Brain Tumor Detection and Classification by Using CNN

Undergraduate Graduation Project (Academic Thesis)

by

MD KAMRUL ISLAM

Student ID: 2017521460107

mdkamrul.islam@hotmail.com

Under the supervision of

Prof. Chen Jie

Date of Creation: 15th December, 2021

Abstract

Tumors within the brain manifest as either malignant or benign accumulations of cells that deviate from normal growth patterns. Such deviations result in the formation of tumors, which may be classified into benign types, for example, meningiomas and pituitary tumors, or malignant types, like gliomas. Furthermore, brain cancers are mostly categorized into primary and secondary based on their growth origin. In accordance with the World Health Organization, brain tumors constitute less than 2% of all cancer cases, yet they are associated with significant morbidity and mortality Stewart and Wild (2014). A comprehensive cross-sectional analysis has revealed that primary brain tumors are responsible for an estimated 21,215 (ranging from 10,427 to 43,165) fatalities annually in China. The appearance of deep learning technologies has markedly advanced the field of medical diagnostics through enhanced medical imaging capabilities, particularly in identifying and diagnosing various diseases. Within this context, the CNNs have evolved as the leading machine learning algorithm for tasks involving visual learning and image recognition. Our study presents a novel CNN framework that is specifically tailored to classify brain cancers using T1-weighted contrast enhanced MRI into 4 categories: meningioma, glioma, pituitary tumor, and lack of tumor. The results show that our CNN model not only outperforms other well-known models like VGG-16 (94%), Xception (97%), ResNet-50 (94%), and Inception-V3 (96%) in terms of validation accuracy, but it also runs with much less computational overhead.

[Key Words] Deep Learning, CNN, Transfer Learning, Xception, VGG-16, ResNet50, Inception-v3

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
CNN	Convolutional Neural Network
ECG	Electrocardiogram
AI	Artificial Intelligence
WHO	World Health Organization
DNN	Deep Neural Network
CBICA	Center for Biomedical Image Computing & Analytics
GPU	Graphics Processing Unit
HGG	High Grade Glioma
CT	Computed Tomography
LGG	Low Grade Glioma
GBM	Glioblastoma Multiform
PET	Position Emission Tomography
FLAIR	Fluid attenuated inversion recovery
RMSprop	Root Mean square Propagation
ADAM	Adaptive Moment Optimization
ReLU	Rectified Linear Unit
CPU	Central Processing Unit
MRI	Magnetic Resonance Imaging
SGD	Stochastic Gradient Descent

1 Introduction

1.1 Introduction

The human brain, a critical organ responsible for regulating fundamental bodily functions and characteristics, is highly susceptible to tumors, which can significantly impact health and well-being. In accordance with the National Brain Tumor Society, approximately 700,000 individuals in the United States are currently have a brain tumor, a number projected to increase to 787,000 by the year 2020 National Brain Tumor Society (2020). In the contemporary era, the prevalence of diseases, including tumors, has been escalating. Tumors, defined by the abnormal growth of tissues, can arise in any part of the body, potentially altering the normal function and structure. Brain tumors, particularly, pose a formidable challenge due to their location and the complexity of treatment. These tumors are broadly categorized into primary and secondary types, with primary tumors originating within the brain and secondary tumors metastasizing from other body parts. Furthermore, brain tumors are differentiated into benign, which are non-cancerous, and malignant, which contain cancerous cells and present a greater risk to life. Detecting and classifying brain tumors is a complex process that is crucial for effective treatment.

Brain tumors, although less common than breast or lung cancer, present a substantial global health burden, ranking as the 10th leading cause of mortality. These tumors arise from abnormal tissue growth within the brain or central spinal area, disrupting normal neurological function. Categorized as either benign or malignant, brain tumors exhibit distinct characteristics: benign tumors are slow-growing and lack cancerous cells, whereas malignant tumors are aggressive, cancerous, and capable of metastasizing to other brain and spinal regions. Meningiomas and pituitary tumors are two common forms of brain tumors, as are gliomas, which develop from brain tissue rather than nerve cells or blood vessels. Meningiomas grow from the membranes that surround the brain and central nervous system, while pituitary tumors arise within the skull. Meningiomas are mostly benign, gliomas are usually malignant, and pituitary tumors, even if benign, can cause serious health problems Cancer Treatments Centers of America (2019); American Association of Neurological Surgeons (2019). Accurately identifying between various tumor forms is an important step in clinical diagnosis and patient care.

Malignant brain tumors pose a serious threat to life, ranking as the tenth greatest cause of death according to recent data. The five-year survival rate for brain and nervous system malignancies is stated to be 34% for men and 36% for women National Health Service (2020). Brain tumors are diagnosed using a variety of procedures, including CT scans and EEGs,

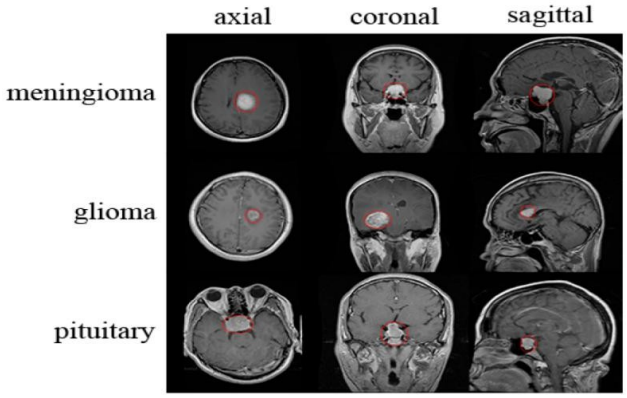

Figure 1.1: Different Types of Tumors in the Brain

but Magnetic Resonance Imaging (MRI) is the most successful and extensively used method. MRI uses intense magnetic fields and radio waves to provide detailed inside organ images that are clearer than other imaging modalities like CT scans and EEGs. Advancements in machine learning and deep learning have transformed medical imaging, allowing for early and efficient disease identification, which was previously difficult and time-consuming. The growing demand for computer-aided diagnostic tools underscores the importance of developing efficient and accurate methods for diagnosing critical conditions like brain tumors. Early detection and classification of brain tumors through medical imaging represent crucial research endeavors, enabling the selection of optimal treatment strategies to improve patient outcomes. Challenges in classifying and detecting MRI images using neural networks often stem from the size of the database and variations in image acquisition planes. Preprocessing plays a vital role in mitigating overfitting risks before conducting image analysis. However, convolutional neural networks (CNNs) offer a significant advantage by integrating pre-processing and feature extraction steps seamlessly.

This study investigates a method for classifying and detecting brain tumors into multiple categories—meningioma, glioma, pituitary, and no tumor—by employing a data augmentation approach and a CNN model. The performance of this approach is compared with other pre-trained models such as Xception, VGG-15, ResNet50, and Inception-v3.

1.2 Motivation

The impetus for developing advanced brain tumor detection methodologies transcends mere identification; it encompasses the ability to differentiate among various tumor types. This distinction is critical, as precise tumor classification can significantly influence treatment strategies. The project at hand introduces a computational system capable of detecting tumor presence in MRI scans and accurately determining the tumor category through the

application of Convolutional Neural Networks (CNNs). This technological advancement promises to enhance the accuracy of diagnoses provided by medical professionals, facilitating a more efficient and targeted approach to treatment.

1.3 Main Work of This Project

This paper's main objective is to investigate the efficacy of a CNN model, alongside established pre-trained models (VGG-16, Xception, ResNet50, and Inception-V3), in classifying three distinct types of brain tumors, as well as cases with no tumor, utilizing a relatively limited dataset. Despite the dataset's constraints, image augmentation techniques were employed to expand the number of images available for training. This research endeavors to demonstrate that a streamlined CNN architecture can achieve comparable, if not superior, performance to its more complex counterparts. The significance of utilizing a less resource-intensive network cannot be overstated, particularly in contexts where computational resources are limited, such as in mobile diagnostics or in medical facilities with restricted technological infrastructure.In addition, the purpose of this work is to test the efficacy of a categorical-cross-entropy-validation method for use in future practical applications and to evaluate the proposed network's generalizability in clinical situations. A novel CNN model was developed for this purpose, utilizing datasets from Kaggle and figshare, with the findings articulated through confusion matrices and accuracy metrics. The comparative analysis includes transfer learning models pre-trained on extensive datasets like ImageNet, highlighting the proposed model's relative accuracy through both testing and validation metrics.

1.4 Objective and Aim

The integration of deep-learning methodologies within healthcare diagnostics marks a significant transition towards more impactful diagnostic solutions. As outlined by the World Health Organization (WHO), achieving precise brain tumor diagnosis entails not only tumor detection but also detailed classification based on factors such as malignancy, grade, and type. This research initiative, centered on Magnetic Resonance Imaging (MRI) for brain tumor analysis, aims not only to detect tumors but also to accurately classify them according to grade, class, and specific location. To enhance the efficiency and accuracy of diagnosis, the project adopts a unified approach employing five distinct models for classification, rather than individual models for each category. This strategy leverages a Convolutional Neural Network (CNN)-based multi-class classification system tailored to effectively detect and classify brain tumors. Such an approach empowers medical professionals in diagnosing tumors and comprehending their underlying characteristics, potentially leading to life-saving early

interventions and informed treatment planning. The overarching objective of this application is to ensure prompt and precise tumor identification, thereby expediting the treatment process and enhancing patient outcomes. Designed to overcome the limitations of manual diagnostic methods, this user-friendly application offers a swifter, more accurate, and efficient solution for both healthcare providers and patients alike.

1.5 Organization and Structure

This document is systematically structured into chapters, each dedicated to a specific aspect of the research, as outlined below:

Chapter	**Content**
Chapter 01	This initial chapter presents an overview of the essential research background related to types of human brain tumors. It also reviews various computational systems developed to address these health issues, offering a critical analysis of their effectiveness and limitations.
Chapter 02	Provides a thorough evaluation of the literature, critically analyzing earlier research, approaches, and conclusions pertinent to the computational diagnosis and categorization of brain tumors.
Chapter 03	explains to the reader the basic principles of brain tumors, the workings of Magnetic Resonance Imaging (MRI), and the basic algorithms used in tumor detection and classification.
Chapter 04	Details the methodology of the project, encompassing the construction of the CNN model, integration of pre-trained models, the workflow of the system, and the approach taken for training and evaluation to derive the final results.
Chapter 05	Explains the project's experimental setup in detail, covering the environment setup, dataset preparation, image processing methods, data augmentation tactics, confusion matrix application for evaluation, and a discussion of the testing processes that led to the final conclusions.
Chapter 06	Ends the thesis with a synopsis of the results, contributions to the area, possible ramifications for further study, and an extensive list of all the references used in the writing.

