A black background with white text

Description automatically generated with low confidence

MSc Data Science Project

7PAM2002-0509-2023

Department of Physics, Astronomy and Mathematics

**Data Science FINAL PROJECT REPORT**

**Project Title: Innovative Deep Learning Models for Brain Tumour Diagnosis Using MRI Imaging**

**Student Name and SRN: Ahsan Farooq**

Supervisor: Vanadana Das

Date Submitted: 28 August 2024

Word Count: 7000

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct and plagiarism information at [Assessment Offences and Academic Misconduct](https://www.herts.ac.uk/__data/assets/pdf_file/0007/237625/AS14-Apx3-Academic-Misconduct-v17.0.pdf) and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6). I have not used ChatGPT, or any other generative AI tool, to write the reportor code (other than were declared or referenced).

I did not use human participants or undertake a survey in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Ahsan Farooq

Student Name signature: Ahsan

Student SRN number: 22080535

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

Acknowledgement

Abstract

This report briefly describes a method to correctly classify brain MRI images into four types of brain tumours using deep learning approaches. This report aims to classify brain MRI images (into four broadly used brain tumour types: glioma, meningioma, pituitary tumour, and no tumour). In a preprocessing pipeline stage, the input images are processed with various image augmentation approaches, such as manipulating brightness and scaling the image. Furthermore, to build the mechanism labels were assigned to each image and the images were split into two sets of data: training and testing data. Two models were built: a custom model, which is a CNN, with multiple convolutional layers, followed by batch normalisation and an activation function (L2 regularisation available). Another model, a transfer learning model built using Xception architecture, is utilized. Finally, both models are trained, and their performance is evaluated using metrics like accuracy, loss, confusion matrices, and ROC curves. A custom CNN model has a final test accuracy of 96.49%. In contrast, the transfer learning model has 98.94% accuracy. From this simple application of deep learning to medical images from the UCI Machine Learning Repository, we can conclude that we can use deep learning approaches to classify medical images and transfer learning can be used to wrangle the weights assigned to the activations in each layer of the neural network to improve the model's performance.

**Keywords— Brain tumour** classification, convolutional neural networks, transfer learning, MRI image classification, Xception, image augmentation, deep learning, medical image analysis.

Table of Contents

Contents

[**Chapter I** 6](#_Toc175591090)

[Introduction: 6](#_Toc175591091)

[**Chapter II** 9](#_Toc175591092)

[Literature Review 9](#_Toc175591093)

[Chapter III 12](#_Toc175591094)

[Methodology: 12](#_Toc175591095)

[**Chapter IV** 17](#_Toc175591096)

[Analysing the Training Results 17](#_Toc175591097)

[Evaluation of models on the test set 19](#_Toc175591098)

[**Chapter V** 24](#_Toc175591099)

[Model Comparison: 24](#_Toc175591100)

[Future Work: 24](#_Toc175591101)

[Limitations: 24](#_Toc175591102)

[Conclusion: 25](#_Toc175591103)

[Chapter VI 26](#_Toc175591104)

[References 26](#_Toc175591105)

[**Chapter VII** 28](#_Toc175591106)

[Appendix: 28](#_Toc175591107)

# **Chapter I**

## Introduction:

# A brain tumour is one of the most devastating diseases of the central nervous system. A brain tumour is the abnormal proliferation of cells presenting in the brain or part of the brain (meninges, cranial nerves, pituitary gland and their surrounding tissues), which can spread within the skull and often to other parts of the body. Brain tumours are classified as primary, originating within the brain, and secondary, or metastatic, derived from a primary tumour somewhere else in the body (Louis et al., 2016). Based on the type, brain tumours can be classified into two groups: benign (non-cancerous) and malignant (cancerous). The malignant tumours can be more aggressive and give a poor prognosis with a greater chance of patient suffering and death.

There is a substantial global burden regarding brain tumours, with about 308,000 new cases diagnosed in 2020 (Global Cancer Observatory, 2020). Some brain tumours grow slowly and are easier to treat, while some are highly aggressive, and the patient may not have much time to go. A brain tumour can disturb crucial functions such as cognition, movement and sensation, depending on its location in the brain (Omuro & DeAngelis, 2013). The patient can present with a variety of neurological manifestations, including headaches, seizures, cognitive impairments, and motor deficits. Due to such information, brain tumours represent a significant public health problem, as they cause high disability-adjusted life years and entail major healthcare expenses.

**Importance of Early and Accurate Diagnosis**

An early and accurate diagnosis of brain tumours is a key that unlocks the door to better treatment outcomes. Since the brain represents the most complex organ in the body, with many vitally important functions, any aberrant growth in the brain can affect an individual’s health, lifestyle and survival. The earlier a brain tumour is diagnosed, the earlier the intervention, which helps to stop further damage to the nervous system and decreases morbidity and mortality (Weller et al., 2015). Moreover, correct classification as a tumour is necessary for making the right treatment strategy; for instance, benign tumours require minimal treatment, such as initial observation, while malignant tumours usually require a combination of treatments, such as surgery, radiation or chemotherapy (Schwartz et al., 2017).

The wrong treatment regime or the risk of complications can result in incorrect classification or poorer patient outcomes. Because of the importance of addressing these issues, medical researchers try to diagnose brain tumours as early as possible and classify them more precisely.

**Role of MRI Imaging in Diagnosing Brain Tumours**

Magnetic Resonance Imaging (MRI) is an essential tool in diagnosing brain tumours. It is a non-invasive imaging technique that uses powerful magnets and radio waves to produce detailed images of the brain's structure (Glover, 2011). Unlike computed tomography (CT) scans, which use ionising radiation, MRI provides superior soft tissue contrast without exposing the patient to potentially harmful radiation. This makes MRI particularly valuable for imaging the brain, where distinguishing between different tissue types is crucial for identifying abnormalities such as tumours.

Since then, MRI has become the diagnostic gold standard for brain tumours. If an MRI scan shows a brain lesion, an anatomical MRI, mainly when performed with a high-field magnet, can identify a tumour's size, location and features (Villanueva-Meyer et al., 2017). Advanced MRI techniques, like diffusion-weighted imaging (DWI), perfusion MRI and functional MRI (fMRI), can reveal more about how the tumour behaves, including its speed of growth (aggressiveness), blood supply (vasculature) and its impact on neighbouring brain functions (Leach et al., 2020). For example, it estimates tumour vascularity to see if it comprises many tiny blood vessels (malignant tumours). In your body to help tumour tissue from surrounding.

MRI is beneficial not only for diagnosing brain tumours but also for planning and following up treatment. Before surgery, MRI scans help neurosurgeons pinpoint the best way to remove a tumour safely and effectively; post-surgical scans are used to screen for recurrence or complications (Lemasson et al., 2013). MRI is also used to guide radiotherapy, where tumours are targeted with radiation to reduce the risk of damage to healthy tissue.

**Complexity and Variability in Tumour Classification:**

Despite these advances, MRI-based manual diagnosis and classification of brain tumours is still complex, tedious, and subject to considerable inter- and intra-rater variability (Menze et al., 2015). One of the primary reasons is the tremendous variability in the characteristics of brain tumours. Unlike other health conditions, brain tumours vary dramatically in size, shape and location, with subtle differences even in their microscopic structure, complicating the classification decisions made by radiologists, who must figure out the boundary between benign and malignant tumours or between different types of malignant tumours (Louis et al., 2016). For instance, gliomas, one of the most common tumours, according to the World Health Organization, can appear in a wide variety of forms, ranging from their low-grade counterparts that are relatively well-delineated to the high-grade glioblastomas that are infiltrative and hard to differentiate from the surrounding tissue.

