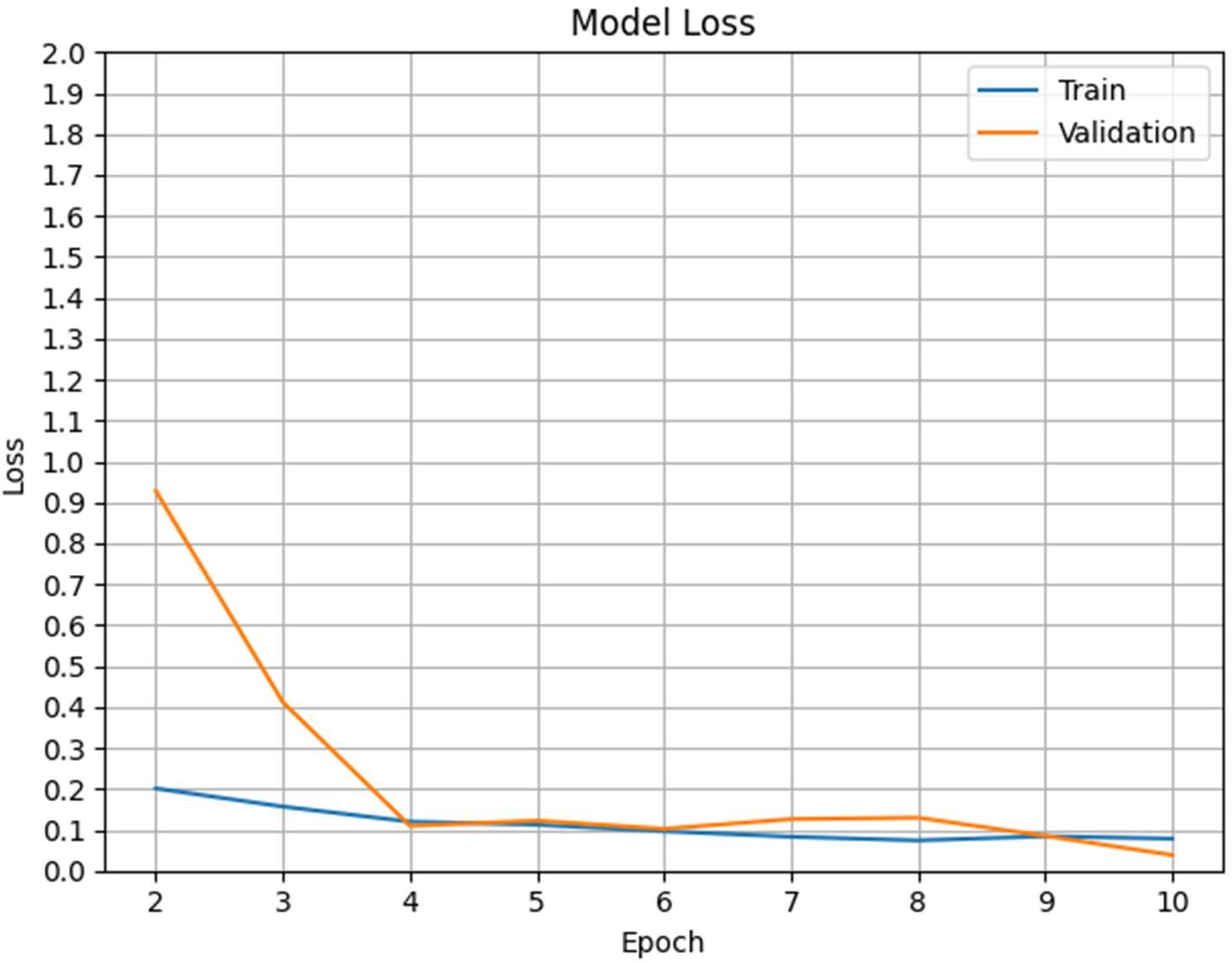


* + - ***Metrics:***

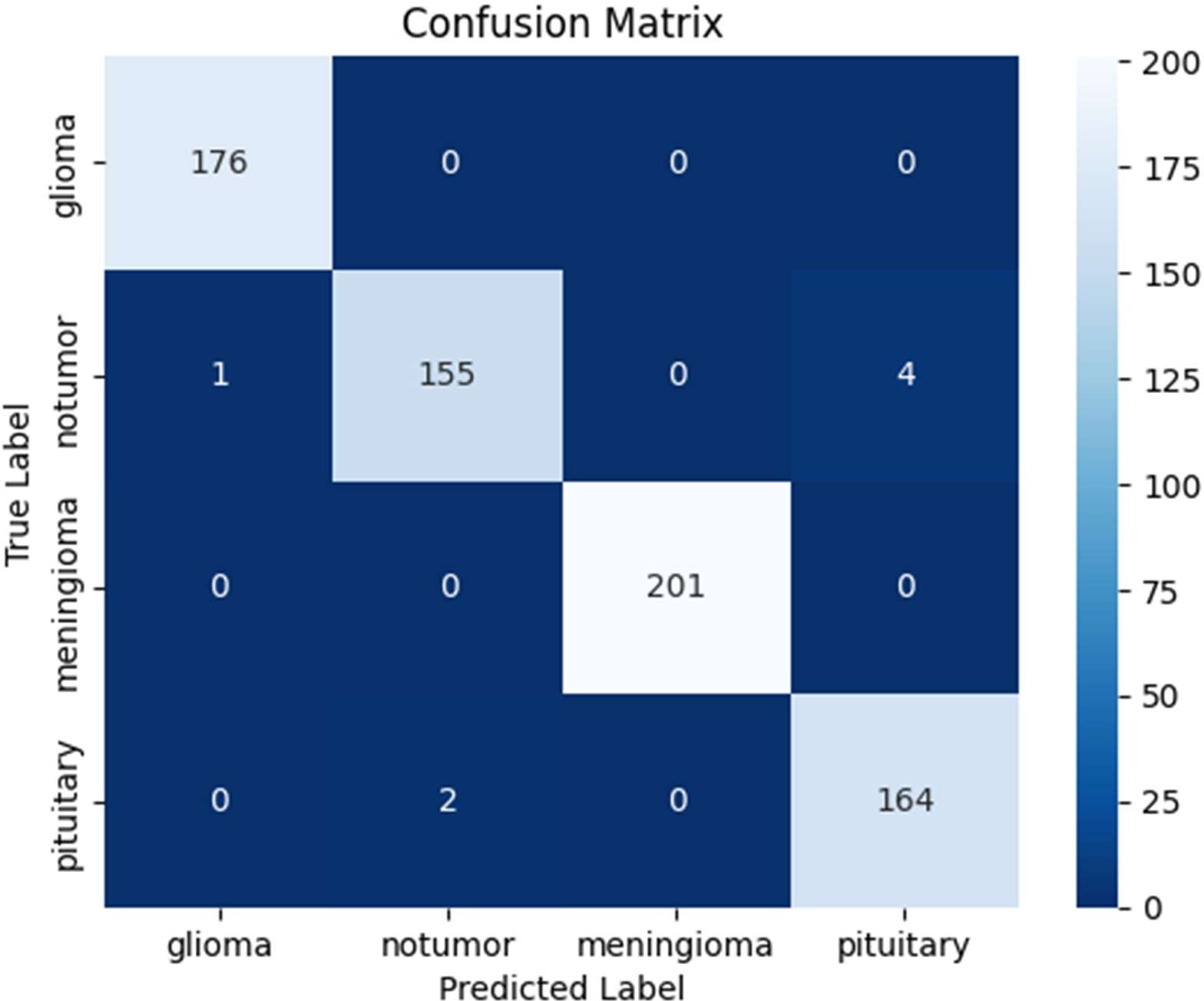
******

## Results of RESNET101:

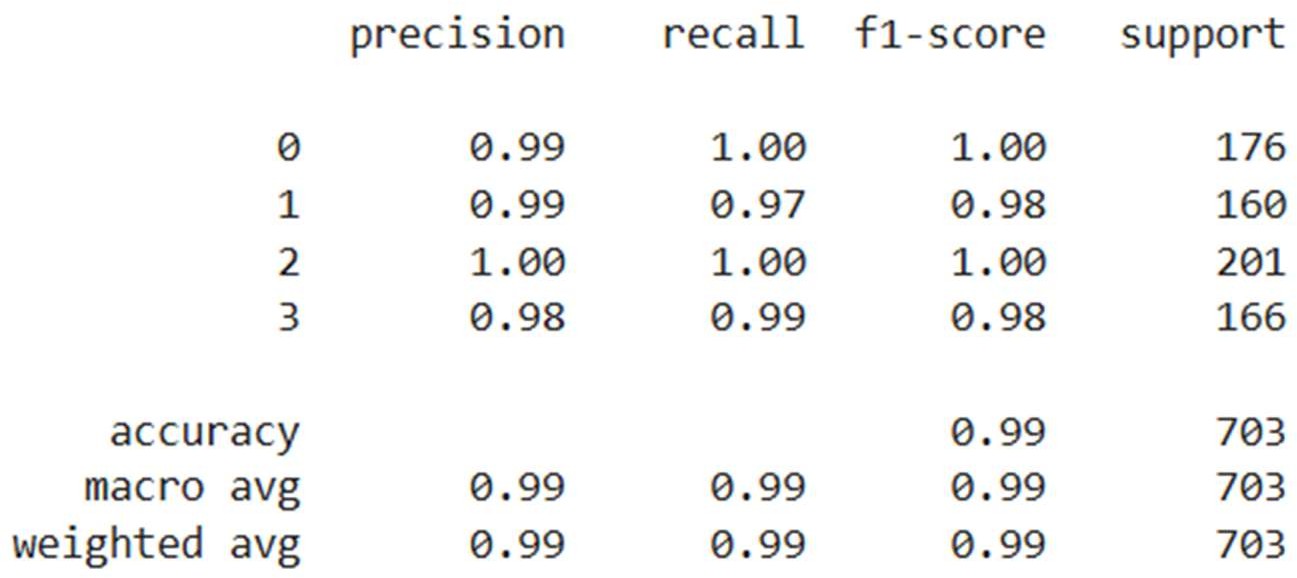
* + - * ***Loss:***



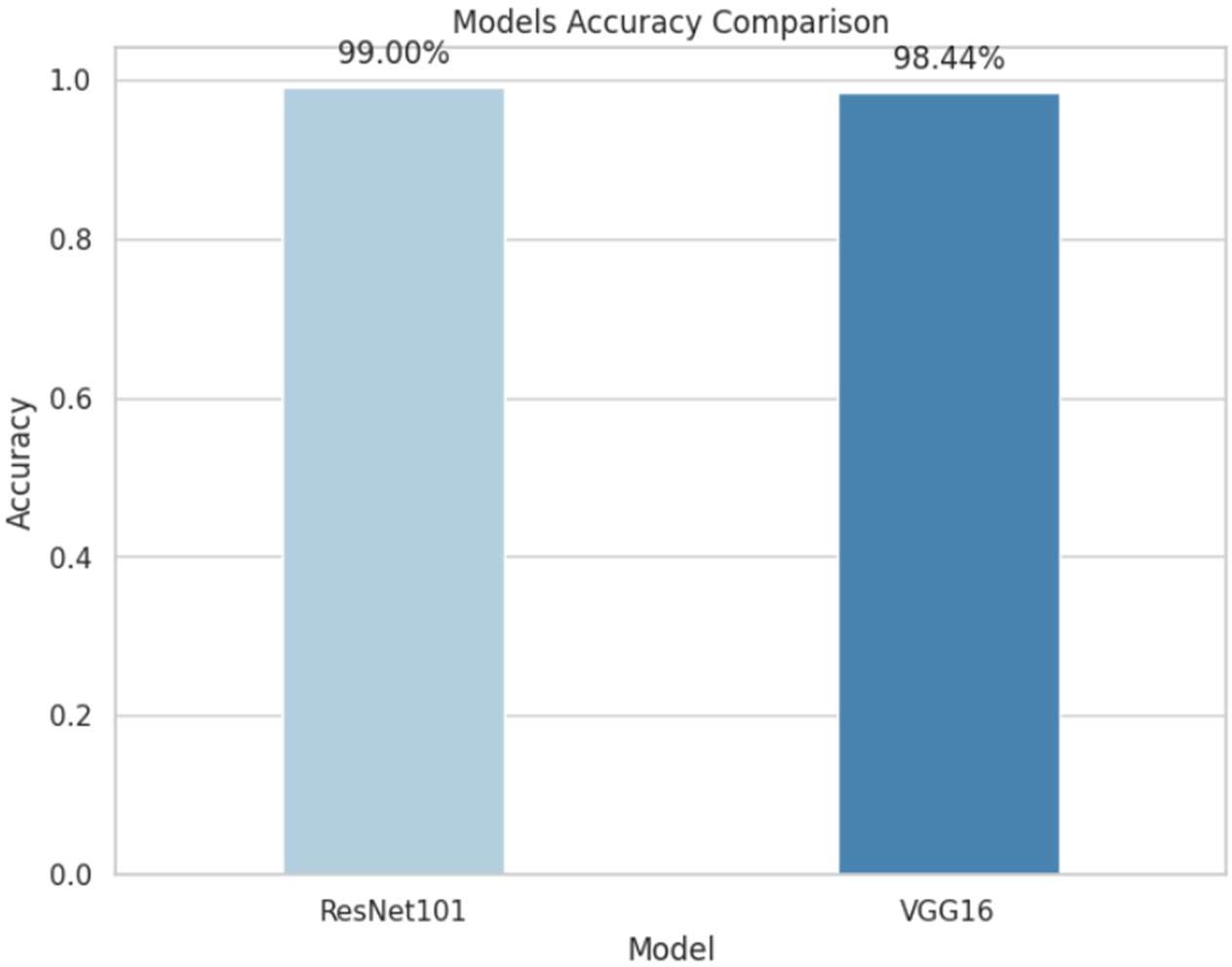
* + ***Confusion Matrix:***

******

* + - ***Metrics:***



## The Accuracy of both:



* So, we can realize that ResNet101 model performance better than Vgg16 model.

# Future Plan

* This MRI tumor classification model has the potential for ongoing refinement and integration into real-world healthcare applications. Some possible future directions and enhancements include:

#### Expanded Dataset and Tumor Types

Expanding the dataset to include more diverse MRI images and additional tumor types could enhance the model’s applicability across different tumor classifications and improve diagnostic flexibility.