Table 1.1: Structure of the Thesis

2 Literature Review

In the realm of image processing, deep learning (DL) and artificial intelligence (AI) approaches are mostly applied to tasks such as MRI scan identification, segmentation, and classification, particularly concerning the categorization and detection of brain tumors. Consequently, a large body of scholarly literature has concentrated on the segmentation and categorization of MRI images that depict brain regions.Sheikh Basheera et al. Basheera and Ram (2019), for instance, presented a method for classifying tumors. This method uses a pre-trained convolutional neural network (CNN) that use stochastic gradient descent optimization to identify the tumor location in an MRI scan and extract it. These contributions are a small part of the vast body of work that is now available globally on deep learning-driven methods for the identification and categorization of brain tumors. Moreover, Muhammad Sajjad and colleagues Sajjad et al. (2019) demonstrate the ability to classify tumors of different grades through the use of an MRI image data augmentation technique that is further enhanced through training with the VGG-19 CNN architecture. A methodology for classifying pituitary adenoma tumors in MRI images was developed by Carlo, Ricciardi, et al. Carlo et al. (2019). It makes use of multinomial logistic regression and k-nearest neighbor algorithms. With an exceptional AUC of 98.4%, a multinomial logistic regression accuracy of 83%, and a k-nearest neighbor test accuracy of 92%, the system functioned well. The AlexNet convolutional neural network (CNN) model was utilized by Khwaldeh, Saed, et al. Khawaldeh et al. (2018) to develop an architecture for classifying brain MRI scans into healthy and sick classes, as well as a grading system for differentiating between low- and high-quality brain pictures.Their method produced a 91% classification accuracy.Nyoman Abiniwanda et al. Abiwinanda et al. (2019) used a convolutional neural network to classify meningioma, glioma, and pituitary tumors—three distinct forms of brain cancer. The model attained 84.19% validation accuracy and 98.51% training accuracy.The model's training accuracy was 98.51%, while its validation accuracy was 84.19%. Sunanda Das et al. Das et al. (2019) used image processing approaches to train a CNN model for brain cancer classification, achieving an average accuracy of 93.33% on validation and 94.39% on training datasets.Romeo, Valeria, and colleagues proposed a machine learning method to predict tumor grades and nodal status of brain tumors from initial CT scans utilizing radio mic data Romeo et al. (2020). Their approach attained an accuracy of 92.9% utilizing k-nearest neighbor and Naive Bayes algorithms. Muhammad Talo et al. Talo et al. (2019) obtained a perfect accuracy of 100% in categorizing normal and sick brain MRI images using the ResNet34 pre-trained CNN model combined with transfer and data augmentation techniques. Arshia Rehman et al. (2020) classified pituitary, glioma, and meningioma brain cancers using three pre-trained CNN models: VGG16, AlexNet, and GoogleNet. The VGG16 model achieved

the maximum accuracy of 98.67% with transfer learning. Ahmet Çinar et al. Cinar and Yldrm (2020) improved the pre-trained ResNet50 CNN model by altering its design. Their deep neural network (DNN) attained an accuracy of 96.97 percent. The literature intensively investigates the use of pre-trained networks and additional algorithms for tasks such as segmentation, classification, and image analysis, notably in medical applications. Various approaches have been assessed across diverse medical datasets, spanning MRI scans of brain malignancies and tumors from other anatomical locations (Veeraraghavan, 2005). The medical sector has benefited greatly from the development of artificial intelligence and machine learning technologies, which have been crucial in supporting imaging among other fields. In the field of MRI image processing, machine learning approaches for segmentation and classification provide radiologists useful insights and alternative views. Launched by the Center for Biomedical Image Computing Analytics (CBICA) at the University of Pennsylvania's Perelman School of Medicine in 2012, the Multi-modal Brain Tumor Segmentation Challenge (BRATS) has grown to be a prominent online competition Akkus et al. (2017); cbi (2019). George and others. According to George et al. (2015), the multi-layer Perceptron (MLP) outperformed the C4.5 classifier using characteristics like the Euler Number and tumor axis lengths. The researchers trained their methods on 174 MRI scans of brain tumors, using C4.5 and MLP, and achieved classification rates of 91.1% and 95.2%, respectively. Devkota et al. Devkota et al. (2017) developed a new segmentation technique that combines spatial Fuzzy C-Means (FCM) with Mathematical Morphological Operations, resulting in improved computational efficiency and encouraging preliminary results. The technique successfully identified cancer with a 92% success rate, although more assessment is needed. Song et al. Song et al. (2016) used FLAIR and T1 modalities to categorize brain tumors into three categories: necrosis, edema, and normal tissue. They used a region-based active contour model to detect abnormal spots in FLAIR images. Then, using T1 modality contrast enhancement via the k-means approach, the differentiation between oedema and tumor tissues within these areas increased. This approach marks a significant development in brain tumor segmentation research, with a Dice coefficient of 73.6% and a sensitivity rate of 90.3%. Our technique attained an impressive accuracy rate of 91.16%, indicating extraordinary proficiency and a significant milestone in our research route. The work conducted by Sajjad, Muhammad et al. Sajjad et al. (2019) stands as a notable milestone in the realm of brain tumor classification, representing a significant advancement in the field. Their innovative approach integrates Convolutional Neural Networks (CNNs) with extensive data augmentation techniques. Central to their methodology is the intricate multi-grade categorization of brain tumors using segmented MRI images. Leveraging the widely recognized VGG-19 CNN architecture and employing transfer learning strategies, they achieved remarkable re-

sults. Notably, the application of data augmentation resulted in a noteworthy increase in accuracy, from an initial 87.38% to a remarkable 90.67%.In a similar spirit, Özyurt, Fatih, et al. Özyurt et al. (2019) attempted to tackle the difficulty of brain tumor classification using a unique integration of CNNs with the nuances of neutrosophic expert maximum fuzzy (NS-CNN) sure entropy. Their methodology begins with the precise segmentation of brain tumors using a neutrosophic set-expert maximum fuzzy-sure strategy, which lays the groundwork for their intricate and sophisticated process. As a result, CNN intervention makes it easier to extract features that Support Vector Machine (SVM) classifiers can accurately classify as benign or malignant. The end result of their efforts is nothing short of extraordinary, with an average success rate of 95.62

In the medical domain, a significant impediment to the widespread integration of deep learning methodologies lies in the constrained availability of labeled datasets. It is well-established that deep learning models exhibit enhanced performance when trained on expansive datasets. Notably, conventional pre-trained architectures such as VGG-16, ResNet-50, and Inception-v3 feature prominently in the literature. Furthermore, fine-tuning of these models proves indispensable for radiological investigations and clinical trials, necessitating adjustments such as freezing certain layers and modifying fully connected layers to align with dataset characteristics. However, it is pertinent to acknowledge a notable drawback of transfer learning, namely, the requisite adaptation of image input dimensions to conform to those of the pre-trained model.

To mitigate the challenges stemming from a limited dataset of brain MRI images, the research endeavors employed data augmentation and image processing techniques to expand the available data pool. Subsequently, a Convolutional Neural Network (CNN) model was trained utilizing the augmented and processed images to discern tumors within MRI scans. Owing to these constraints, the study was constrained to a modest number of brain MRI images. Notably, these images were subjected to enhancement via image processing and data augmentation methodologies prior to training a CNN model from scratch to identify tumor manifestations within the MRI scans.

Table 2.1: Summary of Techniques for Brain Tumor Detection and Classification Using CNN

Study Reference	**CNN Model(s) Used**	**Methodology**	**Key Results**
Basheera et al.	Pre-trained CNN (Specific model not stated)	Segmentation & Classification	Segmented tumor classification
Sajjad et al.	VGG-19	Data Augmentation & Transfer Learning	Classification of multi-grade tumors
Ricciardi et al.	-	Multinomial Logistic Regression, KNN	83% accuracy (Logistic Regression), 92% (KNN), AUC 98.4
Khwaldeh et al.	Modified Alex-Net	Classification & Grading System	91% accuracy
Abiniwanda et al.	CNN (Specific model not stated)	Classification of Tumor Types	98.51% training accuracy, 84.19% validation accuracy
Das et al.	CNN (Specific model not stated)	Classification of Tumor Types	94.39% accuracy, average accuracy 93.33%
Romeo et al.	-	Radiomic Machine Learning	Maximum accuracy 92.9%
Talo et al.	ResNet34	Transfer Learning & Data Augmentation	100% accuracy
Rehman et al.	VGG16, AlexNet, GoogleNet	Transfer Learning	Highest accuracy with VGG16: 98.67%
Çinar et al.	Modified ResNet50	Transfer Learning & Model Modification	97.2% accuracy
Mohsen et al.	DNN (Deep Neural Network)	Classification	96.97% accuracy
George et al.	Multi-layer Perceptron, C4.5	Classification Based on Tumor Shape Features	MLP: 95.2% accuracy, C4.5: 91.1% accuracy
Devkota et al.	-	Morphological Operations & Spatial FCM	Detection success rate of 92%
Yantao et al.	-	Histogram-based Segmentation & Classification	Dice coefficient: 73.6%, Sensitivity: 90.3%
Özyurt et al.	CNN with Neutrosophic Expert Maximum Fuzzy	Segmentation & Classification	Average success rate: 95.62%

3 Introduction to Brain Tumor, MRI and the Basic Algorithm

3.1 Brain Tumor

Within the field of medicine, an aberrant and uncontrolled multiplication of cells in the brain is referred to as a tumor. The human brain, the body's most sensitive organ, regulates muscle tone and processes sensory information such as visual, auditory, tactile, gustatory, and nociceptive impulses. The brain, which is made up of grey matter (GM), white matter (WM), and cerebrospinal fluid (CSF), is essential to human physiology. Tumorous growth identification is facilitated by diagnostic radiology through tissue quantification assessment, anomaly localization, and detection of malfunctions and diseases. These tumors have the ability to seriously harm sensory and motor abilities, even resulting in death. Tumors are categorized according to where they start; primary tumors develop inside the skull, whereas secondary cancers develop outside the brain and then return.