Additionally, ring-enhancing lesions, one of the pathognomonic MRI characteristics of cavernous haemangioma, can occur in many other brain tumours that are seldom found in cavernous haemangioma, such as high-grade glioma, leading to misclassification (Villanueva-Meyer et al., 2017). For example, high-grade glioma and brain metastases that are ring-enhancing can be nearly indistinguishable on MRI alone without the benefit of additional clinical history/examination or advanced imaging.

Moreover, manual interpretation of MRI images is labour-intensive and inter-observer variability (i.e., two different radiologists might reach different opinions when assessing the same set of images) compromises interpretation, which in turn can impact diagnosis, prognosis and consequent treatment planning (Zwanenburg et al., 2020).

Therefore, research into automated and semi-automated AI and machine learning-based methods of brain tumour diagnosis and classification is gaining momentum. The eventual goal is to use them to improve diagnostic accuracy, usually by analysing huge data sets of MRI images for statistical patterns that are not immediately obvious to the unaided human eye (Akkus et al., 2017). Whether and under what circumstances such approaches will prove both valid and implementable remains unclear.

**Motivation and Relevance**

We have more complex and abundant medical imaging data, so healthcare demands more accurate and faster diagnosis tools. Radiology exemplifies how early and accurate diagnosis is critical for treating many diseases, such as brain tumours. However, the traditional way of manually interpreting medical images, such as Magnetic Resonance Imaging (MRI) by radiologists, is tedious and still susceptible to human’s fluctuating judgment on different occasions. In this situation, AI is emerging as a tool for medical image analysis. Specifically, deep learning techniques such as convolutional neural networks (CNNs) can augment radiologists' work to reach fast, accurate, and consistent diagnoses in record time, leading to improved patient outcomes.

**Role of AI in Medical Imaging**

Medical imaging stands out from various fields as one of AI’s most promising applications. In this context, deep learning, one of the most grounded and successful branches of machine learning, has emerged as an impressive application of AI in the last couple of years. Because of their propensity toward modelling the human brain’s neural networks, deep learning models can learn features from images in an end-to-end fashion without the intervention of humans. Several classes of deep learning models are well-suited for applications in medical imaging, and the most successful models are convolutional neural networks (CNNs).

CNNs look for image patterns and structures by applying multiple filters, such as edge, texture, shape and other feature detectors (Litjens et al., 2017). These features could be components such as anatomy lesions or tumours in medical images. After training a CNN on large datasets of labelled medical images, the network has learnt how to identify anomalies and assign them accurately to specific classes of images. This approach has successfully automated complex medical-image classification tasks such as identifying tumours, segmenting organs and identifying disease groups.

Because of their need to analyse vast amounts of data swiftly, one of the most critical roles for AI in medical imaging is to take on some of the load relevant to large-scale image review. A radiologist might have to review hundreds or thousands of images daily, arguably a task likely to lead to fatigue and potentially augmented risk of error (Topol, 2019). AI algorithms can analyse parse images swiftly and draw attention to several abnormal findings requiring the radiologist’s attention, reducing the time needed for image interpretation and, significantly, bolstering diagnostic accuracy with human expertise.

**Need for Automated Systems in Medical Imaging**

The growing use of medical imaging for diagnosis and treatment planning has also increased the pressure on radiologists to keep up with the workload and reduce diagnosis and care delays. The growing demand of patients for imaging services puts great strain on radiologists, increasing the risk of professional burnout, errors and delays in diagnosis (European Society of Radiology, 2019). To address these challenges, AI-driven automated systems can be applied to clinical workflows to support radiologists and improve patient care.

Another reason to expect clinicians to use automated systems is when the stakes for patients are high, as they are with some postoperative outcomes. AI can improve patient outcomes. It can lower the time to diagnosis and inform treatment planning for time‑sensitive conditions such as stroke (causing brain injury) or traumatic brain injury, where early intervention can improve outcomes (Mazurowski et al., 2019). In oncology, AI tools allow the early detection of many tumours, so treatments can be specific to the cancer and administered earlier, improving survival rates. AI systems can also be updated as more data becomes available. This is important over time as the recommendations in medicine change. New patterns become known, and new evidence is published. For AI, this is relatively easy to do.

# **Chapter II**

## Literature Review

Recently, machine learning imaging has been attractive to logicians in diagnostic systems. Moreover, recent achievements have shown that machine or deep learning can significantly simplify complex diagnostic tasks, such as identifying brain tumours. Brain tumours longer they are not diagnosed, and they affect health. Diagnosis of brain tumours is possible at an early age through the non-invasive method, the MRI technique (Sun et al., 2020). Unfortunately, traditional classification methods cannot distinguish complex and heterogeneous tumour structures. Therefore, centralised human diagnosis can be time-consuming and inaccurate. The following literature review shows the evolution of brain tumour classification into machine learning and deep learning principles. It covers traditional to modern and transfer learning.

**Traditional Approaches to Brain Tumour Classification:**

Early clinical approaches used handcrafted features and classic machine learning algorithms (Chaplot et al., 2006), such as 1. extracting texture, shape and intensity features from the images; 2. classifying between different tumours using a support vector machine (SVM), k-nearest neighbours (KNN) or random forests, to name but a few. Handcrafted features are vital for classic machine learning models because the classification uses these features, not images. However, model performance is limited when you cannot guarantee that the extracted handcrafted features represent the complex patterns within biomedical images. Therefore, approaches built on classic machine learning models struggle when generalising across heterogeneous datasets, leading to poor classification performance (Rathore et al., 2017).

For instance, Chaplot et al. (2006) used discrete wavelet transform (DWT) for feature extraction and support vector machine (SVM) for classification to identify brain tumours in MRI scans. While the approach delivered decent accuracy, the manual feature extraction required significant time and domain expertise. Similarly, conventional models such as decision trees and logistic regression fell short of handling the high dimensionality of medical image data, naturally pushing towards overfitting or underfitting (Rathore et al., 2017).

**Emergence of Deep Learning in Medical Imaging:**

Deep learning, particularly CNNs, has transformed medical image classification by automatically allowing the method to learn hierarchical features from raw data. CNNs are particularly well-suited to learn features for object classification when the input data are images because they can learn features at different levels of abstraction, such as edges, textures and complex shapes (Litjens et al., 2017). A human can label images accurately, such as by spotting a pathological area within the image. However, creating a list of features describing what makes the image a positive case is complicated. CNNs learn the right features from the raw data.

Several studies have shown that CNNs are much better at classifying brain tumours than traditional approaches. For example, Pereira et al. (2016) used a CNN architecture for brain tumour classification using MRI scans. They used different layers of convolutions to extract features from brain images, followed by dense layers for final classification. This hierarchical modelling approach helped in the automated learning of discriminative features and improved classification performance.

Similarly, Hossain et al. (2020) used a deep CNN model to classify brain tumours into glioma, meningioma, and pituitary tumours, achieving over 95% accuracy. The authors emphasised the importance of data augmentation techniques, such as rotation and flipping, to improve the model's generalisation. Data augmentation increases the diversity of the training data, helping to reduce overfitting and improve the robustness of the model.

**Transfer Learning in Medical Image Classification:**

Transfer learning is a valuable strategy in medical imaging when labelled datasets are small and limited. Transfer learning entails using a pre-trained deep-learning model, such as a VGGNet, ResNet or Xception architecture, all of which have been trained on large image datasets such as ImageNet and fine-tuning it on the medical data that is much smaller in size (Esteva et al., 2017). Thanks to transfer learning, the knowledge learnt from large-scale images is borrowed and applied, which ultimately allows the model to reach a high level of performance with limited medical data.

Various studies apply transfer learning to brain tumour classification. For instance, Ismail et al. (2021) used a ResNet50 model pre-trained on ImageNet and fine-tuned on the brain tumour MRI dataset. They achieved high accuracy and, compared with traditional CNNs, their model worked much faster with a reduced training time. The authors concluded that their model shows a high degree of performance for brain tumour detection, and pre-trained models show how time-consuming and sophisticated such a task at its best would be for the biomedical community.

