**UNIVERSITY OF BRIGHTON**

**MODULE TUTOR: - Dr Gulden Uchyigit**

**MODULE CODE: - CI642**

**MODULE TITLE: - Advanced AI**

**Assessment title: Advanced AI Coursework**

**STUDENT ID: - 21839630**

**SUBMISSION DATE: - 17/01/2024**

**Brain tumor classification with CNN Model**

Table of Contents

[1.Introduction 3](#_Toc156393862)

[1.1 Dataset 3](#_Toc156393863)

[1.2 project goals 3](#_Toc156393864)

[1.3 AI pipeline 4](#_Toc156393865)

[2.Data preparation 5](#_Toc156393866)

[2.1 Image loading and conversion 5](#_Toc156393867)

[2.2 Data Augmentation and Transformation 5](#_Toc156393868)

[2.3 Dataset Splitting 6](#_Toc156393869)

[2.4 Data Frame Construction 6](#_Toc156393870)

[3.MODEL BUILDING 7](#_Toc156393871)

[3.1 Convolutional Neural Networks 7](#_Toc156393872)

[3.2 Model’s Architecture and design 8](#_Toc156393873)

[3.3 Loss Function and Optimization 9](#_Toc156393874)

[3.4 Training Loop and Backpropagation 9](#_Toc156393875)

[3.5 Validation and Testing 10](#_Toc156393876)

[4.Evaluation Metrics 12](#_Toc156393877)

[4.1 Training and Validation Metrics 12](#_Toc156393878)

[4.1.1 Training Loss and Validation Loss 12](#_Toc156393879)

[4.1.2 Training Accuracy (TA) and Validation Accuracy (VA) 13](#_Toc156393880)

[4.2 Test evaluation Metrics 14](#_Toc156393881)

[4.2.1 Confusion Matrix 14](#_Toc156393882)

[4.2.2 Classification Report 16](#_Toc156393883)

[4.3 Prediction of images using Gradio 17](#_Toc156393884)

[Conclusion 18](#_Toc156393885)

[Reference 19](#_Toc156393886)

# 1.Introduction

The motivation for this project is the development and application of CNNs. CNNs shows the value of highly remarkable success in image classification, which makes them relevant for classifying and analyzing medical field dataset. This gives an opportunity for the doctors to have an early understanding and detection of any disease or any harmful condition of the patient in the early stage. This project helps in integrating with healthcare companies for workflow and ensuring the consistency of Brain tumor analysis .

## 1.1 Dataset

The data set used for this project is the MRI images of Brain tumor which has 44 classes. Each image of the data set has its own characteristics such as tumor presence, tumor types and other important characteristics. They cover a wide range of brain tumor types which are gliomas, meningiomas, pituitary adenomas and others, showcasing the dataset's depth in representation. They might go further in classifying sub-types within the tumors. This level of enhancement, the dataset's value for more research and model development. All the labels are imbalanced.

A graph of a number of blue bars

Description automatically generated with medium confidence

Since it’s a large dataset, it would be efficient for using the CNN model to classify the types of tumors present and beneficial for training and generalizable machine learning models.

## 1.2 project goals

Brain tumor is complex and has different types and numerous sub types, detecting of the tumor in early stage would be an effective for the treatment. So, this project goal is to employ the CNN model train and test it for classification of the tumor. CNN opted for this project because it’s not like traditional approaches which depend on feature engineering rather it would automatically learn relevant features form the medical images.

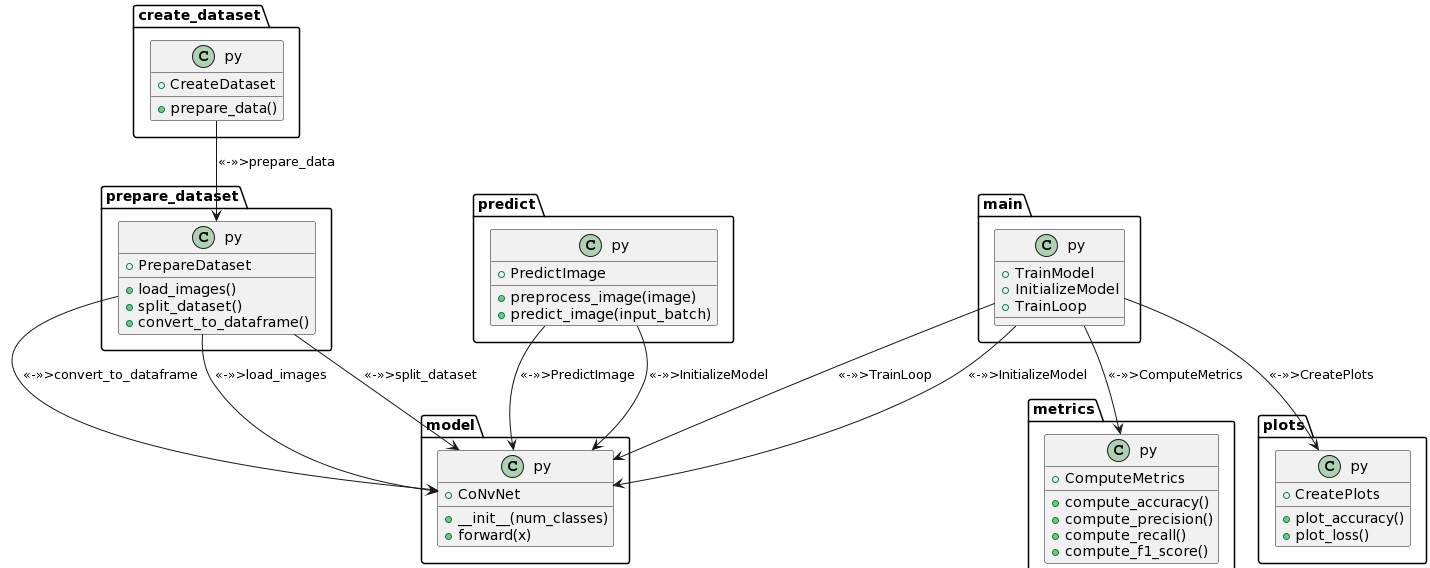
The training approach for this project involves not using the pre-trained model but rather developing the model from scratch for the desire of fine tuning the model and unique characteristics of the images of the dataset. Furthermore, we have used Fast API for deployment. Which is used interaction surface with model for the prediction.