Brain tumors are classified as metastatic bronchogenic carcinoma, glioblastoma, or sarcoma depending on where on the axial plane they occur. While some tumors, like meningiomas, are easier to segment, other tumors, like gliomas and glioblastomas, are more difficult to locate. The World Health Organization (WHO) divides gliomas into two categories: high-grade gliomas (HGG), also referred to as glioblastomas (stage IV/malignant), and low-grade gliomas (LGG, stages II and III/benign). While the majority of LGG tumors grow more slowly and respond better to treatment, a small percentage can progress to GBM if they are not identified and treated right after. Determining the tumor grade accurately and promptly is essential for developing treatment plans that will enhance prognosis and include surgery, radiation, and chemotherapy. Patients with HGG or GBM have a remarkably short life expectancy—12 to 15 months.

Compared to CT, X-ray, and PET scans, magnetic resonance imaging (MRI) is now the most used non-invasive diagnostic method for brain tumors because of its superior soft-tissue contrast and lack of hazardous radiation. MRI pictures provide a thorough view of cerebral neoplasms since they are made up of pixel matrices, with each pixel having distinct properties.

3.2 Magnetic Resonance Imaging (MRI)

A major turning point in medical imaging was reached in 1969 when Raymond v. Damadian developed the first magnetic resonance image. By 1977, the human body had been included in the use of Magnetic Resonance Imaging (MRI) technology, which represented a methodological advance in the capture of finely detailed interior structures. An unmatched picture of the complex anatomy of the brain is made possible by MRI technology, which also makes it possible to distinguish between different types of tissue in the human body. This imaging method is acknowledged for providing better clarity and detail than other modalities, including computer tomography and X-rays R. A. Novellines (2004). When it comes to the identification and assessment of brain tumors, magnetic resonance imaging (MRI) is especially helpful.

Among the various MRI sequences, T1-weighted and T2-weighted images are most com-

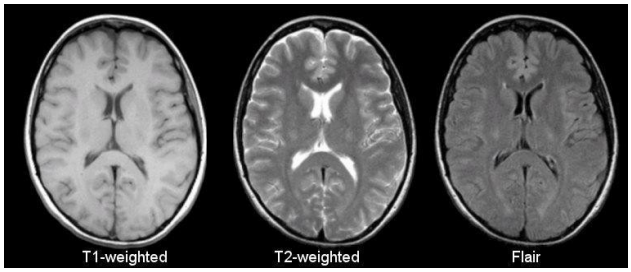

Figure 3.1: T1, T2, and FLAIR weighted MRI images for mapping tumor-induced change R. A. Novellines (2004).

monly employed. T1-weighted imaging is characterized by high signal intensity for fat, whereas T2-weighted imaging distinguishes both fat and water with high signal intensity. The differentiation between these sequences is largely attributable to variations in repetition time (TR) and echo time (TE), both measured in milliseconds (ms). Specifically, T1-weighted sequences utilize a shorter TR, while T2-weighted images employ longer TR and TE times, reflecting the pulse sequence parameters that dictate the timing of magnetic field applications and the measurement of resulting signals Preston (2006). Fluid-attenuated

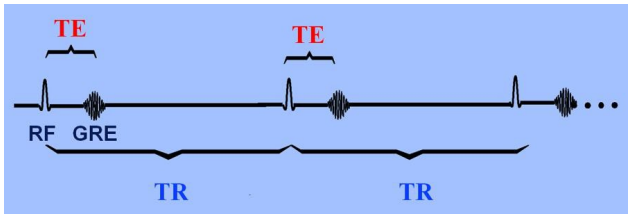

Figure 3.2: Illustration of echo time (TE) and repetition time (TR) in MRI sequences Hendrik (2005).

inversion recovery (FLAIR) imaging represents another pivotal MRI sequence, bearing similarity to T2-weighted imaging but with significantly prolonged TE and TR times. This sequence is particularly adept at suppressing the signal from free water, thereby enhancing the contrast and visibility of lesions, such as those induced by tumors.

Table 3.1: Comparison of TR and TE times for different MRI sequences Preston (2006).

Sequence	**TR (msec)**	**TE (msec)**
T1 Weighted (Short TR and TE)	400	15
T2 Weighted (Long TR and TE)	3000	80
FLAIR (Very Long TR and TE)	8000	115

These advancements in MRI technology not only underscore the versatility and precision of this imaging modality but also highlight its critical role in the ongoing efforts to improve diagnostic accuracy for brain tumors and other neurological conditions.

3.3 Introduction to Convolutional Neural Network

One approach used in deep learning is the convolutional neural network (CNN). It is capable of taking an input image, differentiating between items in the image by applying weights and biases to their respective levels of relevance. A CNN model requires a lot less pre-processing than other classification techniques. In the antiquated techniques, filters are designed by hand. CNN is capable of learning these filters and attributes given sufficient training. The connection pattern of neurons in the human brain is comparable to the architecture of a CNN. By applying filters to an image, a CNN can identify the temporal and spatial con-

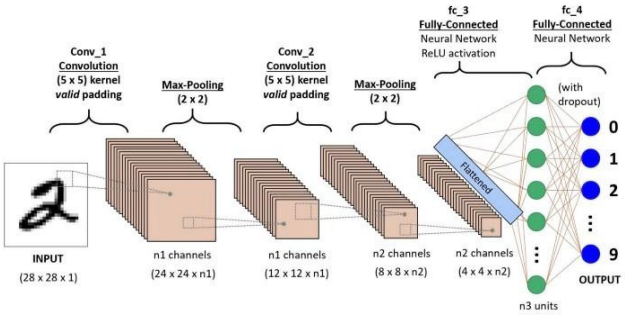

Figure 3.3: An Example of CNN architecture Hendrik (2005).

nections within the picture. The CNN architecture is more effective with image datasets due to its fewer parameters and easier-to-use weights. This allows the network to better understand the complexity of the image during training. The convolution process aims to

extract high-level features such as edges from the input image. Initially, the first CNN layer captures low-level features like edges, color, and gradient exposure. As the layers progress, the architecture adjusts to understand higher-level features, creating a network that comprehensively understands the dataset images much like the human brain. The convolution function results in either a decrease or increase in the dimensionality of the convolved points compared to the input. Decrease in dimensionality is achieved through Valid Padding, while an increase or unchanged dimensionality is achieved through Same Padding. The pooling

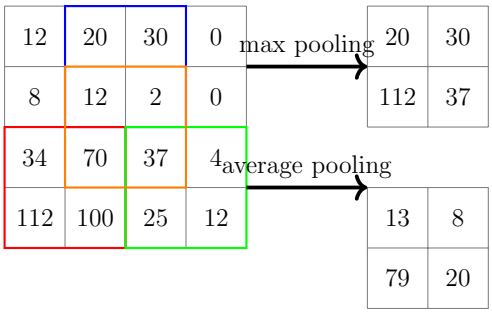

Figure 3.4: Max pooling and average pooling operations

layer is accountable for reducing the spatial size of the convolved feature in the same way as the convolutional layer does. The pooling layer reduces dimensionality, which in turn reduces the amount of processing power required to reprocess the data. It also helps extract dominating features that are rotationally and positionally invariant, so preserving the model's training. Average pooling and maximum pooling are the two types of pooling. First, max pooling is used to return the maximum value from the region of the image that the kernel covers. Second, average pooling is used to create the average of all the transactions from the kernel-covered image region. Furthermore, noise is suppressed via Max Pooling. In addition to eliminating the noisy activation, it also de-noises the dimensionality reduction.

Conversely, Average Pooling reduces dimensionality in a manner similar to that of a noise-suppression method.Consequently, we can say that Max Pooling performs better than Average Pooling. The Pooling Layer and the Convolutional Layer comprise the i-th layer of a CNN algorithm. Depending on the complexity of the photos, it may be possible to increase the number of comparable layers to extract even more low-level details, but doing so will need more processing resources. The final output is flattened and input into a normal neural network for classification. Adding a fully-connected layer is a popular technique for learning non-linear combinations of the high-position properties as represented by the affair of the convolutional layer. The Fully-Connected layer understands a possibly non-linear function in such space.

After the input image was processed and reshaped, it was converted into a single-column

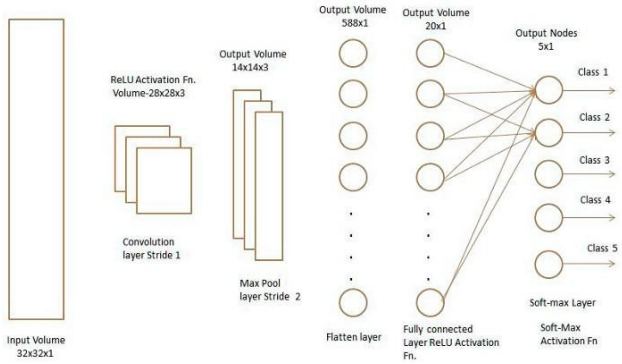

Figure 3.5: CNN fully linked layer categorization CNN categorization layer description Saha (2018).

vector. This vector was then fed into a neural network for forward-propagation, and each training iteration was carried out using back-propagation. After several training epochs, the model could be utilized to classify images using the softmax method and distinguish between brain tumors and specific features at a lower level.

3.4 Transfer Learning

Many machine learning and data mining methods rely on the idea that training and subsequent data should have the same features and patterns. However, this is not always applicable. For example, when categorizing data, we may find that later data is in a different feature space or has a different distribution. In such cases, if knowledge transfer is successful, learning performance can be significantly improved by reducing the need for costly data labeling. To address this challenge, a new approach called transfer learning has emerged. Neural networks using transfer learning require less data and leverage knowledge from past tasks to improve performance on current tasks. By transferring the network parameters learned from previous tasks, we can enhance model accuracy while reducing the need for additional data and training time.

Transfer learning can be classified into three categories: unsupervised, transductive, and inductive. The focus of most previous studies has been on different settings. Knowledge transfer techniques can be divided into four scenarios based on the type of knowledge being transferred: feature-representation-transfer, parameter transfer, relational knowledge transfer, and instance transfer. Pre-training involves training smaller networks, which are then used as a starting point for larger networks. Although this method is effective, it is time-consuming and slow. For example, feature extractors from a bareness detection model can be applied to accelerate learning in computer vision tasks.