A study by Afshar et al. (2019) has been based on transfer learning and the Xception model for brain tumour classification, where the pre-trained Xception model has been trained on a brain tumour dataset and had surprisingly promising results on brain tumour classification. The authors note that transfer learning aids in identifying complex patterns in medical images, and this potency can be further enhanced using data augmentation regularisation and transfer learning.

**Data Augmentation and Preprocessing Techniques:**

Data augmentation is a technique that helps to improve the performance of deep learning models in medical imaging. It can artificially increase the volume and variety of training data by introducing artificial transformations to the training data, including rotation, scaling, brightness, etc., which can reduce overfitting and improve generalisation performance (Shorten and Khoshgoftaar, 2019). According to Pereira et al. (2016) and Hossain et al. (2020), it is indispensable to data augmentation for its high classification performance in brain tumour segmentation.

Besides data augmentation, normalisation, resizing and contrast adjustment, in which input images are standardised before training, are well-accepted and proper image preprocessing techniques (Litjens et al., 2017).

**Evaluation Metrics and Challenges:**

Brain tumour classification models must be carefully evaluated, and to do so, the outcomes of many models must be compared. A set of standard metrics exists to use in making such a comparison. Those include ‘accuracy’, ‘precision’, ‘recall’, ‘F1-score’, and the ‘area under the receiver operating characteristic curve’ (AUC-ROC). These are all measures of how often tumours are correctly identified versus the model producing false positives and false negatives. None is intrinsically superior to the others. (Jain et al., 2019)

Nonetheless, several challenges persist in deep learning-based brain tumour classification. The first problematic issue is linked to the imbalance of the datasets, meaning that the number of patient cases for some tumour types is much lower than for others. The imbalanced class distribution in the datasets may cause undesired biases in model generalisation, resulting in a model that performs well on the majority class but poorly on other rarer classes (Gupta et al., 2019). Various techniques, such as class weighting, oversampling or synthetic data generation, can be implemented to resolve this.

A further issue is the generalisability of models across datasets. Variations in scanners, protocols, and patients mean that not all images will be treated in the exact same way. One potential solution is domain adaptation a set of techniques that aim to transfer knowledge gained from one domain (e.g., hospital or scanner) to another (Ghafoorian et al., 2017).

**Conclusion:**

The literature on brain tumour classification has evolved significantly from traditional machine learning approaches reliant on handcrafted features to modern deep learning techniques like CNNs and transfer learning. While CNNs have demonstrated remarkable success in automating the feature extraction process, transfer learning has emerged as a valuable tool for improving model performance in scenarios with limited labelled data. Despite these advancements, challenges such as data imbalance and model generalisation remain active research areas. Future work in this field may address these challenges through innovative techniques such as domain adaptation and synthetic data generation.

# Chapter III

## Methodology:

This study classified brain-MRI images into four classes: glioma, meningioma, pituitary tumour, and no tumour, using a convolutional neural network (CNN). The brain-MRI data was analysed by implementing dataset preparation, data enhancement, designing a CNN model, training and evaluating the model, and other techniques to enhance the model.

**Dataset Preparation:**

The dataset consists of 4 classes, including brain MRI images stored in a training and testing directory. The training directory contains images of 4 Classes: glioma, meningioma, pituitary tumour, and no tumour. The last directory contains images of no tumour. The dataset was provided on a [Kaggle](https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/data) platform, a public MRI brain tumour dataset downloaded into the Google Colab environment and used to perform further operations.

Loading and Labelling: The dataset was loaded by iterating through the subdirectories representing each tumour class. All image paths were stored along with their respective class labels. The data was shuffled and converted into a Pandas Data Frame.

Distribution chart: Images were distributed among the four classes throughout the training phase, represented as a pie chart in Fig 3.1. The chart shows how the equal distribution of images in the training data might lead to class imbalance.

A diagram of a cancer type distribution

Description automatically generated

**Fig 3.1 Distribution of Brain Tumour Images**

**Data Splitting:**

We then split the dataset into training, validation, and testing sets using the ratio of 80:10:10. Then, we trained the model using the training set before evaluating it using the validation set. The classifier that achieved the lowest ARI on the validation set was then evaluated using the test set, allowing for a relatively robust model evaluation. To achieve these splits, we used the train\_test\_split function from the sci-kit-learn library, which also enabled the stratification of classes across splits.

**Data Augmentation:**

Data augmentation helps to improve training data diversity (so that a model learns more generalised patterns) and avoids overfitting, so this was done in two ways using the ImageDataGenerator class from Keras rescaling of the images from actual pixel values (0-255) to the shape (for the learning process) of values between 0 and 1, and to randomly increase or decrease the brightness of images within a specific range (in our case from 0.8 to 1.2) as shown in fig 3.2. A separate ImageDataGenerator instance was created for the training, validation, and test data. However, the level of augmentation was higher for training data compared to validation and test data (to avoid a problem called data contamination, which will cause the model to lose its ability to evaluate the targeted datasets during training properly). As a result of creating generators, as mentioned earlier, they fed the model with random augmented images during the learning (training) stage and the validation and testing of the trained model on the test set.

A collage of images of a brain

Description automatically generated

**Fig 3.2 Augmented Images for the Training the Models**

**CNN Model Design:**

The study's main objective was to train a convolutional neural network (CNN) and apply various testing procedures to categorise generated brain MRI images. We started by defining the CNN architecture using Keras Sequential API. The objective was to design a deep model through an organised sequence of stages, with three convolutional and a fully connected layer, as shown in Fig 3.3. A detailed layout of the model is presented below:

A colorful cube with different colored cubes

Description automatically generated with medium confidence

**Fig 3.3 Custom CNN Designed Architecture**

**Convolution Layers:** The model was composed of a few convolutional layers at the beginning of the network; the primary purpose of these convolutional layers was to process spatial feature extraction from the input images. ReLU activation and 2D convolution operations with a kernel size of 3x3 pixels were applied in each convolutional layer. Each convolutional layer started with 32 filters in the first convolution layer, and the filter number increased as we proceeded through the deeper layers up to 256 in the last convolutional layer. In the conviction layer, padding was set to 'same', which helped to maintain the spatial dimension of feature maps.

**Batch Normalization:** After every convolutional layer, we applied a batch normalisation operator to standardise the outputs and speed up the training. Additionally, the batch normalisation technique has been proven to help stabilise training and solve the vanishing or exploding gradient problem.

**Max Pooling:** Max pooling layers reduced the spatial dimensions of the feature maps to down-sample the net. The network used a pooling size of 2 x 2.

**Regularisation:** L2 regularisation was added to the convolutional layers to prevent overfitting. Here, model complexity is penalised, encouraging simpler models that generalise to unseen data.

**Fully Connected Layers:** Next, we flattened this vector of feature maps into a single stack and built two stacked fully connected layers with 512 and 256 units before the output layer of 13 units. Besides the usual ReLU activations along the stacks and the layers, we added dropout regularisation with a dropout rate of 0.5 in both fully connected layers. This is to prevent overfitting by temporarily disabling neurons randomly during training to prevent the network from overgeneralising patterns.

**Output Layer:** The CNN's output layer was a Soft Max layer with four fully connected units corresponding to the four classes (glioma, meningioma, pituitary tumour, and no tumour), with Soft Max activation for outputting the probability of each class.

The model was constructed using the Adamax optimiser, with the reduced learning rate set at 0.0001. It also uses categorical cross-entropy as a loss function and accuracy as an evaluation metric.

**Model Training:**

CNN was trained on the prepared data set using data generators with multiple parameters, which helped improve the model's performance. In our model, an early stopping has been introduced to monitor the validation loss by halting the training process whenever there is no decrease in the validation loss at five epochs, and the model weights were restored to the checkpoint of the minor loss. A learning rate reduction callback was also introduced to decrease the learning rate by 0.2, as shown in Fig 3.4.