## 1.3 AI pipeline

The following table describes key components in the project and shows the modules purpose and their functions: -

|  |  |  |
| --- | --- | --- |
| **MODULE** | **PURPOSE** | **FUNCTIONALITY** |
| create\_dataset.py | Creates a custom dataset class for managing the medical image dataset. | Defines a PyTorch dataset class. It reads the dataset, performs data splitting into training, testing, and validation sets, and then applies it to the image and necessary transformations occurs. |
| prepare\_dataset.py | Prepares the dataset for training, testing, and validation. | Goes through folders, collects class names, file paths, and class labels and then converts this information into a DataFrame. |
| model.py | Defines the Convolutional Neural Network (CNN) model for cancer classification. | Implements a CNN (CoNvNet) including three convolutional layers, max-pooling, and fully connected layers also includes methods for the forward pass. |
| main.py | Organises the training and evaluation of the CNN model. | Initializes the Cancer Classifier class that manages model training, validation, and testing. The functions used is data loaders, defined model and optimization. |
| metrics.py | Computes and evaluates various metrics on the model's predictions. | reads the test data, computes the F1 score, recall, accuracy, and precision, and creates a confusion matrix and a classification report. The CSV files holds the results. |
| plots.py | Creates plots to visualize the training and validation process. | reads the training performance metrics and plots the training and validation accuracy against the epochs of loss. |
| predict.py | Uses the trained model for making predictions on new images. | defines the predict image function, which loads the training model, pre-processes the input image, and predicts the class label. To create a basic interface for interacting with the model, Gradio is used. |

* For a clear understanding the UML flowchart below shows the connection between the modules.



# 2.Data preparation

## 2.1 Image loading and conversion

The raw images of the dataset are not in a proper format for the usage of training the model. The data should be involved in converting and loading the images to suitable format for training. For loading the image, a method **‘getitem’** which is in the ‘**CreateDataset**’ class is used for processing individual images based on it file path.

The function from the python imaging library is used for the reading the images and method ‘convert("RGB")’ is makes sures that the images are converted into the RGB format for maintaining the consistency.

## 2.2 Data Augmentation and Transformation

Data augmentation is applied when the image is loading, so that the ability of the model can be generalized and the transform parameters which are used for allowing the various image transformation using PyTorch's transformation functions. This includes resizing the image into to a standardized format (224x224 pixels), converting images to PyTorch tensors and augmentations like random rotations or flips.

## 2.3 Dataset Splitting

Using the scikit-learn function the dataset is divided into three parts which are train, test and validation using the **‘train\_test\_split’.**  The training set is used for training the model, where it learns the patterns, features, and representations portion of the data. Whereas the testing and validation sets are used for evaluating the model performance.

A screen shot of a computer program

Description automatically generated

This splitting is done in the init method of the CreateDataset class. The train\_size, test\_size, and valid\_size parameters control the proportions of data allocated to each split.

## 2.4 Data Frame Construction

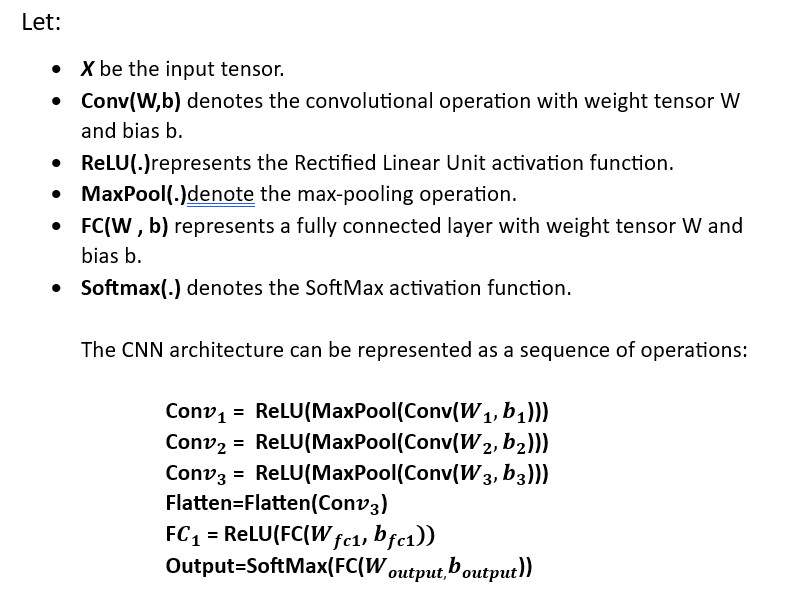
Pandas’ library has been used for constructing the data frame. They are used for maintaining the organization and representation of the data. A data frame provides a structure to organize and store your data. It arranges the data in rows and columns, making it easy to represent and manipulate and offers a convenient way to access, retrieve, and manipulate specific subsets of the data. You can easily index, and filter data based on conditions.

The image\_paths function is responsible for extracting class names, file paths, and class labels from the dataset directory. These details are then organized into lists, and the convert\_to\_df function constructs a Pandas DataFrame (dataset.csv). This DataFrame serves as a main data structure for efficient handling and indexing during stages of the project.

# 3.MODEL BUILDING

## 3.1 Convolutional Neural Networks

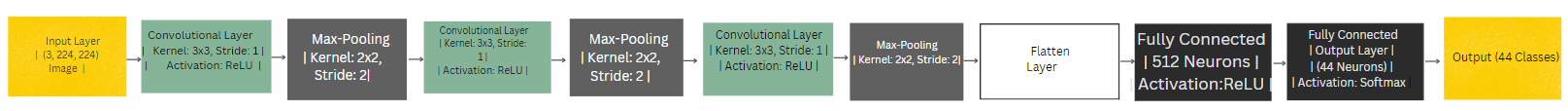
The core of the project is design and architecture of convolutional Neural networks. The CoNvNet is a deep learning model which is specifically used for image classification tasks, adept at extracting features from medical images of the dataset  (Tammina, 2019). The architecture consists of three convolutional layers with ReLU activation functions and max-pooling operations. After the convolutional layers, the output is flattened and passed through two fully connected layers for classification. The final output is obtained through a softmax activation function. Below is the formula for the model.



This model is designed to learn hierarchical features from input images and make predictions across 44 classes. The convolutional layers capture spatial patterns, while the fully connected layers perform classification based on the learned features. The softmax activation at the output layer provides normalized class probabilities.