Using a pre-trained model can significantly reduce the time needed for feature engineering

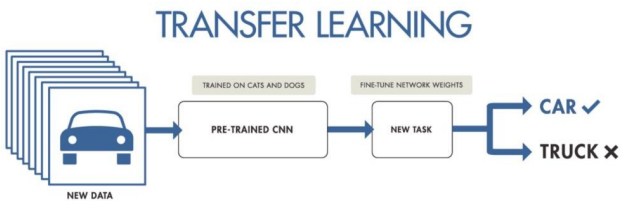

Figure 3.6: An example of Transfer Learning

and training.

Select Source Model: The initial step involves selecting a source model, preferably one that has a substantial dataset for training. Many research institutions make these models and datasets available as open-source projects, eliminating the need to create them from scratch.

Exercise the model: The next step is to determine which layers to utilize again in the network. The objective is to create a structure that is more effective than a simplistic model, ensuring that some new feature learning occurs.

Tune the model: In general, deeper layers are commonly repurposed because they are typically more generic, while the top layers are usually more specifically tailored to a particular issue.

In the end, the dataset is used to train the new model. One key benefit is that the model reaches convergence faster, requiring less data and compute time. Transfer learning is employed to save time or improve performance through optimization. When using transfer learning, there are three potential advantages to consider.

1. Enhanced initial proficiency: The source model demonstrates a higher level of inherent skill compared to its typical state.

2. Accelerated skill acquisition: The rate of skill enhancement during the training phase of the source model surpasses the standard pace.

3. Elevated skill plateau: The final expertise achieved by the trained model exceeds the usual level it would attain.

3.4.1 The VGG-16 Model

Convolutional Neural Networks (CNNs) represent a pivotal advancement in deep learning, primarily due to their capacity to autonomously extract and classify features from input images. Unlike other classification algorithms, CNNs require significantly less preprocessing, thus streamlining the model training process. One notable implementation of this architecture is the VGG-16 model, which is characterized by its depth of 16 convolutional layers.

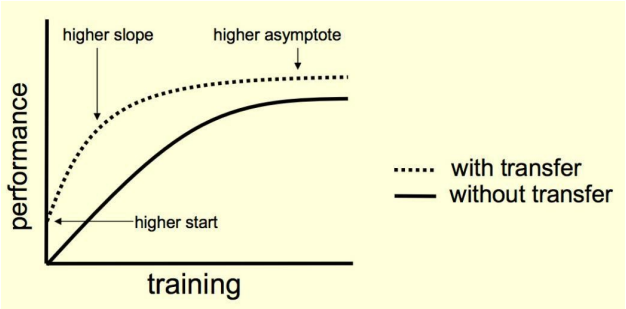

Figure 3.7: Transfer Learning for Deep Learning With CNN

This depth contributes to the model's robustness but also implies longer training durations due to the comprehensive processing of input images across these multiple layers. Specifically, the VGG-16 model is adept at categorizing images into distinct classes, ranging from the detection of specific objects to identifying the presence or absence of tumors in medical images. Developed by K Simonyan and A Zisserman, the VGG-16 model has demonstrated remarkable performance, achieving a top-5 test accuracy of 92.7% on the ImageNet database Simonyan and Zisserman (2014a). Within the context of brain tumor detection, the VGG-16 model is employed to classify MRI scans based on tumor presence, showcasing its versatility and effectiveness in medical image analysis.

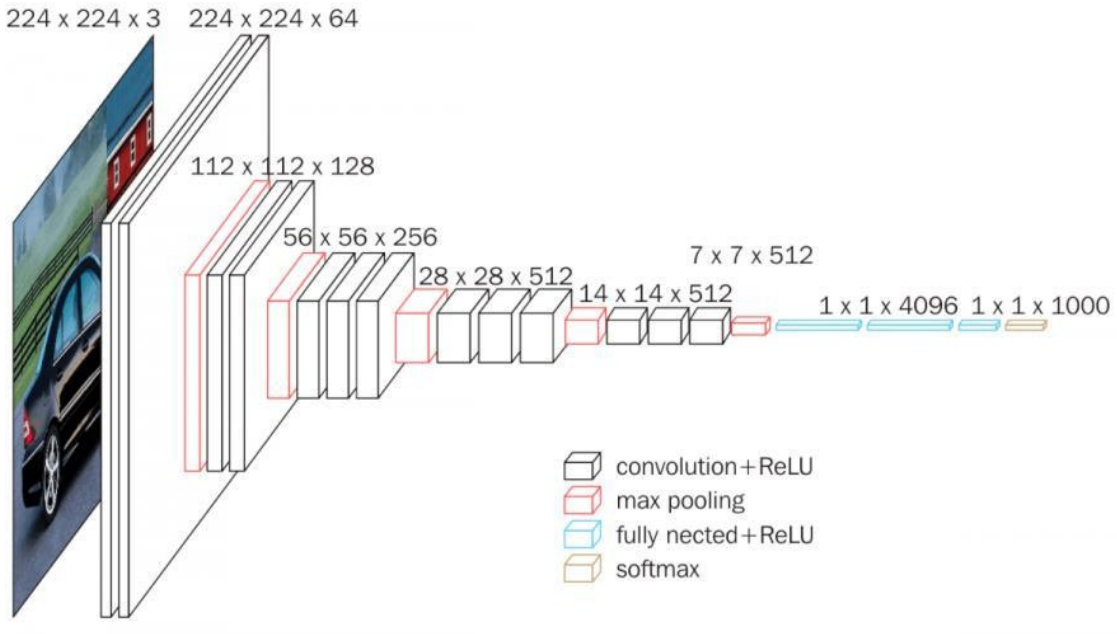

Figure 3.8: VGG-16 Model

The VGG-16 architecture is intricately designed with multiple layers, each contributing to the model's ability to process and classify input images through a series of steps:

1. **Input Image:** The process begins with the model accepting an input image of dimensions 224 × 224 pixels.

2. **Convolution:** Each layer incorporates a 3 × 3 convolutional kernel that convolves across the input image with a stride of 2 pixels, resulting in the generation of a feature map. This feature map represents the convolution of the kernel with the input image and serves as input to the subsequent layer.

3. **Max Pooling:** To enhance feature detection while reducing the feature map's dimensionality, max pooling is applied using a 3×3 filter with a stride of 2 pixels. This step selectively retains the most significant features.

4. **Normalization:** The activation function used is the Rectified Linear Unit (ReLU), which transforms negative pixel values to zero, effectively normalizing the feature map and eliminating non-essential pixels.

5. **Dropout:** To prevent overfitting and encourage the model to learn alternative pathways for feature representation, dropout layers randomly nullify the output of selected neurons.

6. **Softmax Function:** In the final layer, the softmax function converts the neural network's output into a probability distribution, facilitating the classification of the input image based on the probability scores derived from the model's learned features.

The VGG-16 model exemplifies the sophisticated architecture and processes inherent in CNNs, offering a powerful tool for image classification tasks, including the nuanced field of medical imaging for brain tumor detection.

Algorithm 1 VGG-16 Model for Brain Tumor Detection and Classification

1: **Input**: Image dataset with shape $(224, 224, 3)$
2: **Output**: Classification into 4 categories
3: Load VGG16 weights from *vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5*
4: Initialize VGG16 model with weights, excluding top, and input shape $(224, 224, 3)$
5: Initialize Sequential model
6: Add VGG16 base model to Sequential model
7: Add Flatten layer
8: Add Dropout layer with rate 0.5
9: Add Dense layer with 4 units and softmax activation
10: Set VGG16 base model layers to non-trainable
11: Compile model with categorical crossentropy loss, RMSprop optimizer with learning rate $1e-4$, and accuracy metric
12: Display model summary
13: Fit model on training data with steps per epoch $= \frac{5712}{32}$, epochs $= 30$, validation data, and validation steps $= \frac{1311}{32}$

3.4.2 The Xception Model

Francois Chollet introduced the Xception architecture, a novel extension to the inception model designed to enhance image detection capabilities Chollet (2017). The Xception model innovates by replacing standard inception modules with depth-wise separable convolutions, offering a more efficient mechanism for feature extraction and processing. Trained on the expansive ImageNet database, which comprises over one million images, the Xception model excels in capturing the generalized features across a diverse array of images. For the specific task of brain tumor detection, the Xception architecture has been meticulously fine-tuned on a dedicated dataset, enabling the precise classification of brain tumors into four distinct categories. This fine-tuning process leverages the model's ability to extract relevant features from the fully connected layers, subsequently employing a softmax classifier to achieve accurate categorization.

Figure 3.9: Xception Model

The application of the Xception model to the domain of brain tumor detection and classification is delineated through the following algorithmic procedure:

Algorithm 2 Xception Model for Brain Tumor Detection and Classification

1: **Input**: Image dataset with shape $(224, 224, 3)$
2: **Output**: Classification into four categories
3: Load Xception weights from *xception_weights_tf_dim_ordering_tf_kernels_notop.h5*
4: Initialize Xception model with preloaded weights, excluding the top layer, configured for an input shape of $(224, 224, 3)$
5: Construct a Sequential model
6: Integrate the Xception base model into the Sequential framework
7: Append a Flatten layer
8: Incorporate a Dropout layer with a rate of 0.5
9: Add a Dense layer, equipped with four units and softmax activation function
10: Designate the layers of the Xception base model as non-trainable
11: Compile the model, specifying categorical crossentropy as the loss function, RMSprop optimizer with a learning rate of $1e-4$, and accuracy as the performance metric
12: Provide a summary of the model configuration
13: Train the model on the specified dataset, setting steps per epoch to $= \frac{5712}{32}$, total epochs to 30, and incorporating validation data with validation steps $= \frac{1311}{32}$

3.4.3 The ResNet50 Model

The ResNet50 model, a variant of the Residual Network architecture, represents a significant advancement in the field of deep learning, introduced by Microsoft in 2015 He et al. (2016). Comprising 50 layers and approximately 26 million parameters, ResNet50 has been designed to address the challenges associated with training very deep neural networks. The fundamental innovation of the Residual Network is its use of residual learning, where the network focuses on learning the residuals or differences between the input and output features, rather than the features themselves. This is achieved through the implementation of skip connections, which directly link the input of an n^{th} layer to the $(n+x)^{th}$ layer. Such connections facilitate the flow of information across layers, allowing for the construction of deeper networks without succumbing to the vanishing gradient problem. The application of