A computer code with text

Description automatically generated with medium confidence

**Fig 3.4 Setting Up Callbacks and Training**

if the validation loss plateaus to improve the convergence of the model. Overall, it creates a good model by training it for up to epochs 50 with a batch size 32 parameter and monitoring the train and validation accuracy loss figuring, which is given below for easy reconciliation. The model has shown excellent accuracy, up to 99.80 % on the training data versus 96.34 % on the validation set. This indicates good performance and generalisability.

Then, on the test set, we check the performance of our model. We can calculate the test accuracy, which measures how well the model performs on unseen data. In our example, we achieved a final test accuracy of 96.51% with a test loss of 2.6987 if we added X\_test to the data. This test loss value represents our good performance on the test samples. We can visualise the model's classifications in a confusion matrix with four classes and calculate the precision, recall and F1-scores across all metrics, achieving good performance - perfect recall in the class 'no tumour'. It is also helpful to look at ROC curves, which we can plot for each class, showing the level of performance across varying thresholds for classifying our samples with high AUC values showing strong classification performance across class thresholds.

**Saving the Model:**

We saved the trained CNN model to disk as a single HDF5 file to use the model to predict new MRI scans without retraining the model.

**Transfer Learning:**

Also, I had a transfer learning experiment using CNN with Xception architecture, with pre-training on the ImageNet dataset, tweaking top layers to fully connected layers, and training it on a brain tumour dataset consisting of MRI data. This gives a test accuracy 98.94, better than the 96.51 accuracy of the custom CNN model. Also, I had ten epochs for the transfer learning model on the Adamax optimiser with a learning rate of 0.001.

**Additional Enhancement Techniques:**

These training techniques were used to ensure the models were robust and generalised. Cross-validation is one method that ensures we have a reasonable estimate of how the model will perform on an unseen data set. K-fold CV, which was not used in this study due to being computationally prohibitive, helps ensure the model's performance holds up on different splits of the data.

Furthermore, the data preprocessing steps included in reducing the computational complexity of the model were resizing the images to 128x128 pixels and normalising pixel values in the [0, 1] range. These steps also played a vital role. This was because, in many cases, the images were too large for the GPU memory, forcing the user to reduce the size of the images to make them runnable. In some instances, the image values were normalised because there was a range of values, and normalising the image values would help speed up training by eliminating values that the model was not even looking for. This was done to ensure consistency of the input data and guide the model in learning reliable and applicable features in the classification stage.

Finally, the fact that the research can be reproduced does offer something else. (Nominally, it does fit the bill.) Starting from the raw data, all the code and procedures were made freely available for replicating the experiment, along with details of the random data splits and model hyperparameters. This degree of disclosure, while still far from complete, at least leaves us with the possibility of someone else picking up the baton and following the study’s method.

**Conclusion:**

The proposed methodology in this study proves that CNNs and transfer learning are robust in classifying brain tumours from MRI images. Its custom CNN model performed well, and transfer learning with the Xception model increased accuracy. Overall, this study shows the promise of deep learning models as some of the recent applications in medical diagnosis, and more so in the automation of a diagnosis by altering the nature of imaging processes like brain tumour classification from MRI scans. The future of research in this area is implementing cross-validation, thorough data augmentation, and more extensive hyperparameter tuning to boost the model's effectiveness in balancing performance in a wide range of datasets.

# **Chapter IV**

## Analysing the Training Results

Through their training curves, the custom Convolutional Neural Network (CNN) and the transfer learning model (transfer learning uses the Xception architecture) offer crucial insights into the model's performance, learning pattern, and generalisation in its training lifecycle.

**Custom CNN Model:**

The figure below shows the change in accuracy and loss of the custom CNN model on training and validation sets across 50 epochs. The following are the main observations concerning the training and validation curves, as shown in Fig 4.1:

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

**Fig 4.1 Training Curves of Custom CNN Model**

**Training Accuracy and Loss:** As shown in the training accuracy curve, the model's accuracy improves over time, reaching almost 99.80% by epochs 50, which I find rather excellent compared to other models I have trained. At the same time, the training loss reduces dramatically as the model starts learning from the data and improving its error function value. This is because, for every layer of training, the CNN model successfully identifies critical features in the MRI images through convolutional operations and fine-tuning its predictions.

**Validation Accuracy and Loss:** The validation accuracy graph likewise shows significant improvement over the epochs, dropping at the end of the training to close to 96.34%, but the change is much laxer than the training loss. The validation loss also does not go down as sharply as the training loss, showing fluctuations that indicate that the model is likely going through some overfitting. However, we avoided severe overfitting since the best weights were restored thanks to the early stopping method based on the validation loss.

**Overfitting and Regularisation:** While the custom CNN model achieves high accuracy, the small gap between training and validation accuracy does suggest some overfitting. The application of L2 regularisation and dropout layers helped to minimise overfitting. However, the slight kink in the curves does suggest that the model could have still learned some idiosyncratic features in the training set that are not generalising perfectly to the validation set.

Considering all that, we did well. Here are the results for our custom CNN model: Notice that both training and validation accuracy look very good. Based on these training curves, we can say that the network learned a lot from the underlying patterns in the MRI images. It is possible, though, that these curves would be slightly better if we fine-tuned our model and implemented more regularisation.

**Transfer Learning Model (Xception Architecture)**

The transfer learning model (Xception architecture) showed different training dynamics from the custom CNN. It was pre-trained on a large-scale dataset (ImageNet) and fine-tuned on brain tumour datasets; thus, it had a faster and more stable learning process. The summary of training curves is as follows, as shown in Figure 4.2.

A graph of a training and training

Description automatically generated with medium confidence

**Fig 4.2 Transfer Learning Training Curves**

**Accuracy and Loss Curves during Training:** The training accuracy rapidly increases, achieving almost 99.45% accuracy within the first few epochs. The training loss also decreases rapidly within the first few epochs and quickly settles. This fast learning can be attributed to the fact that since we are using pre-trained weights of the base Xception model, it has already learnt a plethora of features of the ImageNet dataset, which are fine-tuned to the Brain Tumour dataset, thus enabling our model to learn the features of the new dataset faster.

**Validation Accuracy and Loss:** In the case of Xception, the validation accuracy also increases very quickly. and hits 98.94% accuracy on the last epoch. As for the validation loss, it also decreases smoothly and has fewer oscillations than for the custom CNN model, which is why it might generalise better on the validation set and be less likely to overfit (because it has many layers with stable weights and only a few of the layers are fine-tuned on our target dataset).

**Learning Rate and Training Duration:** The number of epochs needed for the transfer learning model to converge was less than that for the custom CNN. This is another feature of transfer learning which allows the model to leverage pre-trained knowledge to attain good performance with fewer data and iterations. The learning rate reduction callback further helped the model reduce the learning rate when the validation loss plateaued.

Overall, the transfer learning model performed consistently better than the custom CNN model regarding accuracy and stability. The figures of the graph of training curves convey that transfer learning with a pre-trained model works remarkably well for brain tumour classification, particularly when faced with limited labelled data. The smoother validation loss curve and higher final accuracy indicate a better generalisation of Xception model to unseen data.

## Evaluation of models on the test set

**Custom CNN Model:**

Once both the self-architected custom Convolutional Neural Network (CNN) and the transfer learning architecture based on Xception architecture were trained, the next step was to evaluate how well they performed on the test set to check their capacity to generalise the model to any new data. To do that, we arrive at various important aggregate metrics such as accuracy, loss, confusion matrix, precision, recall, F1-score, and ROC curves. To sum up, below is an exercise explaining the steps followed in the evaluation process involving both models. The accuracy of the test set is shown in Fig 4.3 below.