## 3.2 Model’s Architecture and design

The architecture and design of the model is composed in several layers, each giving a specific purpose in the feature extraction and classification process. Below is a clear explanation of the architecture of the model using a flow chart:



1. **Input Layer:**
   * Accepts RGB images with a size of (3, 224, 224), representing the colour channels and image dimensions.
2. **Convolutional Layers:**
   * Three convolutional layers are employed, each followed by Rectified Linear Unit (ReLU) activation functions to introduce non-linearity.
   * ReLU activation of the model is used for non-linearity, allowing the model to learn complex patterns and representations.
   * These layers use the filters which are small sized to capture local features, with max-pooling operations to reduce spatial dimensions.
   * Max-pooling is applied to down sample the occupying the dimensions and to reduce the computational complexity while retaining essential features.
3. **Flatten Layer:**
   * The output from the convolutional layers is flattened into a 1D tensor, preparing it for the fully connected layers.
   * The flatten layer converts the output from convolutional layers into a 1D tensor, easing connectivity with fully connected layers.
4. **Fully Connected Layers:**
   * Two fully connected layers contribute to the final classification.
   * The first fully connected layer has 512 neurons, promoting feature understanding.
   * The second fully connected layer produces the final output with the number of neurons corresponding to the number of output classes.
5. **Output (44 Classes):**

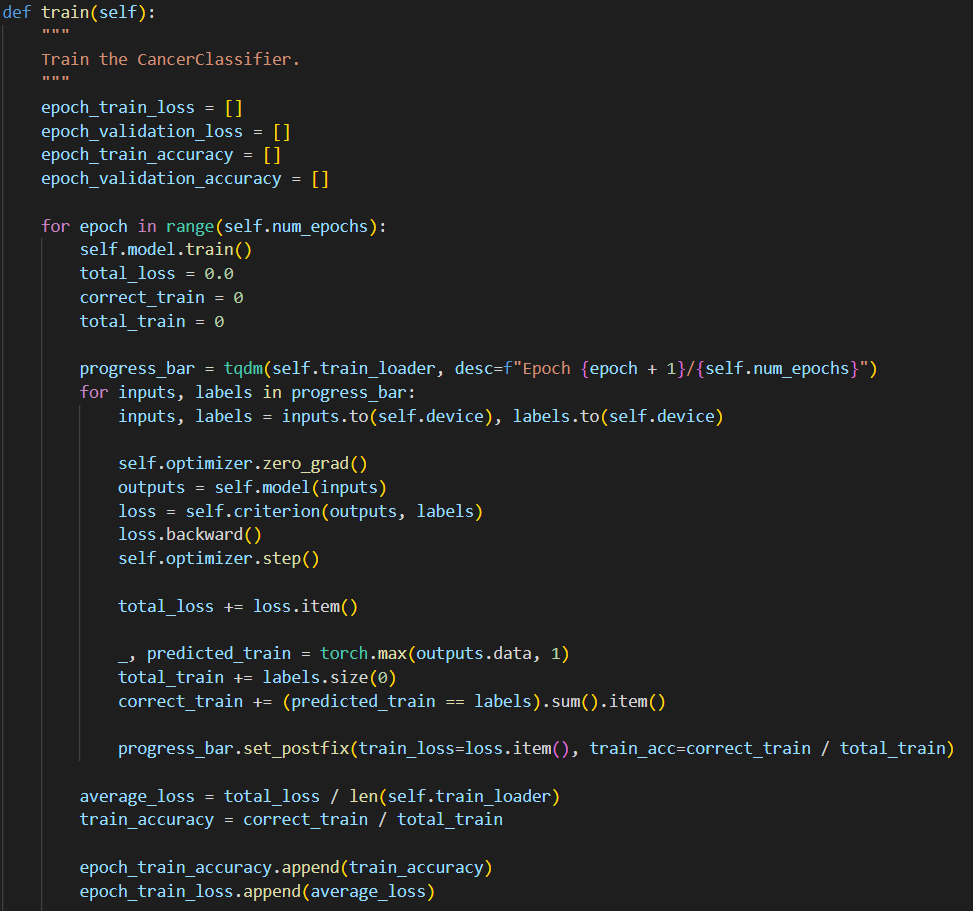
* The final output layer with 44 neurons represents the predicted probabilities for each class. The class with the highest probability is the predicted class.

## 3.3 Loss Function and Optimization

During the training process, the model optimizes its parameters utilizing the Cross-Entropy Loss. This loss function quantifies the dissimilarity between the predicted probability distributions and the actual class labels. Specifically chosen for classification tasks, Cross-Entropy Loss guides the model towards more accurate predictions. The Adam optimizer, known for its adaptive learning rate mechanism, dynamically adjusts the learning rate during training. This adaptability enhances convergence speed and mitigates the risk of getting stuck in local minima. The synergy of Cross-Entropy Loss and Adam optimization ensures efficient parameter updates and effective learning (Yaqub et al., 2020).

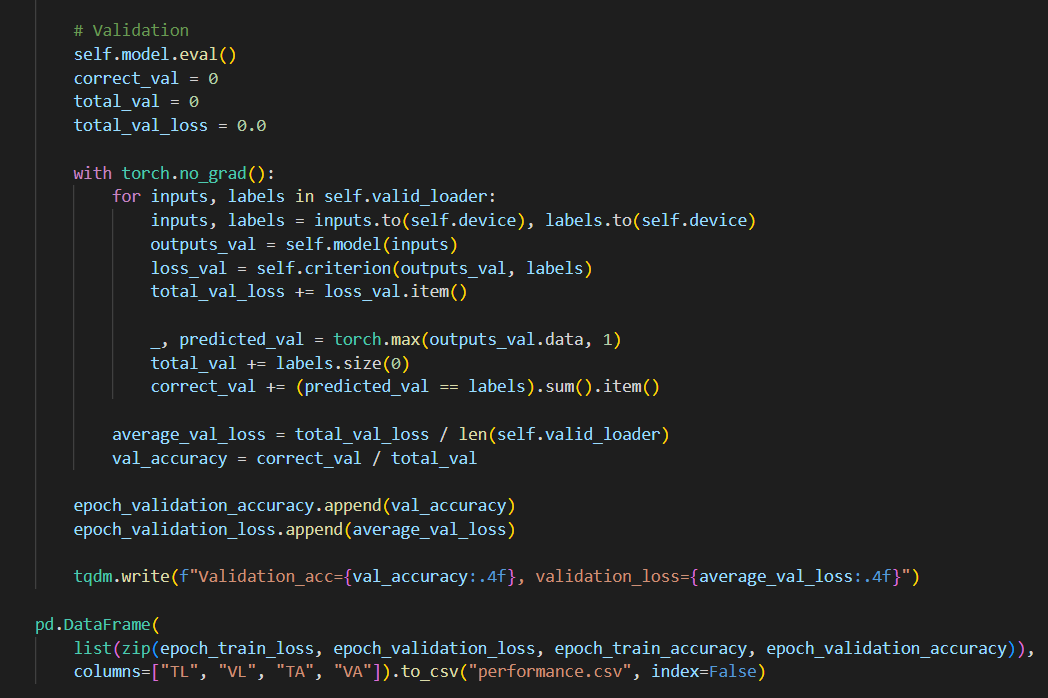
## 3.4 Training Loop and Backpropagation

The training mechanism resides in the training loop embedded in `main.py`. This iterative loop processes the batches of training data and facilitates the optimization of the CoNvNet's parameters. The deep learning concept takes main stage during each iteration. it involves computing gradients of the loss with respect to the model's parameters (Liu et al., 2020). These gradients show the adjustment of weights, steering the model towards minimizing the overall loss. The nature of the training loop allows the CoNvNet to repeatedly refines its understanding of the dataset.