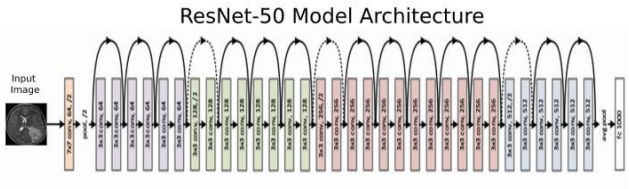

Figure 3.10: ResNet50 Model

the ResNet50 model in the domain of brain tumor detection and classification is structured as follows:

Algorithm 3 ResNet50 Model for Brain Tumor Detection and Classification

1: **Input**: Image dataset with shape $(224, 224, 3)$
2: **Output**: Classification into four categories
3: Initialize ResNet50 model with specified input shape $(224, 224, 3)$, excluding the top layer to adapt the model for custom classification tasks.
4: Construct a Sequential model framework.
5: Integrate the ResNet50 base model within this Sequential framework.
6: Append a Flatten layer to convert the multidimensional output tensors into a flat vector.
7: Incorporate a Batch Normalization layer to standardize the inputs to the subsequent layer, enhancing stability and performance.
8: Add a Dense layer with 256 units and initialize weights using the he_uniform method, optimal for layers with ReLU activation.
9: Insert another Batch Normalization layer to further stabilize learning.
10: Employ a Dropout layer with a rate of 0.5 to prevent overfitting by randomly omitting a proportion of layer outputs.
11: Affix a Dense output layer with four units and a softmax activation function for multi-class classification.
12: Designate all layers of the ResNet50 base model as non-trainable to preserve pre-trained features.
13: Compile the model specifying categorical crossentropy as the loss function, utilizing the Adam optimizer for efficient stochastic optimization, and measuring performance through accuracy.
14: Exhibit a summary of the model to review its architecture and parameters.
15: Train the model on the provided dataset, setting the number of steps per epoch to $= \frac{5712}{32}$, total epochs to 30, and specifying validation data along with validation steps $= \frac{1311}{32}$.

3.4.4 The Inception-V3 Model

Originally introduced by Google in 2014, the Inception network, also known as GoogleNet, marked a significant advancement in the field of deep learning Szegedy et al. (2015). This network comprises 22 layers and is characterized by its relatively modest parameter count of 5 million. The Inception model employs convolutional filters of varying sizes (1×1, 3×3, and 5×5) to efficiently extract features at multiple scales, incorporating max pooling to further refine the feature extraction process. The introduction of 1×1 convolutional filters serves to reduce computational time without compromising the model's efficacy. Building upon this foundation, Google introduced an enhanced version, InceptionV3, in 2015 Szegedy et al. (2016), which incorporates factorized convolutional layers. This optimization involves substituting 5×5 convolutional filters with two sequential 3×3 filters, thereby reducing

the number of parameters and computational load while maintaining performance integrity. The InceptionV3 architecture, which expands to 48 layers, has been fine-tuned for specific applications, including brain tumor detection, to mitigate the risk of overfitting.

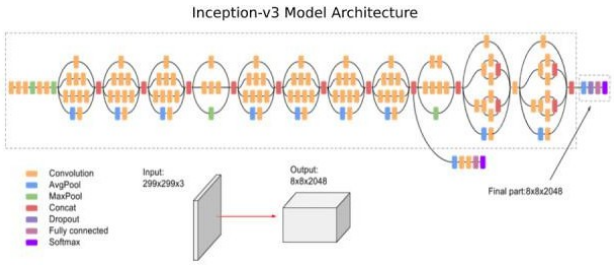

Figure 3.11: Inception-V3 Model

The application of the InceptionV3 model in the detection and classification of brain tumors is outlined in the following algorithmic procedure:

Algorithm 4 InceptionV3 Model for Brain Tumor Detection and Classification

1: **Input**: Image dataset with shape $(224, 224, 3)$
2: **Output**: Classification into four categories
3: Load InceptionV3 weights from *inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5*
4: Initialize the InceptionV3 model with these weights, omitting the top layer, and setting the input shape to $(224, 224, 3)$
5: Construct a Sequential model
6: Incorporate the InceptionV3 base model into the Sequential framework
7: Append a Flatten layer
8: Integrate a Dropout layer with a rate of 0.5
9: Add a Dense layer comprising four units with softmax activation for classification
10: Designate the layers of the InceptionV3 base model as non-trainable to preserve the pre-trained features
11: Build the model using the RMSprop optimizer with a learning rate of $1e-4$, categorical crossentropy as the loss function, and accuracy as the evaluation metric.
12: Generate a summary of the model architecture
13: Train the model on the designated dataset, with steps per epoch set to $= \frac{5712}{32}$, a total of 30 epochs, validation data included, and validation steps set to $= \frac{1311}{32}$

4 Methodology

Brain tumors detection and classification methods are crucial in ensuring higher accuracy in safety-critical healthcare processes. The detection process involves collecting MRI scans and pre-processing the data.

Initially, grayscale images are converted into RGB with three channels. The input shape is then defined as (224, 224, 3). The Data Augmentation method is used to expand the size of the training data because the dataset is quite small. More detailed information on Data Pre-processing can be found in other sections.

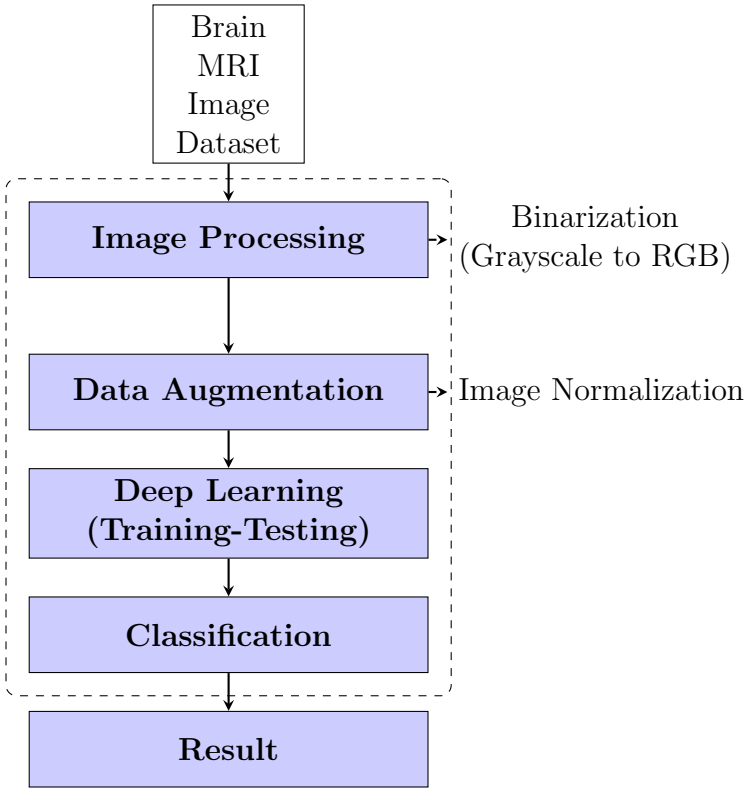

Figure 4.1: Flowchart of the image processing pipeline.

We began by building a CNN model from scratch and utilizing the dataset to train it. The model was run through 30 epochs, and accuracy was noted as we tested it using the test dataset. Our experiment was conducted on Google Colab, using TensorFlow and Keras libraries in Python, which took close to two hours to train all the models. We used the same pre-processed dataset to train pre-existing models like ResNet50, Xception, VGG-16, and Inception-V3. These models accurately classified the dataset into four classes: meningioma, glioma, pituitary, and no tumor.

4.1 CNN Model Building

4.1.1 Layers, Activation Function and Optimizers

Layers in the proposed CNN model:

The Convolution (Conv2D) Layer: We have utilized the Keras Sequential API, which necessitates the addition of layers beginning with the input and progressing downwards. The initial layer is the convolutional (Conv2D) layer. This layer creates a tensor of outputs, which are made up of different trainable filters, by wrapping a convolution kernel around the input layers. For the first two Conv2D layers, we have opted to employ 64 filters each. These filters utilize the kernel size to modify specific regions of the image, as the entire image is passed through the kernel filter matrix. The application of filters is often seen as a way to manipulate the image.

Maxpooling2D: The MaxPool2D layer is an essential component of the CNN, serving as a downsampling filter by selecting the maximum pixel value. The use of pooling helps reduce computational costs and mitigate overfitting, with common practices including pixel selection. Determining the pooling size is crucial as it impacts downsampling and the ability to connect local features with global image aspects. By combining convolutional and pooling layers, CNN is able to effectively link and explore both local and global image features.

Dense: Every neuron in the dense layer is connected to every other neuron in the layer above, indicating that the layer is fully connected. An artificial neural network (ANN) using features from a single fully connected (Dense) layer is the sole classifier that has been deployed thus far. Because every neuron in a layer receives input from every other neuron in the layer before it, neurons in a layer are highly connected.

Dropout:A regularization approach called dropout involves randomly deactivating a portion of a layer's nodes by setting their weights to zero during training. The network is compelled to learn features in a more distributed way by deleting nodes at random, which improves generalization and avoids overfitting.

Flatten: The final feature of the maps has been transformed into a single 1D vector using this layer. Using a fully connected layer after some convolutional/max pool layers requires the flattening step. Combining all of the local features discovered by the earlier convolutional layers is the primary task. It has no bearing on the batch size.

Batch Normalization: Internal co-variate shift in neural networks is efficiently resolved by batch normalization, which standardizes inputs for every mini-batch during training. This facilitates the use of greater learning rates and a more seamless input transition across layers. Batch normalization helps to formalize the training process and drastically lowers the number of epochs required to train deep networks by continuously removing dropouts. This

helps to stabilize the learning process and makes it possible to train neural networks that are really deep.