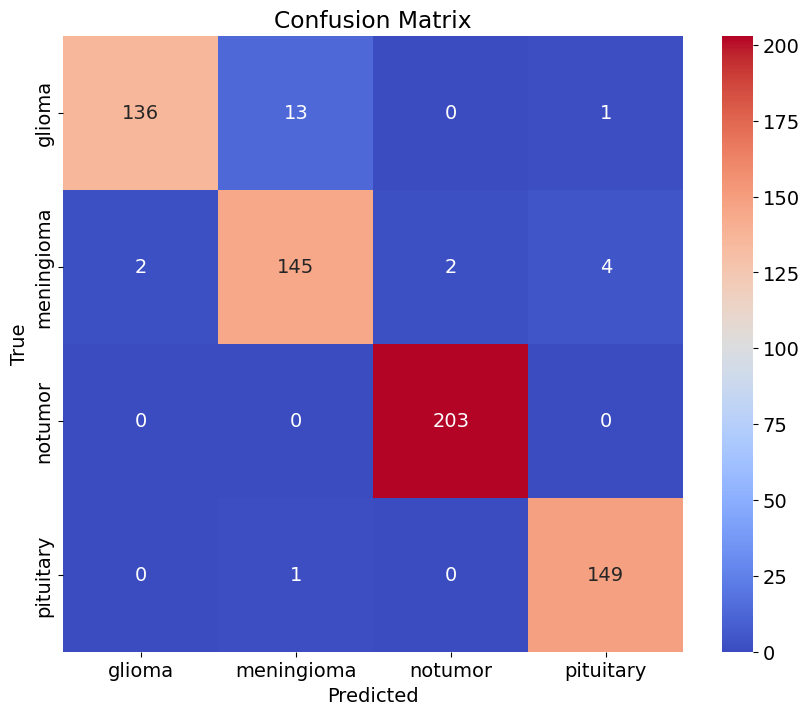
A screenshot of a computer code

Description automatically generated

**Fig 4.3 Custom Model Results on the Test Set**

**Accuracy and Loss:** Final test accuracy = 96.51%; final test loss = 2.6987. The model performed well in the test set, considering it predicted the suitable tumour class for most of the MRI images. The test set's accuracy was slightly lower than the validation set's (96.34 per cent). However, considering the negligible difference between the two validation numbers, we can argue that the model generalised well from the validation of the test set. The minimum test loss indicates that the CNN model was relatively robust in minimising classification errors.

**Confusion Matrix:** The confusion matrix is another view to assess the model performance more granularly. It reflects the number of correct and incorrect predictions for each class in the test dataset. We can see that the custom CNN model performs excellently in all four classes glioma, meningioma, pituitary tumour, and no tumour. Because glioma and meningioma sometimes have similar features in MRI images, it resulted in some misclassification among the three tumour types. However, as shown in Figure 4.4, the custom CNN model still has a high precision and recall for all classes.



**Fig 4.4 Confusion matrix for Custom CNN model**

**Precision, Recall, and F1-Score:** These are additional measures of the quality of the model’s performance. They tell us how well the model correctly detects positives (precision), how well it retrieves all relevant cases (recall), and how well it balances precision and recall (F1-score). The custom CNN achieved high precision, recall, and F1 scores in all four classes, with high performance close to perfect recall in the ‘no tumour’ class.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Precision | Recall | F1-Score |  |
| Glioma | 0.99 | 0.91 | 0.94 |  |
| Meningioma | 0.91 | 0.95 | 0.93 |  |
| No tumour | 0.99 | 1.00 | 1.00 |  |
| Pituitary | 0.97 | 0.99 | 0.98 |  |
| Accuracy |  |  |  | 0.96 |

**Table 4.1 Showing Performance matrix of custom CNN model**

**ROC Curves:** The ROC curves were plotted as shown in Fig 4.5 to examine the model’s performance at various thresholding levels. The ROC curves for the custom CNN mechanism demonstrate high AUCs across the tumour classes, specifying more robust discrimination between various tumour classes and healthy cases. Furthermore, the AUC values nearing 1.0 for most classes signify fewer false positives or false negatives made by the model, signifying the model’s reliability.

A graph of a function

Description automatically generated with medium confidence

**Fig 4.5 Roc Curve for Custom CNN Model**

**Transfer Learning Model (Xception Architecture)**

**Accuracy and Loss:** Test accuracy = 98.94%, Test loss = 0.0434. It is brilliant. The Xception model, as a transfer learning model, could generalise easily to the test set and classify most of the MRI data correctly. This is evident from the over 98% accuracy. The test loss was also very close to zero, whose value indicates that the transfer learning model can minimise the classification errors with the right kind of weight alterations and initialisation weights that can have a value of close to zero, as shown in Fig 4.6.

A screenshot of a computer code

Description automatically generated

**Fig 4.6 Accuracy and loss of Transfer Learning on test set**

**Confusion Matrix**: The confusion matrix for the Xception model also shows fewer misclassifications. The Xception model proved to be the best in classifying all tumour types with great accuracy, and there was never any confusion between similar classes. The confusion matrix shows that there was almost a perfect classification of the tumours in the 'no tumour' and 'pituitary tumour' classes. The Xception model was good at distinguishing between tumours of different types, as shown in Fig 4.7.

A diagram of a number of different colored squares

Description automatically generated with medium confidence

**Fig 4.7 Confusion matrix for Transfer Learning model**

**Precision, Recall, and F1-Score:** The transfer learning model recorded high precision, recall, and F1 scores for all four classes. The highest score for high performance is specifically for the ‘no tumour’ and ‘pituitary tumour’ categories, which depict the true positives and the prediction. Precision, Recall, and F1 scores pertain to false positives and false negatives. The highest precision, recall, and F1-scores show minimised false positive and false negative cases. The F1-scores, which balance precision and recall, depict that the model has maintained high accuracy, and the critical cases are not missed, as shown in Table 4.2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Precision | Recall | F1-Score |  |
| Glioma | 0.99 | 0.91 | 0.94 |  |
| Meningioma | 0.91 | 0.95 | 0.93 |  |
| No tumour | 0.99 | 1.00 | 1.00 |  |
| Pituitary | 0.97 | 0.99 | 0.98 |  |
| Accuracy |  |  |  | 0.96 |

**Table 4.2 Showing the Performance matrix of the Transfer Learning model**

**Class-wise ROC Curves:** The ROC curves for Xception reveal near-perfect AUC values across tumour-type classes as shown in fig 4.8. These values are almost 1.0 for classes and are highly desirable for making decisions, as classifying between tumour-type classes involves very few errors. From the smoothness and magnitude of the ROC curve, we could conclude that the Xception classifier model achieved almost perfect performance that maximised the actual positive rate (sensitivity) and false positive rate (specificity) for all the test samples.

A graph of a function

Description automatically generated with medium confidence

**Fig 4.8 ROC Curve for Transfer Learning Model**

# **Chapter V**

## Model Comparison:

Custom CNN Model: Although the custom CNN model performed well, it seemed to be slightly overfitting with the rise and fall of the validation loss and the slight dip in test accuracy. The model could classify brain tumours effectively but struggled to differentiate similar types of tumours from one another the most.

The Transfer Learning Model: The Xception with pre-trained features generalised well to the test set. It also took less time to train, converged faster, and was more accurate and stable across all evaluation metrics. This shows that pre-trained transfer learning applied to medical images is potent, especially in cases where the number of labelled training data is limited.

## Future Work:

Future research goals involve making the model more robust and generalisable by implementing k-fold cross-validation, which involves using multiple data splits with balanced sample sizes to derive more reliable estimates about the model's performance. More elaborate data augmentation pipelines will also be tested to increase accuracy, and other pre-trained architectures, such as EfficientNet, will also be investigated. Domain adaptation techniques by which a model trained on one task aids in solving another task will also be implemented to counteract the variability of datasets across MRI machines/institutions. Integrating an attention mechanism into the CNN structure might be interesting to better feature extraction of the tumours for improved classification. Testing on more extensive and diversified datasets will be required for further validation.