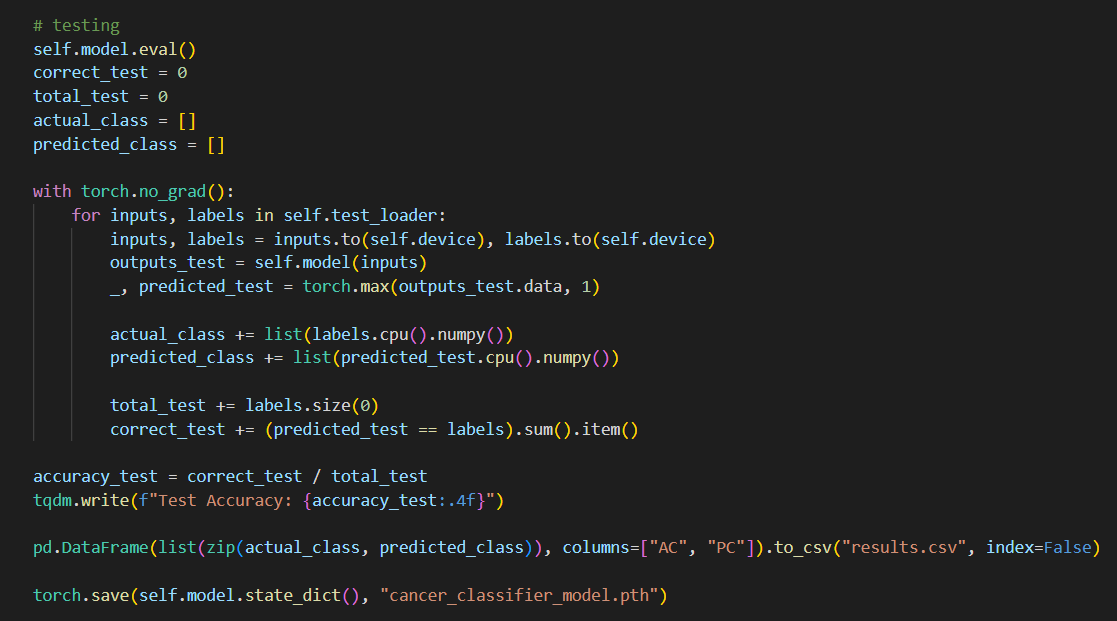


## 3.5 Validation and Testing

To make sure that the CoNvNet generalizes well to previously unseen data, a dedicated validation phase is integrated into the training loop. At specified intervals, the model's performance is assessed on a validation set, providing insights into its generalization capability.



the completion of training epochs, the model undergoes testing on an independent test set. The resulting the accuracy, along with a detailed confusion matrix, is recorded in `results.csv`. Additionally, the state dictionary of the trained model is saved as `cancer\_classifier\_model.pth`, facilitating further process for the model deployment.



# 4.Evaluation Metrics

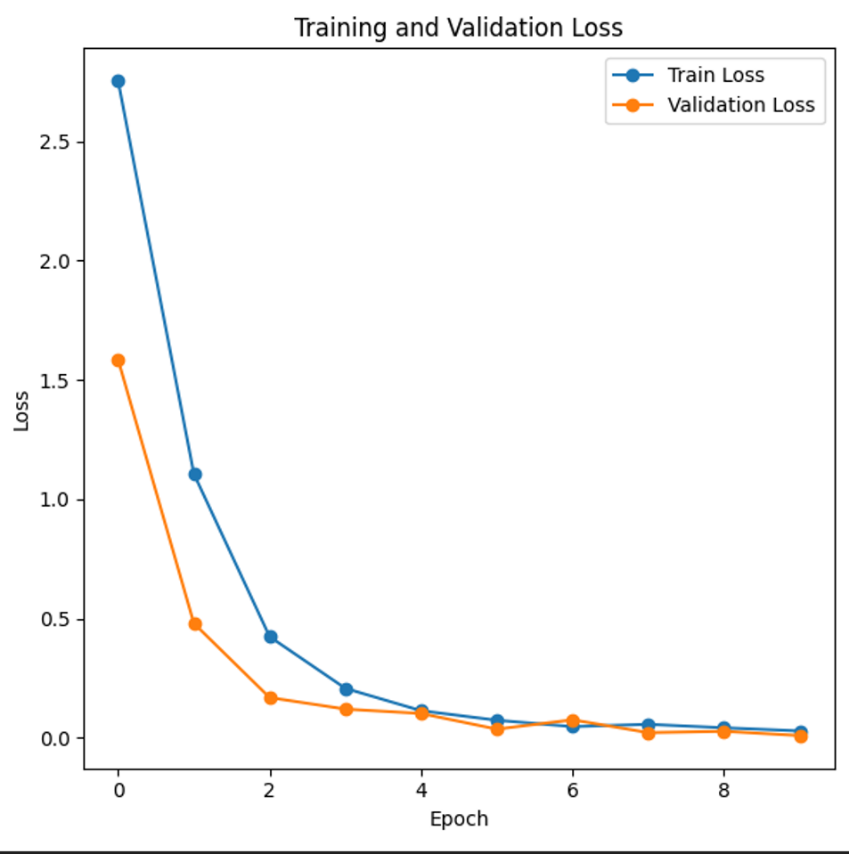
## 4.1 Training and Validation Metrics

The training and validation process was monitored over multiple epochs, and key metrics were recorded in the "performance.csv" file. Let's delve into these metrics into:

### 4.1.1 Training Loss and Validation Loss

* **Training Loss (TL):** Represents the error during the training phase. It should ideally decrease over epochs, indicating that the model is learning and adjusting its weights to minimize errors.
* **Validation Loss (VL):** Indicates the error on a separate validation dataset. Its trend is crucial to identify overfitting. A decreasing trend aligns with model improvement.