Activation Functions:

- **ReLU:** The Rectified Linear Unit (ReLU) activation function has been applied to each convolutional layer. The output of that node, which is acquired by turning on the weighted input sum, is indicated by Vinod and Hinton.Hinton (2012) In the hidden layers of a convolutional neural network, the rectifier linear unit function is frequently employed. The symbol for ReLU in mathematics is. When z is the input and z is negative or equal to zero, it turns the negative input to zero. If the input is more than 0, the outcome will be 1. ReLUs' derivative will therefore be

$$f(z) = \max(0, z) ReLU Activation Function \tag{1}$$

$$f'(x) = \begin{cases} 1, & \text{for } x \geq 0 \\ 0, & \text{for } x < 0 \end{cases} \tag{2}$$

Therefore, in ReLU function, the neuron is dead and will not be activated if the input is 0.

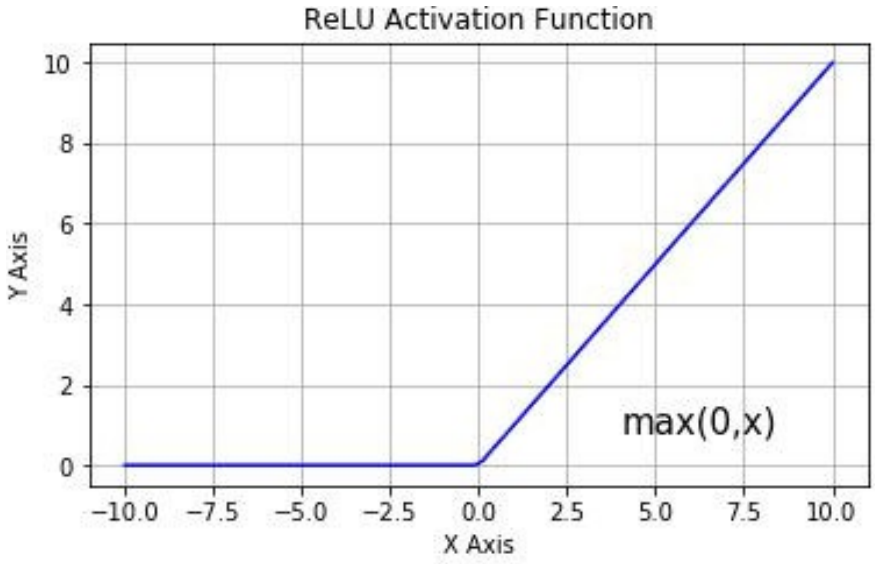

Figure 4.2: ReLU Activation Function

- **Softmax:** : The activation function in neural network models' output layer that forecasts a multinomial probability distribution is called the softmax function. When

solving multi-class classification issues requiring class membership on more than two class labels, Softmax is frequently utilized.

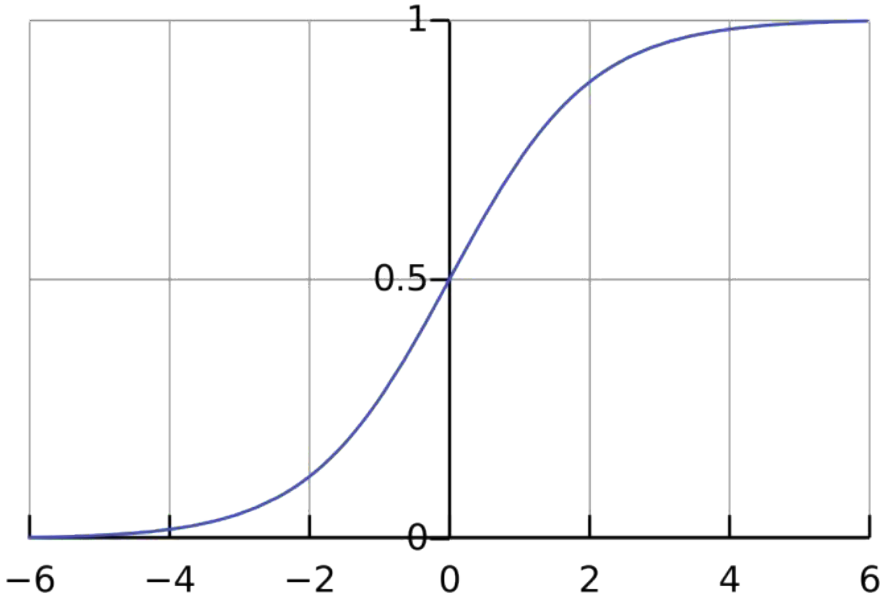

Figure 4.3: Softmax Activation Function

Loss Function: The loss function is a deep learning concept that measures the difference between the actual values and the predicted labels produced by the algorithm. The default loss function for multi-class classification tasks gives each class a unique integer value between 0 and (num_classes – 1). Across all classes, it calculates the average difference between the observed and projected probability distributions. A prediction with a lower score—ideally with a cross-entropy value of zero—is more accurate. In this work, the total loss is evaluated using the Categorical Cross-Entropy.

$$\text{Loss} = -\sum_{i=1}^{\text{output}} y_i \cdot \log \hat{y}_i \tag{3}$$

Here, $\hat{y}_i$ depicts the i-th scalar value in the output, y_i represents the target value, and the output size presents to the number of scalar values in the output. This loss function effectively measures the dissimilarity between two discrete probability distributions. y_i represents the probability of event i occurring, with the sum of all y_i equaling 1, indicating that only one event can occur. As the distributions converge, the negative sign ensures a reduction in loss. One-hot encoding of the target values is required in order to correctly apply the categorical cross-entropy loss function. The output layer consists of n nodes, one for each class (i.e., 10 nodes in this case), and a "softmax" activation function is used to estimate

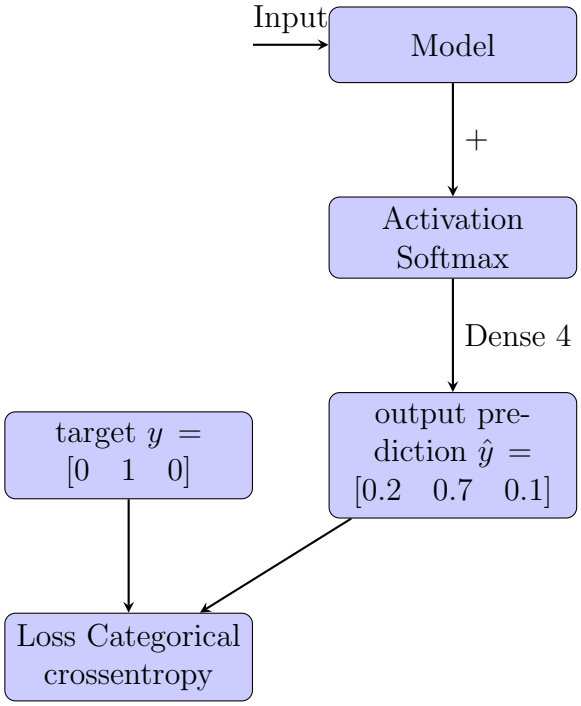

Figure 4.4: Categorical Cross-Entropy

the probability for each class.

Optimizer:

- **Stochastic Gradient Descent:** The Stochastic Gradient Descent method was first presented in 1951 by Herbert and Sutton Robbins and Munro (1951). This method involves multiplying the learning rate by the derivatives of weights (dW) and biases (DB) for each epoch.

$$W = W - \eta \times d \tag{4}$$

$$b = b - \eta \times db \tag{5}$$

The moving mean of our gradients is the stochastic gradient descent with momentum V, whereas the moving mean between 0 and 1 is the result of calculating dW and db on the current batch.

$$V_{dw} = \beta \times V_{dw} + (1 - \beta) \times d \tag{6}$$

$$V_{db} = \beta \times V_{db} + (1 - \beta) \times db \tag{7}$$

Since this is a multi-class classification challenge, categorical cross-entropy has been employed to compile the proposed CNN model. Formally speaking, its purpose is to ascertain how two probability distributions differ from one another Peltarion Platform (2019). When averaged over numerous mini-batches or samples, it can somewhat preserve the gradient direction while lowering the gradient's computation cost.The suggested CNN model has been assembled using the SGD optimizer.

- **RMSprop:** Changes in perpendicular direction are limited by the RMSprop optimizer. Therefore, accelerating the learning rate and the suggested CNN method could result in quicker and more significant vertical direction clustering stages. The method used to calculate the gradients distinguishes gradient descent from RMSprop.Similar to SGD, Root Mean Squared Prop is an adaptive learning rate algorithm that was introduced by Geoff Hinton Hinton (2012). The exponential moving mean square of gradients is employed in RMSprop. Thus, the hyperparameter Beta β in RMSprop regulates exponentially weighted means. The Inception-V3 Model has been assembled using the RMSprop optimizer.

$$S_{dw} = \beta \times S_{dw} + (1 - \beta) \times dW^2 \tag{8}$$

$$S_{db} = \beta \times S_{db} + (1 - \beta) \times d \tag{9}$$

- **Adam:** : Adam is typically the most efficient adaptive optimizer. Effective with sparse data The adaptive learning rate is best suited for this type of dataset.Through the use of transfer learning models, the Adam optimizer performs noticeably better than the one that was previously in use. It provides an ideal gradient descent, outperforming others by a wide margin. As a result, we combine the features of the weighted mean of the squares of past gradients and the weighted mean of prior gradients to apply the Adam optimization technique. Consequently, the updated weights and bias of the Adam optimizer will be

$$W = W - \eta \times \left(\frac{V_{dw}}{\sqrt{S_{dw}} + \varepsilon} \right) \tag{10}$$

$$b = b - \eta \times \left(\frac{V_{db}}{\sqrt{S_{db}} + \varepsilon} \right) \tag{11}$$

While is a learning rate with a different range of values, Epsilon is a tiny number (Epsilon = 0.00000001) that prohibits zero division. An optimizer trains a model

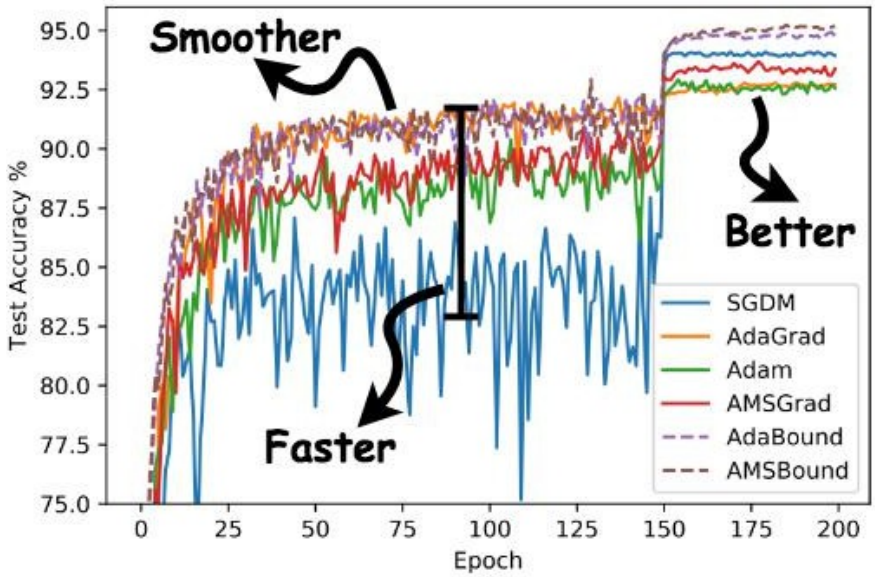

Figure 4.5: Different Optimizers in CNN

faster and good for developing state-of-the-art deep learning models on a wide variety of popular tasks in the field of CV, NLP, Etc. Luo et al. (2019). Adam optimizer has been used to train the Resnet50 and Inception-V3 model and RMSprop for Xception and VGG-16.