## Limitations:

Notwithstanding its promising results, this study has several significant limitations that could be addressed in future work but that reduce the generalisability of these models. First, the publicly available dataset included relatively few patients with manual segmentations, an MRI machine, and an imaging protocol that differed from other medical settings. Additionally, the custom CNN model showed some evidence of overfitting, meaning that regularisation measures may not have been fully effective. While introducing transfer learning with Xception improved performance, this technique remained limited by the features learned from a non-medical dataset (ImageNet). Lastly, computational limitations on available memory limited the implementation of more extensive cross-validation and hyperparameter-tuning paradigms. Future work on this problem must address these issues by utilising a larger, more diverse dataset and more robust evaluation.

## Conclusion:

This study successfully demonstrated the potential of deep learning techniques, particularly convolutional neural networks (CNNs) and transferred learning, for brain tumour classification using MRI images. The custom CNN model achieved strong performance, with a test accuracy of 96.51%, proving its ability to learn and generalise from the dataset. However, based on the Xception architecture, the transfer learning model significantly outperformed the custom model with a test accuracy of 98.94%, showcasing the advantages of leveraging pre-trained networks.

The results show that deep learning can be successfully employed to classify medical images and, ultimately, to help physicians diagnose patients more rapidly and with a higher accuracy. However, further challenges, such as overfitting in the custom CNN or the reliance on more extensive and more diverse datasets, remain. Transfer learning outperforms the custom CNN, exhibiting better generalisation and overall success. More medical data is necessary for medical professionals to hone their craft, but by using pre-trained models, AI can play a significant role in boosting performance.

In conclusion, this work emphasises deep learning and transfer learning, which are very effective. Their presence must be used in medical imaging, especially for the task of identifying brain tumours. They can also be the guideline for further research on how to improve model generalisation and accuracy, which can be considered for future studies.

# Chapter VI

## References

* Akkus, Z., Galimzianova, A., Hoogi, A., Rubin, D. L., & Erickson, B. J. (2017). Deep learning for brain MRI segmentation: state of the art and future directions. *Journal of Digital Imaging*, *30*(4), 449-459.
* Global Cancer Observatory (2020). Brain cancer statistics. International Agency for Research on Cancer. Available at: <https://gco.iarc.fr>
* Glover, G. H. (2011). Overview of functional magnetic resonance imaging. *Neurosurgery Clinics of North America*, *22*(2), 133-139.
* Lemasson, B., Valable, S., Farion, R., Krainik, A., & Barbier, E. L. (2013). MRI characterization of tumour microenvironment. *Journal of Neural Transmission*, *120*(5), 701-710.
* Leach, M. O., Brindle, K. M., & Howe, F. A. (2020). Advanced imaging techniques in cancer research. *Nature Reviews Cancer*, *20*(7), 442-454.
* Louis, D. N., Perry, A., Wesseling, P., Brat, D. J., Cree, I. A., Figarella-Branger, D., ... & Ellison, D. W. (2016). The 2016 World Health Organization classification of tumours of the central nervous system: a summary. *Acta Neuropathologica*, *131*(6), 803-820.
* Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., & Kirby, J. (2015). The multimodal brain tumour image segmentation benchmark (BRATS). *IEEE Transactions on Medical Imaging*, *34*(10), 1993-2024.
* Omuro, A., & DeAngelis, L. M. (2013). Glioblastoma and other malignant gliomas: a clinical review. *JAMA*, *310*(17), 1842-1850.
* Schwartz, M., Brown, T. J., Krings, T., & Thomas, A. J. (2017). Managing brain tumours: A comprehensive approach. *The Lancet Oncology*, *18*(3), e159-e168.
* Villanueva-Meyer, J. E., Mabray, M. C., & Cha, S. (2017). Current clinical brain tumour imaging. *Neurosurgery*, *81*(3), 397-415.
* Weller, M., van den Bent, M., Hopkins, K., Tonn, J. C., Stupp, R., Falini, A., ... & Reifenberger, G. (2015). EANO guidelines for the diagnosis and treatment of anaplastic gliomas and glioblastoma. *The Lancet Oncology*, *15*(9), e262-e272.
* Zwanenburg, A., Leger, S., Vallières, M., & Löck, S. (2020). Image biomarker standardisation initiative. *Radiology*, *295*(2), 328-338.
* Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A., Ciompi, F., Ghafoorian, M., & Van Ginneken, B., A survey on deep learning in medical image analysis, *Medical Image Analysis*, 2017, 42, pp. 60-88.
* Topol, E. J., High-performance medicine: the convergence of human and artificial intelligence, *Nature Medicine*, 2019, 25(1), pp. 44-56.
* European Society of Radiology (ESR), The current status of radiological clinical audit: an ESR perspective, *Insights into Imaging*, 2019, 10(1), pp. 1-16.
* Mazurowski, M. A., Buda, M., Saha, A., & Bashir, M. R., Deep learning in radiology: An overview of the concepts and a survey of the state of the art, *Radiology*, 2019, 292(2), pp. 330-344.
* Afshar, P., Mohammadi, A., Plataniotis, K.N. and Oikonomou, A., 2019. Brain tumor type classification via capsule networks. *Proceedings - International Conference on Image Processing, ICIP*, 2019-September, pp. 1364–1368.
* Chaplot, S., Patnaik, L.M. and Jagannathan, N.R., 2006. Classification of Magnetic Resonance Brain Images Using Wavelets as Input to Support Vector Machine and Neural Network. *Biomedical Signal Processing and Control*, 1(1), pp. 86-92.
* Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M. and Thrun, S., 2017. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639), pp.115-118.
* Ghafoorian, M., Karssemeijer, N., Heskes, T., Barontini, S., Išgum, I., Andriessen, T.M., de Leeuw, F.-E. and Platel, B., 2017. Transfer Learning for Domain Adaptation in MRI: Application in Brain Lesion Segmentation. *MICCAI 2017: Medical Image Computing and Computer Assisted Intervention*, Lecture Notes in Computer Science, vol 10434. Springer, pp. 516-524.
* Gupta, A., Sivapriya, S. and Sharma, V., 2019. Addressing Data Imbalance Problem in Medical Image Classification. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 8(12), pp. 1691-1696.
* Hossain, A., Park, S., and Kim, J.-M., 2020. Brain Tumor Classification Using Convolutional Neural Network and Deep Learning. *Proceedings of the 2020 11th International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan, 7-9 April 2020. IEEE, pp. 226-230.
* Ismail, R.I., Halim, Z. and Karim, M.R., 2021. Brain Tumor Classification Using Transfer Learning and Fine-Tuning. *Journal of King Saud University - Computer and Information Sciences.*

# **Chapter VII**

## Appendix:

# -\*- coding: utf-8 -\*-

"""Brain\_Tumor\_Classification.ipynb

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/1JGacu\_7lS8uZOmn8UEBL4zUjkHPzPnxn

# Importing the Libraries

"""

# To visualize the network in 3D

!pip install visualkeras

# For Data Processing

import numpy as np # Core library for numerical computations with support for arrays and matrices

import pandas as pd # Powerful data manipulation and analysis library, useful for handling data in tabular form (DataFrames)

from sklearn.utils import shuffle # Utility to shuffle the dataset, ensuring randomization

from sklearn.model\_selection import train\_test\_split # Function to split dataset into training and testing sets

from sklearn.metrics import roc\_curve, auc # Functions for computing Receiver Operating Characteristic (ROC) curves and the Area Under the Curve (AUC)

from sklearn.preprocessing import label\_binarize # Utility to convert class labels to a binary matrix form for multi-class classification

from sklearn.metrics import confusion\_matrix, classification\_report # Tools for evaluating model performance through confusion matrix and detailed classification report

from tqdm import tqdm # Library to create progress bars for loops, useful for tracking the progress of tasks

import os # Module to interact with the operating system, useful for file and directory operations

import random # Provides functions for generating random numbers, useful for data augmentation and splitting

from PIL import Image, ImageEnhance, ImageOps, ImageFont # Python Imaging Library for opening, manipulating, and saving images

from tensorflow.keras.preprocessing.image import load\_img, ImageDataGenerator # Functions for image loading and real-time data augmentation

from tensorflow.keras.layers import BatchNormalization # Layer that normalizes activations, stabilizing and speeding up training