|  |  |  |
| --- | --- | --- |
| **Epoch** | **Train Accuracy (TA)** | **Validation Accuracy (VA)** |
| **1** | 25.78% | 52.46% |
| **2** | 67.04% | 75.22% |
| **3** | 85.44% | 82.59% |
| **4** | 92.80% | 86.16% |
| **5** | 96.13% | 89.06% |
| **6** | 97.83% | 85.94% |
| **7** | 97.95% | 86.83% |
| **8** | 98.75% | 90.62% |
| **9** | 99.43% | 89.51% |
| **10** | 99.35% | 89.96% |

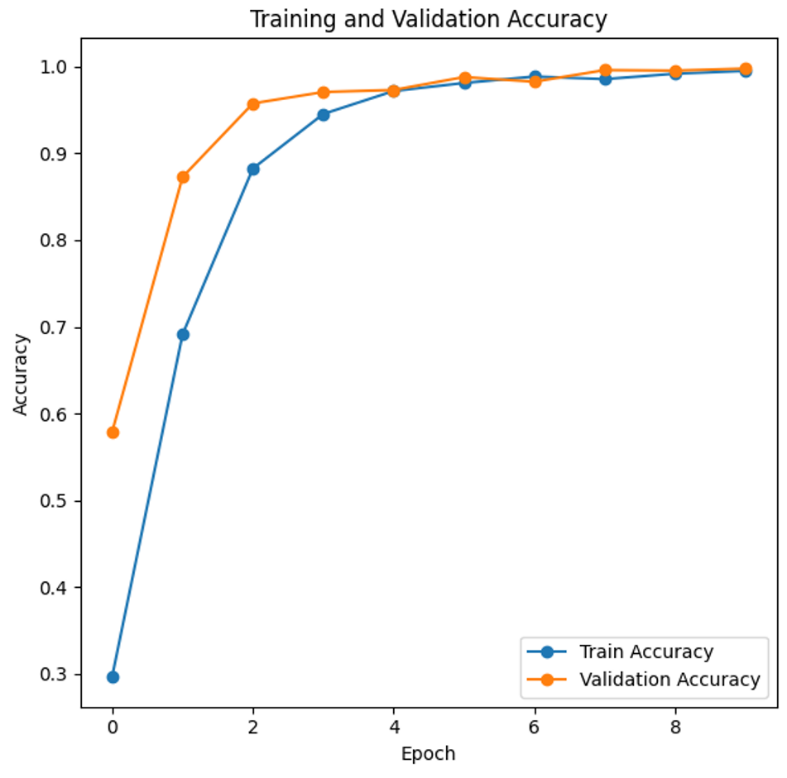


The decreasing TL and VL indicate that the model is effectively learning from the dataset, demonstrating convergence.

### 4.1.2 Training Accuracy (TA) and Validation Accuracy (VA)

* **Training Accuracy (TA):** Represents the accuracy of the model on the training set. It's the ratio of correctly predicted instances to the total instances in the training set.
* **Validation Accuracy (VA):** Indicates how well the model generalizes to new, unseen data.

|  |  |  |
| --- | --- | --- |
| **Epoch** | **Train Loss (TL)** | **Validation Loss (VL)** |
| **1** | 2.841 | 1.772 |
| **2** | 1.228 | 0.915 |
| **3** | 0.521 | 0.724 |
| **4** | 0.248 | 0.519 |
| **5** | 0.137 | 0.416 |
| **6** | 0.082 | 0.508 |
| **7** | 0.087 | 0.521 |
| **8** | 0.051 | 0.368 |
| **9** | 0.025 | 0.419 |
| **10** | 0.025 | 0.404 |



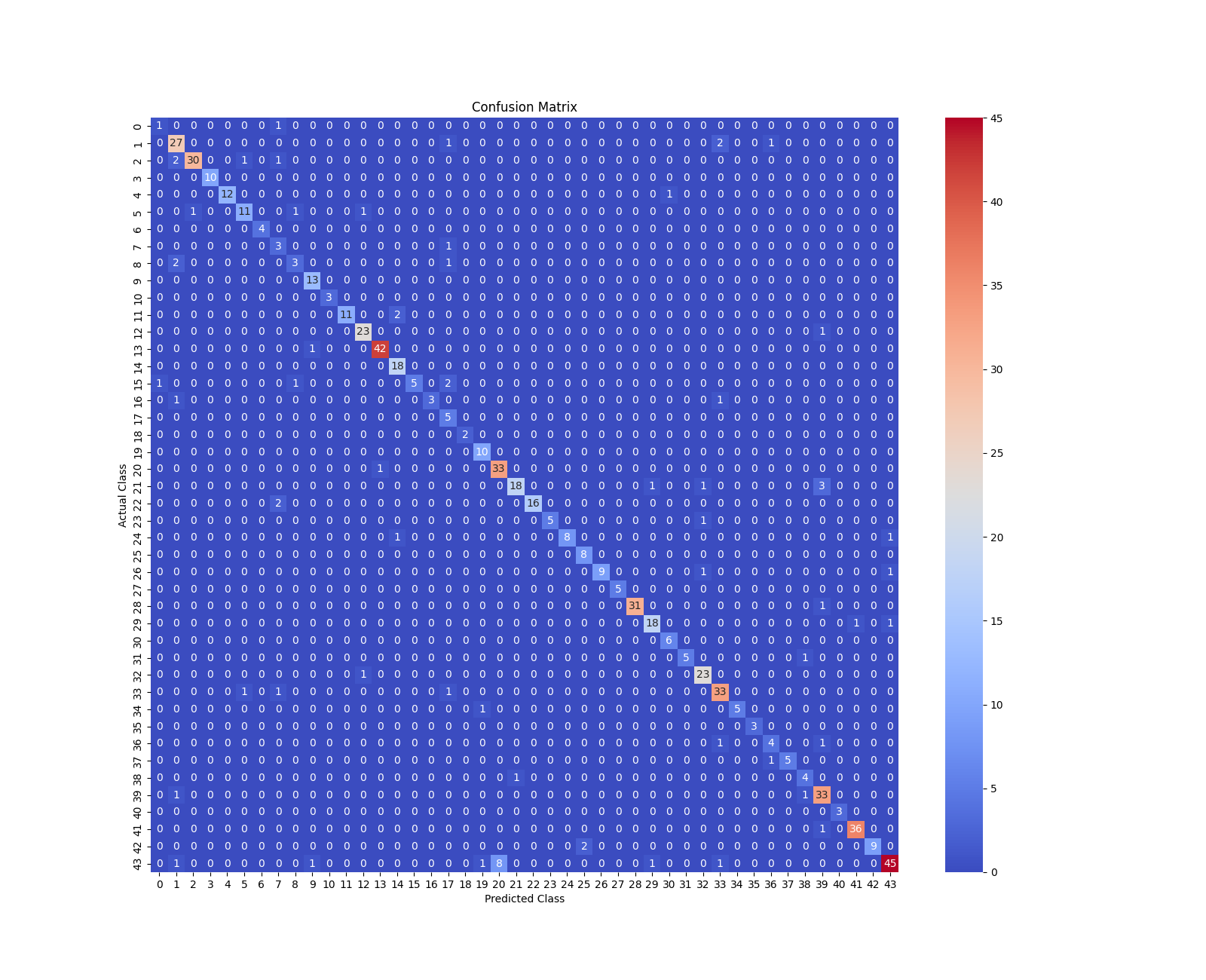
The increasing trend in TA and VA indicates that the model is learning and generalizing well to both training and validation sets.