4.1.2 Structure of proposed CNN model

Six essential processes must be followed in order to create a CNN: convolution, fully connected layers, pooling, flattening, ReLU, and Softmax function. The input image is filtered in the convolution stage in order to identify image features. ReLU preserves positive values while converting negative values to zero in order to expedite training. Through down sampling, pooling reduces the number of parameters and simplifies the result. These stages are iterated across several layers in order to discern distinct characteristics. Arrays are flattened to provide a linear vector that fully connected layers can use. The features found in convolutional layers are included in fully connected layers. The final classification result is then produced using the softmax function.

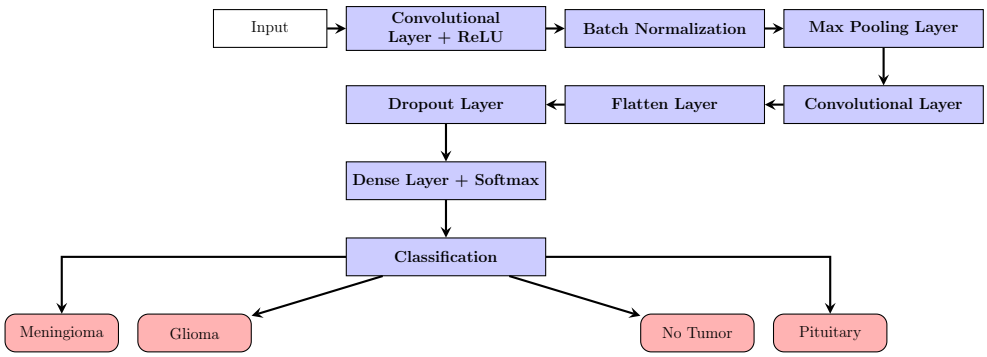

Figure 4.6: Neural network architecture for brain tumor classification.

4.1.3 Proposed CNN model architecture

In this study, a CNN model for analyzing MRI images is presented. The model is fed 224x224 enhanced data with 32 batch sizes and RGB color channels. A 64-filter convolutional layer with a 7×7 kernel makes up the first layer, which is intended to recognize lines, corners, and edges in brain MRI pictures. After that, a condensed perspective of the image is added by adding a max-pooling layer with a 2×2 kernel. To recognize larger patterns, the number of convolutional layers and filters is then increased to 128, 256, and 512, all using a 7×7 kernel. To enhance the results, max-pooling layers are added to each convolutional layer. In the end, the probability score for each class is determined by using a fully connected dense layer with 1024 neurons and a softmax output layer, which helps to classify meningioma glioma, pituitary, or no tumor. Figure 4.15 shows the suggested CNN model's design.

Table 4.1: Proposed CNN Model Architecture

Layer No	Layer Name	Layer Properties
1	Input Image	224×224×3 images
2	Convolutional	64 [7×7] convolutions with stride [1 1] and padding 'same'
3	ReLU	Rectified Linear Unit
4	Batch Normalization	Normalization to reduce training time
5	Max Pooling	[2×2] max pooling with stride [1 1] and padding [0 0 0 0]
6	Convolutional	128 [7×7] convolutions with stride [1 1] and padding 'same'
7	ReLU	Rectified Linear Unit
8	Batch Normalization	Normalization to reduce training time
9	Max Pooling	[2×2] max pooling with stride [1 1] and padding [0 0 0 0]
10	Convolutional	128 [7×7] convolutions with stride [1 1] and padding 'same'
11	ReLU	Rectified Linear Unit
12	Batch Normalization	Normalization to reduce training time
13	Max Pooling	[2×2] max pooling with stride [1 1] and padding [0 0 0 0]
14	Convolutional	256 [7×7] convolutions with stride [1 1] and padding 'same'
15	ReLU	Rectified Linear Unit
16	Batch Normalization	Normalization to reduce training time
17	Max Pooling	[2×2] max pooling with stride [1 1] and padding [0 0 0 0]
18	Convolutional	256 [7×7] convolutions with stride [1 1] and padding 'same'
19	ReLU	Rectified Linear Unit
20	Batch Normalization	Normalization to reduce training time
21	Max Pooling	[2×2] max pooling with stride [1 1] and padding [0 0 0 0]
22	Convolutional	512 [7×7] convolutions with stride [1 1] and padding 'same'
23	ReLU	Rectified Linear Unit
24	Batch Normalization	Normalization to reduce training time
25	Max Pooling	[2×2] max pooling with stride [1 1] and padding [0 0 0 0]
27	Fully Connected Layer	3 hidden neurons in the fully connected layer
28	ReLU	Rectified Linear Unit
29	Softmax	Softmax activation function
30	Classification output	Output classes: Glioma, Meningioma, No tumor, Pituitary

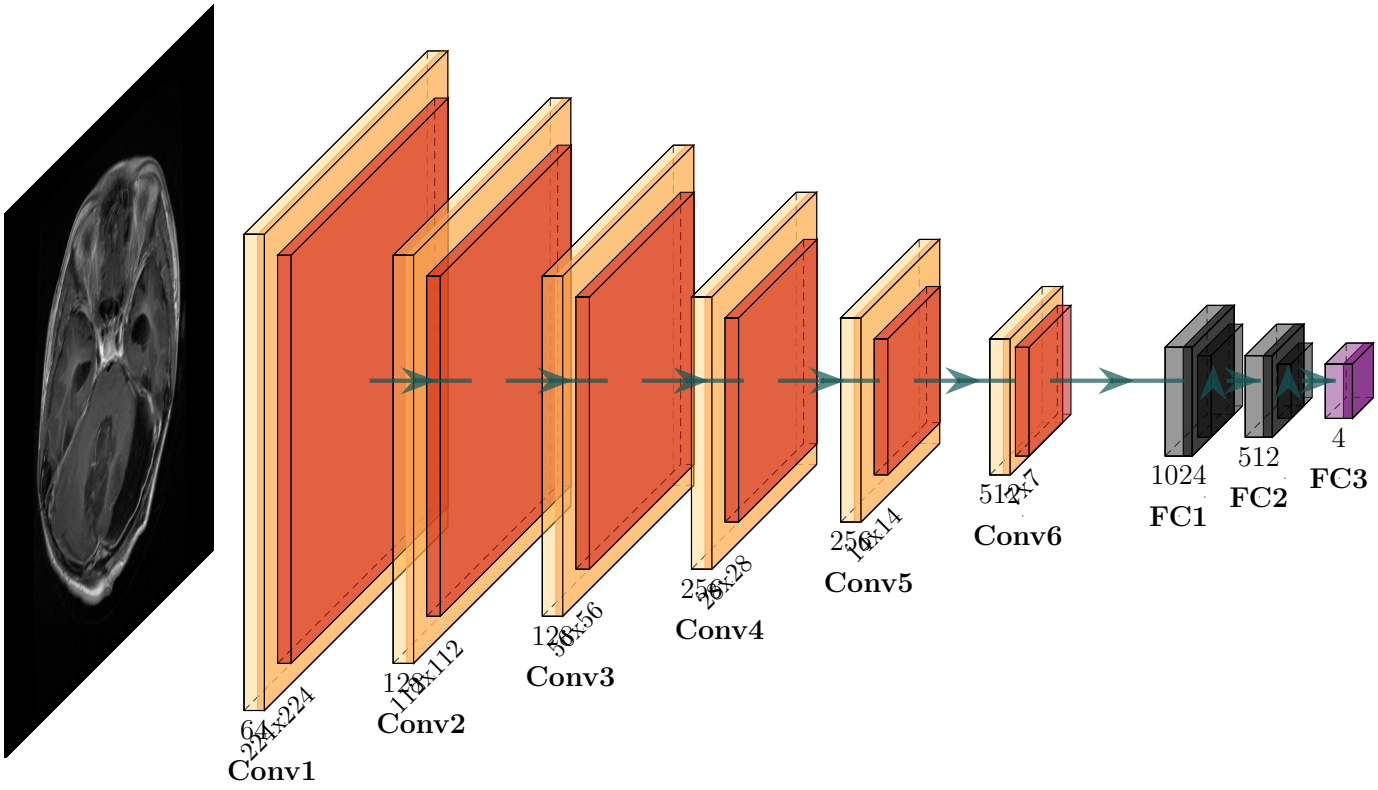

Figure 4.15: Proposed CNN Model Architecture

4.1.4 Proposed CNN model training

It will iterate (epochs) over several iterations after fitting the suggested CNN model for iteration. Epoches are nothing more than the optimization iterations used in model training. As a result, by doing further optimization rounds, or epochs, the error in the training data will be much decreased. But if the training dataset is over fitted, the model may begin to perform worse. Over-fitting may be indicated if the validation error rises. It is advised to set the maximum number of epochs and stop the training by examining the error rates. Typically, an epoch is one whole processing run through the whole training dataset. It is the final version of the dataset that was sent. This takes several stages to complete.