# For Data Visualization

import matplotlib.pyplot as plt # Library for creating static, animated, and interactive visualizations in Python

import seaborn as sns # Data visualization library based on Matplotlib, provides a high-level interface for drawing attractive statistical graphics

from itertools import cycle # Utility to cycle through elements, commonly used for color cycling or repetitive tasks

# For Machine Learning and Deep Learning

import tensorflow as tf # Open-source platform for machine learning and deep learning models

from tensorflow.keras.regularizers import l2 # Regularization technique to prevent overfitting by penalizing large weights

import visualkeras # Visualization library for displaying the architecture of neural networks.

from tensorflow.keras.models import Sequential # Sequential model type for stacking layers linearly in Keras

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau # Callbacks for early stopping, saving best model checkpoints, and reducing learning rate when a metric plateaus

from tensorflow.keras.layers import BatchNormalization # Batch normalization layer to stabilize and accelerate training by normalizing the input of each layer

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout # Commonly used layers in Convolutional Neural Networks (CNNs) for image classification

from tensorflow.keras.optimizers import Adam, Adamax # Optimization algorithms used for minimizing the loss function and updating model weights

"""# Loading the Dataset"""

from google.colab import drive

drive.mount('/content/drive')

train\_dir = '/content/drive/MyDrive/MRI Brain Tumor Dataset/Training/'

test\_dir = '/content/drive/MyDrive/MRI Brain Tumor Dataset/Testing/'

train\_paths = []

train\_labels = []

# Iterate over each label (subdirectory) in the training directory

for label in os.listdir(train\_dir):

# Iterate over each image file in the label subdirectory

for image in os.listdir(train\_dir+label):

# Append the full image path to the train\_paths list

train\_paths.append(train\_dir+label+'/'+image)

# Append the label to the train\_labels list

train\_labels.append(label)

train\_paths, train\_labels = shuffle(train\_paths, train\_labels)

# Convert to DataFrame

train\_df = pd.DataFrame({

'Image Path': train\_paths,

'Class Label': train\_labels

})

"""# Distribution Of Image Data"""

def plot\_tumor\_distribution(train\_labels):

plt.figure(figsize=(8, 6))

colors = ['#4285f4', '#ea4335', '#fbbc05', '#34a853']

plt.rcParams.update({'font.size': 14})

labels = ['pituitary', 'notumor', 'meningioma', 'glioma']

sizes = [len([x for x in train\_labels if x == label]) for label in labels]

plt.pie(sizes, labels=labels, colors=colors, autopct='%.1f%%',

explode=(0.025, 0.025, 0.025, 0.025), startangle=30)

plt.title('Tumor Type Distribution')

plt.show()

plot\_tumor\_distribution(train\_labels)

test\_paths = []

test\_labels = []

for label in os.listdir(test\_dir):

for image in os.listdir(test\_dir+label):

test\_paths.append(test\_dir+label+'/'+image)

test\_labels.append(label)

test\_paths, test\_labels = shuffle(test\_paths, test\_labels)

# Convert to DataFrame

test\_df = pd.DataFrame({

'Image Path': test\_paths,

'Class Label': test\_labels

})

"""\*\*Distribution of Data in training and testing\*\*"""

plt.figure(figsize=(8,6))

colors = ['#4285f4', '#ea4335', '#fbbc05', '#34a853']

plt.rcParams.update({'font.size': 14})

plt.pie([len(train\_labels), len(test\_labels)],

labels=['Train','Test'],

colors=colors, autopct='%.1f%%', explode=(0.05,0),

startangle=30);

"""# Spliting the Data"""

valid\_df, ts\_df = train\_test\_split(test\_df, train\_size=0.5, random\_state=20, stratify=test\_df['Class Label'])

# Parameters

batch\_size = 32

img\_size = (128, 128)

brightness\_range = (0.8, 1.2)

rescale\_factor = 1/255

num\_classes = 4 # Example number of classes

epochs = 10 # Number of epochs in training

"""# Data Augmentation"""

# Define data augmentation parameters for training

train\_data\_gen\_args = dict(

rescale=rescale\_factor,

brightness\_range=brightness\_range,

)

# Define data augmentation parameters for validation and testing (usually less augmentation)

valid\_test\_data\_gen\_args = dict(rescale=rescale\_factor)

# Create ImageDataGenerators with parameters

train\_val\_gen = ImageDataGenerator(\*\*train\_data\_gen\_args)

valid\_test\_gen = ImageDataGenerator(\*\*valid\_test\_data\_gen\_args)

# Function to create generators from dataframes

def create\_generators(tr\_df, valid\_df, ts\_df, batch\_size, img\_size):

if tr\_df.empty or valid\_df.empty or ts\_df.empty:

raise ValueError("One of the dataframes is empty. Please check your dataframes.")

train\_gen = train\_val\_gen.flow\_from\_dataframe(

tr\_df,

x\_col='Image Path',

y\_col='Class Label',

batch\_size=batch\_size,

target\_size=img\_size,

class\_mode='categorical' # Assuming you have categorical labels

)

valid\_gen = valid\_test\_gen.flow\_from\_dataframe(

valid\_df,

x\_col='Image Path',

y\_col='Class Label',

batch\_size=batch\_size,

target\_size=img\_size,

class\_mode='categorical'

)

test\_gen\_flow = valid\_test\_gen.flow\_from\_dataframe(

ts\_df,

x\_col='Image Path',

y\_col='Class Label',

batch\_size=16,

target\_size=img\_size,

class\_mode='categorical',

shuffle=False

)

return train\_gen, valid\_gen, test\_gen\_flow

# Create generators

train\_gen, valid\_gen, test\_gen\_flow = create\_generators(train\_df, valid\_df, ts\_df, batch\_size, img\_size)

# Visualize some augmented images from the training generator

images, labels = next(train\_gen)

class\_names = list(train\_gen.class\_indices.keys())

plt.figure(figsize=(20, 20))

for i in range(min(len(images), 16)):

plt.subplot(4, 4, i + 1)

plt.imshow(images[i])

class\_name = class\_names[np.argmax(labels[i])]

plt.title(class\_name, color='k', fontsize=15)

plt.axis('off')

plt.show()

"""# CNN Model Design"""

# Model Architecture

model = Sequential()

# Convolutional layers with L2 regularization and Batch Normalization

model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input\_shape=(128, 128, 3), kernel\_regularizer=l2(0.002)))

model.add(BatchNormalization())

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel\_regularizer=l2(0.001)))

model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel\_regularizer=l2(0.001)))

model.add(BatchNormalization())

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel\_regularizer=l2(0.001)))

model.add(BatchNormalization())

model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel\_regularizer=l2(0.001)))

model.add(BatchNormalization())

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (3, 3), activation='relu', padding='same', kernel\_regularizer=l2(0.002)))

model.add(BatchNormalization())

model.add(Conv2D(256, (3, 3), activation='relu', padding='same', kernel\_regularizer=l2(0.002)))

model.add(BatchNormalization())

model.add(MaxPooling2D((2, 2)))

# Flatten the output

model.add(Flatten())

# Fully connected layers with increased dropout and L2 regularization

model.add(Dense(512, activation='relu', kernel\_regularizer=l2(0.002)))

model.add(Dropout(0.5)) # Increased dropout rate

model.add(Dense(256, activation='relu', kernel\_regularizer=l2(0.002)))

model.add(Dropout(0.5)) # Increased dropout rate

# Output layer

model.add(Dense(num\_classes, activation='softmax'))

# Compile the model with a reduced learning rate

model.compile(optimizer=Adamax(learning\_rate=1e-4),

loss='categorical\_crossentropy',

metrics=['accuracy'])

"""# Model Summery"""

# Display the model summary

model.summary()

# Let's visualize the model

font = ImageFont.truetype("/content/drive/MyDrive/Sunshiney-Regular.ttf", 32) # using comic sans is strictly prohibited!

visualkeras.layered\_view(model, legend=True, font=font) # font is optional!