## 4.2 Test evaluation Metrics

The trained model was evaluated on a separate test set to assess its performance on unseen data. The results were recorded in the "results.csv" file. These metrics collectively indicate the model's ability to correctly classify brain tumour images. The confusion matrix and classification report has been used for test metrics evaluation.

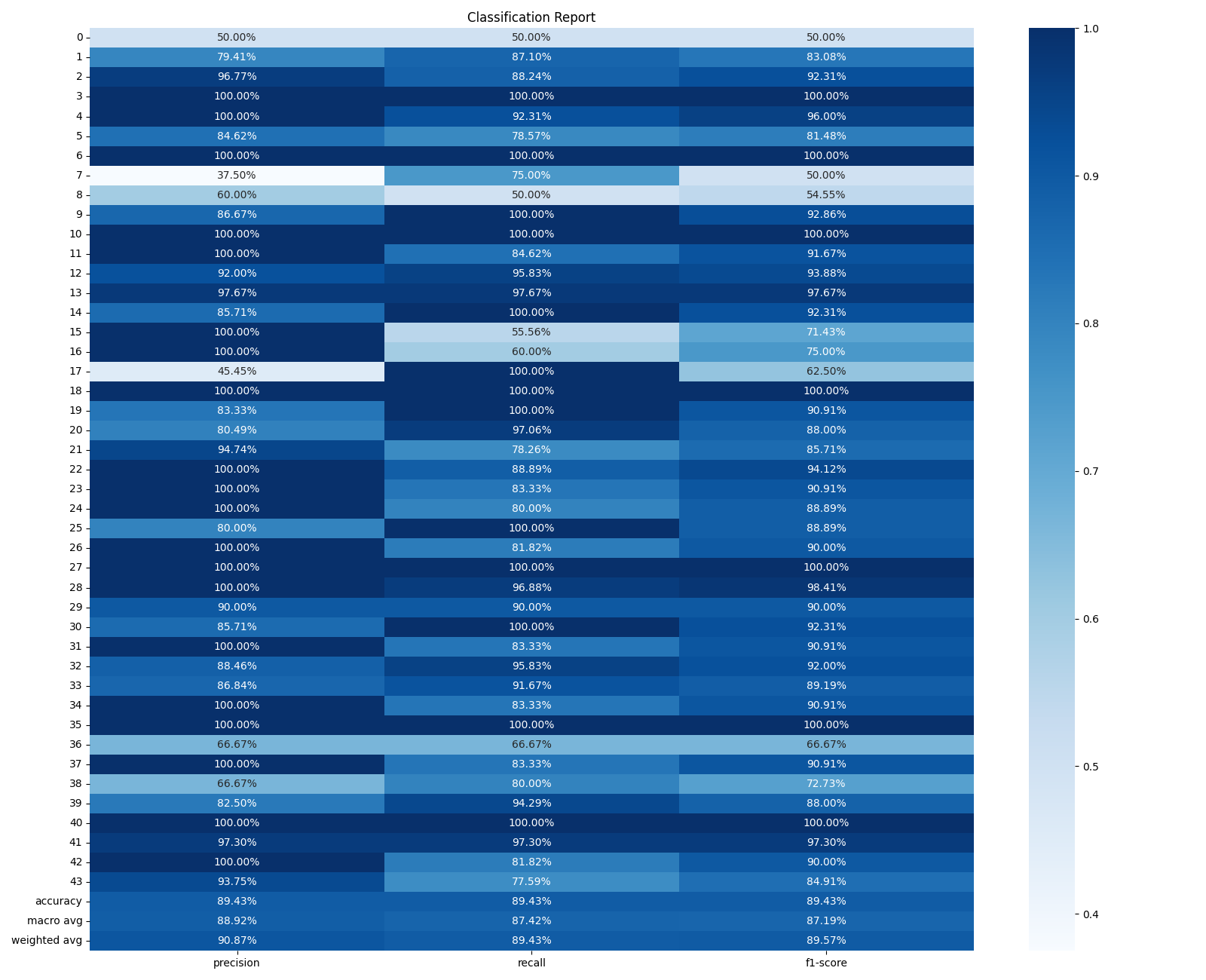
### 4.2.1 Confusion Matrix

A confusion matrix provides a detailed breakdown of the model's predictions versus the actual class labels. It helps identify true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Given that there are 44 classes, the confusion matrix would be a 44x44 matrix.



* **True Positives (TP):** The number of instances where the model correctly predicted a positive class. It’s the corresponding cases where the model correctly identified a brain tumour belonging to a specific class.
* **True Negatives (TN):** The number of instances where the model correctly predicted a negative class. In a multi-class problem, this is a bit less simple, as each class beyond the one of interest is treated as a negative class. TN is the sum of correct predictions for all the other 43 classes.
* **False Positives (FP):** The number of instances where the model incorrectly predicted a positive class. These are cases where the model predicted a brain tumour for a class, but there was no tumour of that class in the actual data.
* **False Negatives (FN):** The number of instances where the model incorrectly predicted a negative class. In the cases where the model failed to predict a brain tumour of a certain class that was present in the actual data.