#### Validation on Real-World Clinical Data

Test the proposed models on real-world clinical MRI data.

Compare performance with experimental procedures to assess practical medical applicability.

#### Continuous Training with Fresh Data

Setting up a pipeline for continuous training with anonymized data from new cases would help keep the model updated, adapting to trends and variations over time and enhancing accuracy in a production setting.

#### Transfer Learning from Advanced Architectures

Exploring advanced architectures, such as EfficientNet or Vision Transformers, could lead to performance gains in both accuracy and inference speed. This may be particularly beneficial for resource-constrained clinical environments.

#### Integration with Clinical Systems

Developing integrations with clinical management systems (e.g., PACS) could allow medical professionals to leverage this model directly in their workflows, potentially reducing diagnosis time and supporting early intervention.

#### Real-Time Prediction and Mobile Support

Implementing real-time prediction capability and deploying the model on mobile or edge devices could expand accessibility, allowing clinicians to use the model in remote or low- resource areas where immediate diagnostic assistance is needed.

#### Application to Other Imaging Modalities

Extend the approach to different medical imaging types, such as:

* + X-rays
  + CT scans
  + Ultrasounds

Evaluate the model's effectiveness in various medical imaging applications.

#### Small Lesion Detection Improvements

Address challenges in detecting small lesions by:

* + Synthetic Image Generation: Create synthetic datasets using medical expert knowledge.
  + Ensemble Models: Combine predictions from multiple models trained on different data subsets.
  + Expert Involvement: Include medical experts in the annotation process to prioritize difficult cases.

By implementing these improvements, the model could become a more reliable, accessible, and valuable tool in clinical practice, assisting healthcare providers in tumor classification and early intervention strategies.

# References

*Understanding and visualizing ResNets | by Pablo Ruiz | Towards Data Science*

*The Power of VGG16: A Deep Dive into One of the Most Influential Neural Networks in Image Recognition - Wisdom ML*

*https://keras.io/api/callbacks/*

*https://keras.io/keras\_tuner/api/tuners/base\_tuner/#tuner-class*

*https://keras.io/api/applications/vgg/*

*https://keras.io/api/applications/resnet/*