# Callbacks

early\_stopping = EarlyStopping(monitor='val\_loss', patience=5, restore\_best\_weights=True)

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.2, patience=2, min\_lr=1e-6)

# Training the model with data augmentation

history = model.fit(train\_gen,

epochs=50,

validation\_data=valid\_gen,

callbacks=[early\_stopping, reduce\_lr])

"""## Traning Curves"""

def plot\_training\_history(history):

# Extract values from the history object

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

# Plot training and validation accuracy values

plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)

plt.plot(epochs, acc, 'g', label='Training accuracy')

plt.plot(epochs, val\_acc, 'r', label='Validation accuracy')

plt.title('Training and Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

# Plot training and validation loss values

plt.subplot(1, 2, 2)

plt.plot(epochs, loss, 'g', label='Training loss')

plt.plot(epochs, val\_loss, 'r', label='Validation loss')

plt.title('Training and Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

# Usage

plot\_training\_history(history)

"""# Model Evaluation"""

# Evaluate the model on the test dataset

final\_test\_loss, final\_test\_accuracy = model.evaluate(test\_gen\_flow, verbose=1)

print(f'Final Test Loss: {final\_test\_loss}')

print(f'Final Test Accuracy: {final\_test\_accuracy}')

# Predict on the test dataset

final\_test\_predictions = model.predict(test\_gen\_flow)

predicted\_classes = np.argmax(final\_test\_predictions, axis=1)

true\_classes = test\_gen\_flow.classes

"""# Confusion Matrix"""

def plot\_confusion\_matrix(true\_classes, pred\_classes, class\_names):

cm = confusion\_matrix(true\_classes, pred\_classes)

plt.figure(figsize=(10, 8))

sns.heatmap(cm, annot=True, fmt='d', cmap='coolwarm', xticklabels=class\_names, yticklabels=class\_names)

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()

# Get class names

class\_names = list(test\_gen\_flow.class\_indices.keys())

# Plot confusion matrix

plot\_confusion\_matrix(true\_classes, predicted\_classes, class\_names)

# Print classification report

print(classification\_report(true\_classes, predicted\_classes, target\_names=class\_names))

"""# ROC Curves"""

# Function to plot ROC curves for multi-class classification

def plot\_roc\_curves(true\_classes, pred\_probabilities, class\_names):

# Binarize the true classes

true\_classes\_binarized = label\_binarize(true\_classes, classes=range(len(class\_names)))

n\_classes = len(class\_names)

# Calculate ROC curve and AUC for each class

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(n\_classes):

fpr[i], tpr[i], \_ = roc\_curve(true\_classes\_binarized[:, i], pred\_probabilities[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

# Plot all ROC curves

plt.figure(figsize=(10, 8))

colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'darkred', 'green', 'purple', 'brown'])

for i, color in zip(range(n\_classes), colors):

plt.plot(fpr[i], tpr[i], color=color, lw=2,

label=f'ROC curve of class {class\_names[i]} (area = {roc\_auc[i]:0.2f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curves')

plt.legend(loc='lower right')

plt.show()

# Get class names

class\_names = list(test\_gen\_flow.class\_indices.keys())

# Predict probabilities for the ROC curve

predicted\_probabilities = model.predict(test\_gen\_flow)

# Plot ROC curves

plot\_roc\_curves(true\_classes, predicted\_probabilities, class\_names)

"""# Saving the CNN Model"""

# Save the entire model to a file

model.save('/content/drive/MyDrive/CNN\_model.h5')

"""# Transfer Learning"""

img\_shape=(128,128,3)

base\_Model = tf.keras.applications.Xception(include\_top= False,weights= "imagenet",

input\_shape= img\_shape, pooling= 'max')

"""# Defining Model"""

Model\_tr = Sequential([

base\_Model,

Dropout(rate= 0.5),

Dense(128, activation= 'relu'),

Dropout(rate= 0.25),

Dense(4, activation= 'softmax')

])

"""# Compiling the Model"""

Model\_tr.compile(Adamax(learning\_rate= 0.001),

loss= 'categorical\_crossentropy',

metrics= ['accuracy'])

"""# Training the Model"""

historyy=Model\_tr.fit(train\_gen,epochs=10,

validation\_data=valid\_gen,

shuffle=False)

"""# Ploting the Curves"""

def plot\_training\_history(history):

# Extract values from the history object

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

# Plot training and validation accuracy values

plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)

plt.plot(epochs, acc, 'g', label='Training accuracy')

plt.plot(epochs, val\_acc, 'r', label='Validation accuracy')

plt.title('Training and Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

# Plot training and validation loss values

plt.subplot(1, 2, 2)

plt.plot(epochs, loss, 'g', label='Training loss')

plt.plot(epochs, val\_loss, 'r', label='Validation loss')

plt.title('Training and Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

# Usage

plot\_training\_history(history)

"""# Model Evaluation"""

# Evaluate the model on the test dataset

final\_test\_loss, final\_test\_accuracy = Model\_tr.evaluate(test\_gen\_flow, verbose=1)

print(f'Final Test Loss: {final\_test\_loss}')

print(f'Final Test Accuracy: {final\_test\_accuracy}')

# Predict on the test dataset

final\_test\_predictions = Model\_tr.predict(test\_gen\_flow)

predicted\_classes = np.argmax(final\_test\_predictions, axis=1)

true\_classes = test\_gen\_flow.classes

"""# Confusion Matrix"""

def plot\_confusion\_matrix(true\_classes, pred\_classes, class\_names):

cm = confusion\_matrix(true\_classes, pred\_classes)

plt.figure(figsize=(10, 8))

sns.heatmap(cm, annot=True, fmt='d', cmap='coolwarm', xticklabels=class\_names, yticklabels=class\_names)

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()

# Get class names

class\_names = list(test\_gen\_flow.class\_indices.keys())

# Plot confusion matrix

plot\_confusion\_matrix(true\_classes, predicted\_classes, class\_names)

# Print classification report

print(classification\_report(true\_classes, predicted\_classes, target\_names=class\_names))

"""# Roc Curves"""

# Function to plot ROC curves for multi-class classification

def plot\_roc\_curves(true\_classes, pred\_probabilities, class\_names):

# Binarize the true classes

true\_classes\_binarized = label\_binarize(true\_classes, classes=range(len(class\_names)))

n\_classes = len(class\_names)

# Calculate ROC curve and AUC for each class

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(n\_classes):

fpr[i], tpr[i], \_ = roc\_curve(true\_classes\_binarized[:, i], pred\_probabilities[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

# Plot all ROC curves

plt.figure(figsize=(10, 8))

colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'darkred', 'green', 'purple', 'brown'])

for i, color in zip(range(n\_classes), colors):

plt.plot(fpr[i], tpr[i], color=color, lw=2,

label=f'ROC curve of class {class\_names[i]} (area = {roc\_auc[i]:0.2f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curves')

plt.legend(loc='lower right')

plt.show()

# Get class names

class\_names = list(test\_gen\_flow.class\_indices.keys())

# Predict probabilities for the ROC curve

predicted\_probabilities = Model\_tr.predict(test\_gen\_flow)

# Plot ROC curves

plot\_roc\_curves(true\_classes, predicted\_probabilities, class\_names)

"""# Saving the Transfer Learning Model"""

# Save the entire model to a file

Model\_tr.save('/content/drive/MyDrive/tr\_model.h5')