### 4.2.2 Classification Report

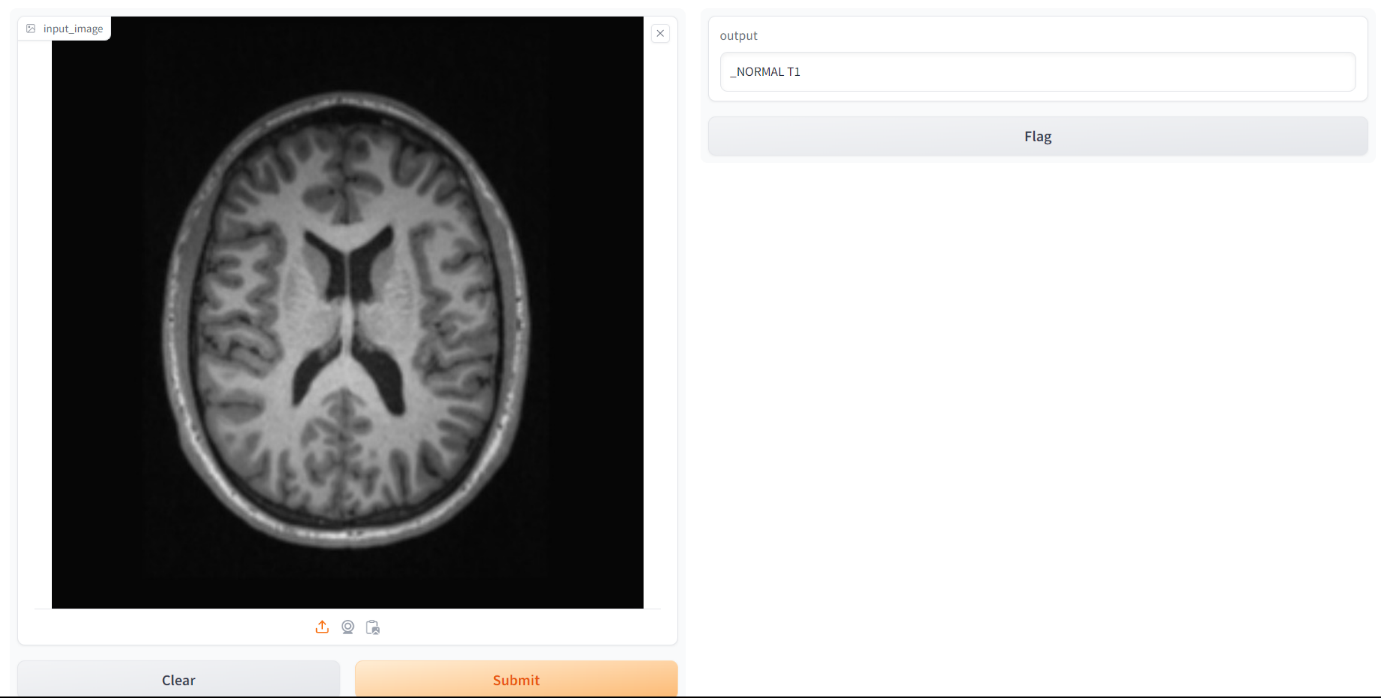


* **Accuracy (89.43%):** The proportion of correctly classified instances out of the total instances. A high accuracy indicates overall good performance.
* **Precision (90.87%):** The ratio of correctly predicted positive observations to the total predicted positives. It is particularly relevant in scenarios where false positives are critical.
* **Recall (89.43%):** The ratio of correctly predicted positive observations to the actual positives. It is crucial when the cost of false negatives is high.
* **F1 Score (89.57%):** The mean of precision and recall. It balances precision and recall, providing a single metric to assess the model's overall performance.

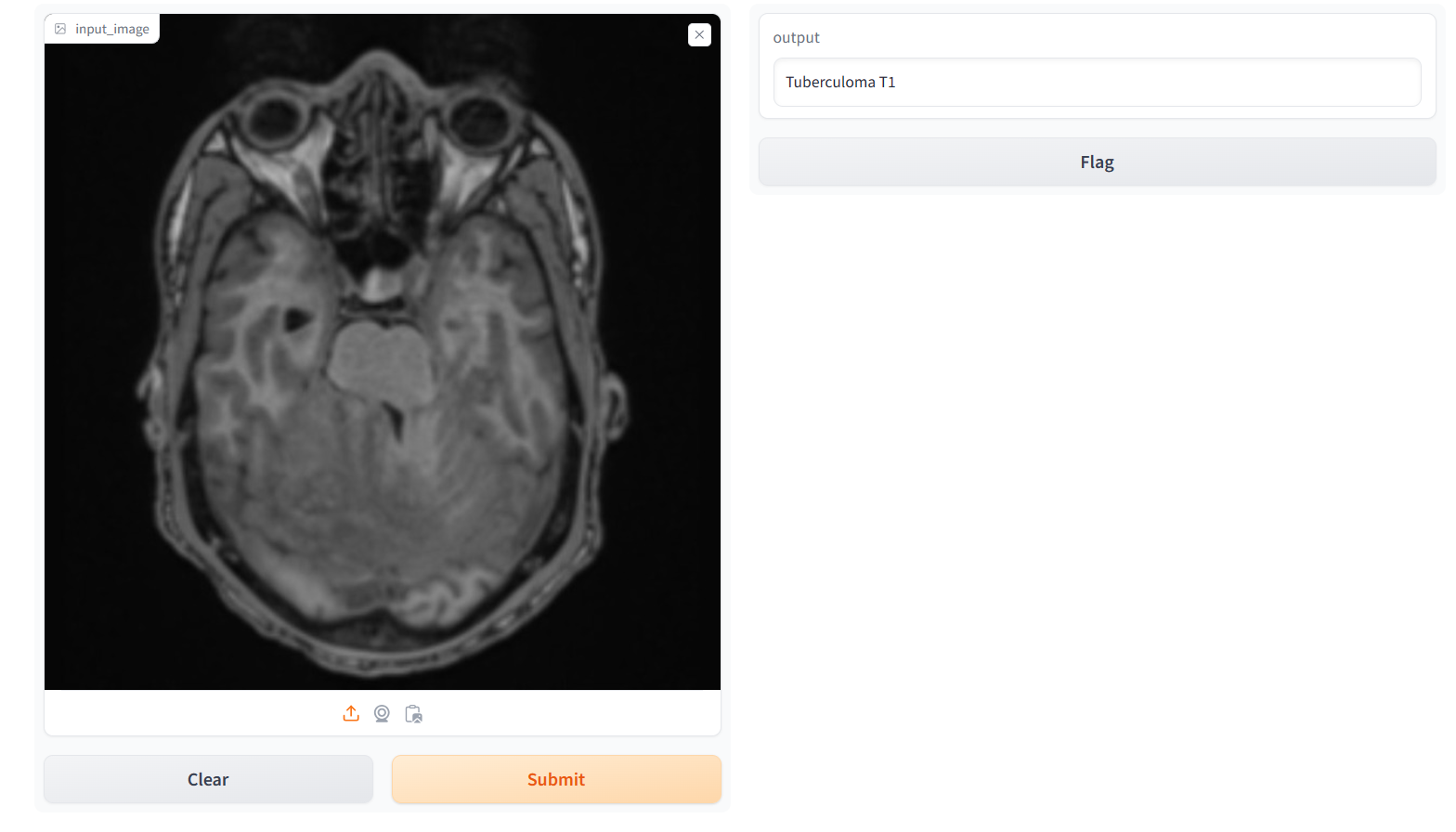
## 4.3 Prediction of images using Gradio

To do the Prediction, the trained model is saved as `cancer\_classifier\_model.pth`, has been deployed using gradio to classify through the local host. Three images were uploaded, and the model predicted them correctly. The screenshots of the predictions are below.

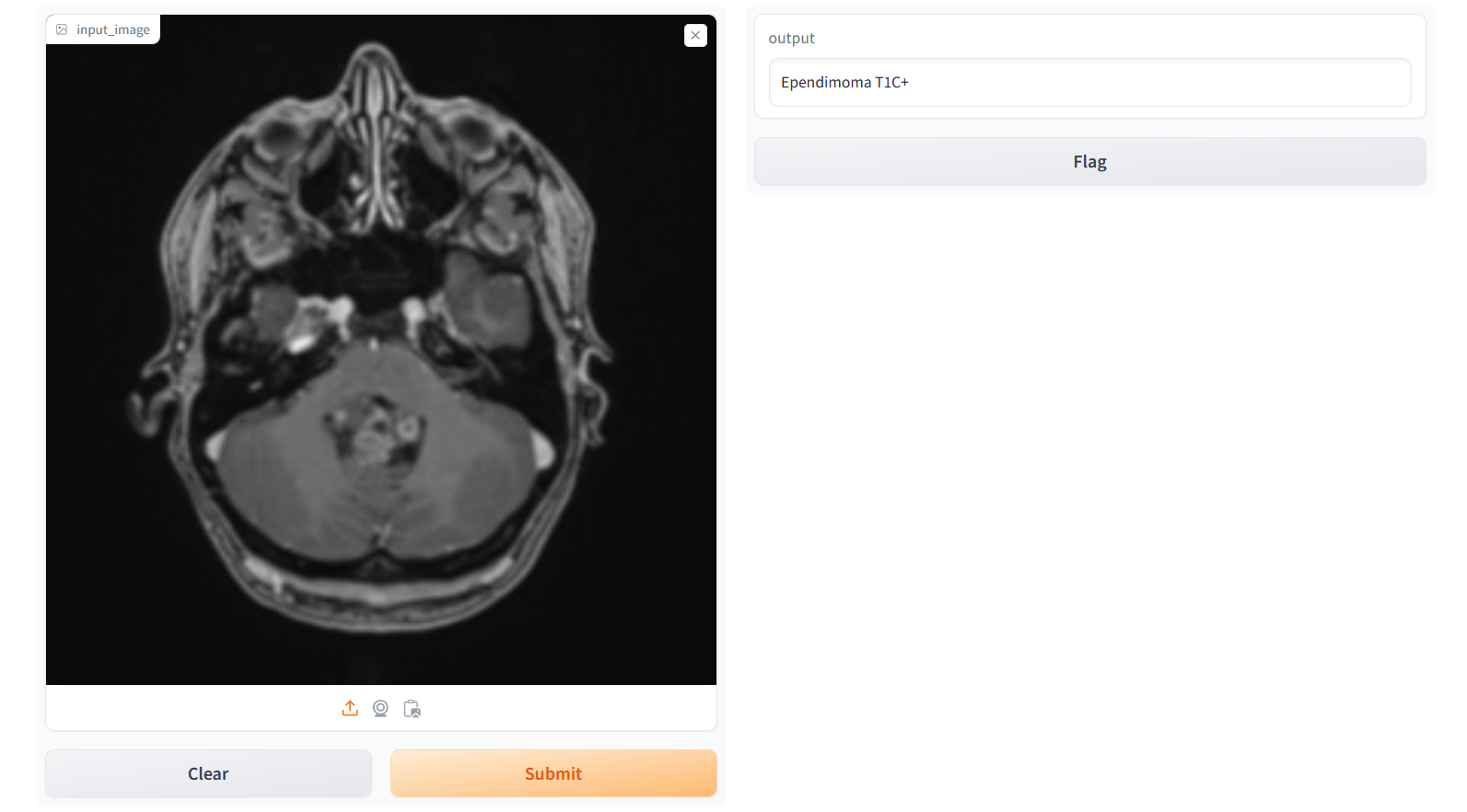
1. The Image used for the prediction is Normal T1



1. The image used for the prediction ia tuberculoma T1



1. The image used for the prediction is epedimoma T1c+



# Conclusion

In conclusion, this project has developed a Convolutional Neural Network, for the classification of brain tumor images. The decision to train the CNN model from scratch rather than using pre-trained models allowed for a customized approach, has adapted for the specifics of brain tumor image. The Cross-Entropy Loss function during training proved facilitating the model's ability to learn the features and patterns across the 44 classes.

The adaptive learning rate of the Adam optimizer further expedited convergence, ensuring efficient weight updates. Core concepts like backpropagation, epochs, and validation were integral to refining the model iteratively. The convergence maintaining effective weight adjustments. Core concepts like backpropagation, epochs, and validation were integrated to the model iteratively. The evaluation on the model was done to show metrics of the model. Moreover, the predictability of the model in real-world scenarios, as demonstrated by the user-friendly prediction interface, underscores its practical applicability and potential impact on medical diagnostics through deep learning techniques.

# Reference

Badža, M.M. and Barjaktarović, M.Č. (2020). Classification of Brain Tumors from MRI Images Using a Convolutional Neural Network. *Applied Sciences*, 10(6), p.1999. doi:<https://doi.org/10.3390/app10061999>.

Liu, P., Shi, L., Miao, Z., Jin, B. and Zhou, Q. (2020). Relative Distribution Entropy Loss Function in CNN Image Retrieval. *Entropy*, 22(3), p.321. doi:<https://doi.org/10.3390/e22030321>.

Tammina, S. (2019). Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. *International Journal of Scientific and Research Publications (IJSRP)*, 9(10), p.p9420. doi:<https://doi.org/10.29322/ijsrp.9.10.2019.p9420>.

Tandel, G.S., Balestrieri, A., Jujaray, T., Khanna, N.N., Saba, L. and Suri, J.S. (2020). Multiclass magnetic resonance imaging brain tumor classification using artificial intelligence paradigm. *Computers in Biology and Medicine*, 122, p.103804. doi:<https://doi.org/10.1016/j.compbiomed.2020.103804>.

Yaqub, M., Feng, J., Zia, M.S., Arshid, K., Jia, K., Rehman, Z.U. and Mehmood, A. (2020). State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images. *Brain Sciences*, [online] 10(7), p.427. doi:<https://doi.org/10.3390/brainsci10070427>.