For the model, the number of epochs has been standardized to 30, as too many iterations will cause the learning rate to deteriorate. There has been a slow decline in the model's validation accuracy between the 17th and 22nd epochs, which may be a more encouraging indicator. Moreover, the over-fitting of the data causes the model to become less effective. As a result, the model becomes less accurate as its learning rate drops. As a result, it is crucial to constantly check the model loss and the accuracy of the findings generated at the end of each epoch. When the model completes each epoch, we get the metrics such as the training

accuracy, training loss, validation accuracy, and validation loss. Plotted matrices are the outcomes for each of these indicators as well as the analysis derived from the performance graphs. Overall, it took the suggested model close to an hour in Google Colab to finish all of the epochs and eventually become trained. It indicates that the model is prepared to be tested using the test data to determine its efficacy. Reviewing the model accuracy and prediction scores can help us to get there. Moreover, the weight will be automatically saved if the accuracy of the category validation increases with each iteration. The model was fitted with 5712//32 = 178 photos in each epoch, and the validation steps were completed successfully with 1311//32 41.

```
Epoch 00020: val_categorical_accuracy improved from 0.98672 to 0.98750, saving model to CNN_model_weights.h5
Epoch 21/30
178/178 [==============================] - 49s 276ms/step - loss: 0.0152 - categorical_accuracy: 0.9965 - val_loss: 0.0579 - val_categorical_accuracy:
0.9867

Epoch 00021: val_categorical_accuracy did not improve from 0.98750
Epoch 22/30
178/178 [==============================] - 49s 274ms/step - loss: 0.0115 - categorical_accuracy: 0.9982 - val_loss: 0.0723 - val_categorical_accuracy:
0.9812

Epoch 00022: val_categorical_accuracy did not improve from 0.98750
Epoch 23/30
178/178 [==============================] - 49s 274ms/step - loss: 0.0118 - categorical_accuracy: 0.9981 - val_loss: 0.0681 - val_categorical_accuracy:
0.9852

Epoch 00023: val_categorical_accuracy did not improve from 0.98750
Epoch 24/30
178/178 [==============================] - 49s 274ms/step - loss: 0.0121 - categorical_accuracy: 0.9975 - val_loss: 0.1019 - val_categorical_accuracy:
0.9773

Epoch 00024: val_categorical_accuracy did not improve from 0.98750
Epoch 25/30
178/178 [==============================] - 49s 276ms/step - loss: 0.0133 - categorical_accuracy: 0.9972 - val_loss: 0.0586 - val_categorical_accuracy:
0.9883

Epoch 00025: val_categorical_accuracy improved from 0.98750 to 0.98828, saving model to CNN_model_weights.h5
Epoch 26/30
178/178 [==============================] - 49s 275ms/step - loss: 0.0083 - categorical_accuracy: 0.9981 - val_loss: 0.0649 - val_categorical_accuracy:
0.9859

Epoch 00026: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.

Epoch 00026: val_categorical_accuracy did not improve from 0.98828
Epoch 27/30
178/178 [==============================] - 49s 275ms/step - loss: 0.0081 - categorical_accuracy: 0.9981 - val_loss: 0.0687 - val_categorical_accuracy:
0.9844

Epoch 00027: val_categorical_accuracy did not improve from 0.98828
Epoch 28/30
178/178 [==============================] - 49s 275ms/step - loss: 0.0054 - categorical_accuracy: 0.9995 - val_loss: 0.0683 - val_categorical_accuracy:
0.9844

Epoch 00028: val_categorical_accuracy did not improve from 0.98828
Epoch 29/30
178/178 [==============================] - 49s 274ms/step - loss: 0.0061 - categorical_accuracy: 0.9986 - val_loss: 0.0722 - val_categorical_accuracy:
0.9844

Epoch 00029: val_categorical_accuracy did not improve from 0.98828
Epoch 30/30
178/178 [==============================] - 49s 274ms/step - loss: 0.0059 - categorical_accuracy: 0.9991 - val_loss: 0.0815 - val_categorical_accuracy:
0.9797

Epoch 00030: val_categorical_accuracy did not improve from 0.98828
```

Figure 4.16: Screenshot showing the results of epochs in Proposed CNN model (20-30 epochs)

Figure 4.16 shows detail for each epoch, including test accuracy and validation accuracy. Furthermore, the approximate duration of every step in every era is given. At the conclusion of each epoch, the validation accuracy increases as the model repeats the procedure with a progressively higher learning rate. The model has been built and is able to categorize photos in the test dataset after 30 epochs. Notably, the validation categorical accuracy of the proposed CNN model shows improvement at the 23rd epoch, and it saves its weight automatically.

```
model.evaluate(test)
```

```
41/41 [==============================] - 7s 163ms/step - loss: 0.0789 - categorical_accuracy: 0.9809
[0.07886818051338196, 0.9809305667877197]
```

Figure 4.17: Screenshot of Evaluating the Proposed CNN Model

With a validation accuracy of 98% and a training accuracy of 99.93%, brain cancer detection

and classification have reached an astonishing degree of accuracy. To find the best alternative for detecting and diagnosing brain cancers, we will investigate the suggested CNN model in this chapter together with other pre-trained models.

4.2 Approach to Transfer Learning

Deep learning occasionally makes use of transfer learning. Rather than creating a new CNN model for image classification, an existing one that has been trained on a large benchmark dataset, like "ImageNet," is repurposed.

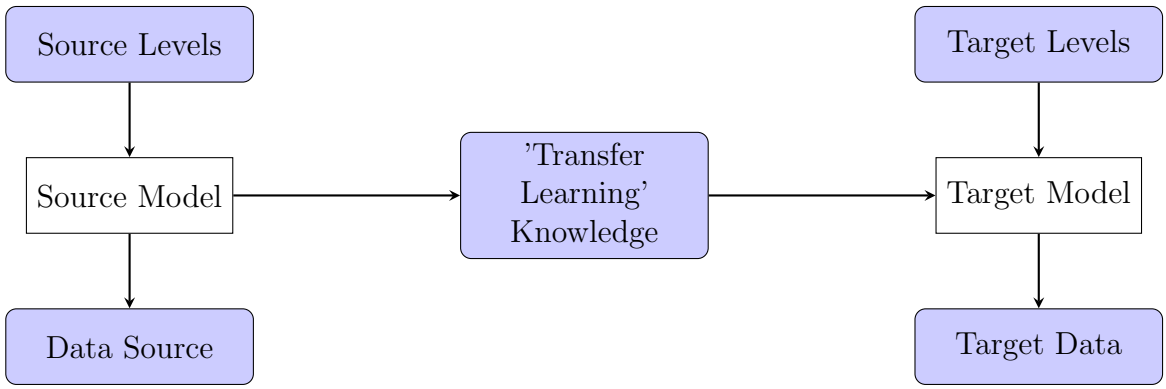

Figure 4.18: Transfer learning process

Transfer learning makes use of prior knowledge rather than starting the learning process from scratch. In this work, four pre-trained models for identifying brain tumors and categorizing them into four groups were employed.

4.2.1 VGG-16 Model training and building

First, we uploaded the VGG-16 Convolutional Neural Network Model package that came pre-installed. This model is widely used for deep learning-based image classification. We downloaded the pre-trained VGG-16 package from the internet and included it into our algorithm by giving the file location. To include more pre-trained models in our training, such as Xception, ResNet50, and Inception-V3, we used a similar process. In order to prevent overfitting caused by a small dataset, we modified a pre-trained VGG-16 Convolutional neural network model in our experiment by freezing several layers. VGG-16, first presented by Karen Simonyan and Andrew Zisserman Simonyan and Zisserman (2014b), has an input form of 224×224×3 and is composed of 16 convolutional layers. It uses the softmax activation function, flatten, dense, dropout layers, and convolution layers with a fixed 7x7 kernel size. We employed the RMSprop optimizer with a learning rate of 1e-4 to improve the model's performance. The VGG-16 model has a deep network topology with many convolutional

layers that are designed to learn complex features, and it has about 138 million parameters. Next, after importing the VGG-16 model, it must be trained through multiple iterations

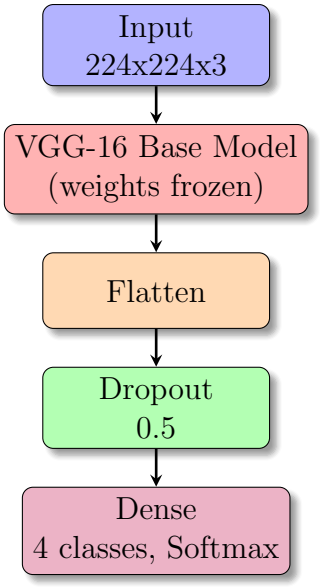

Figure 4.19: Enhanced VGG-16 Model Architecture

(called Epochs in Machine learning). After running the VGG-16 model for 30 epochs,94%

```
vgg16.evaluate(test)
```

```
41/41 [==============================] - 12s 287ms/step - loss: 0.1632 - accuracy: 0.9405
[0.1632436066865921, 0.9405034184455872]
```

Figure 4.20: Screenshot showing the results of epochs in Proposed CNN model (20-30 epochs)

validation accuracy were achieved.

4.2.2 Xception Model building and training

It is pertinent to highlight that to attain high accuracy on Xception, a comparable architecture to VGG-16 was constructed. Despite VGG-16 not yielding a satisfactory outcome, the Xception model delivered a desirable result. The aforementioned observation is in line with recent research that suggests that transfer learning via pre-trained models aids in the development of deep neural networks. Xception, which is a deep convolutional neural network, has demonstrated notable performance in a variety of visual recognition tasks. Its success can be attributed to its depth and the utilization of depthwise separable convolutions, which minimize computational cost while enhancing model accuracy.
In order to enhance the accuracy of the model, the input shape of the image has been set to

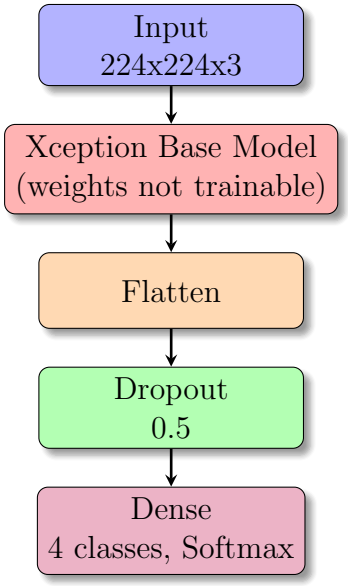

Figure 4.21: Xception Model Architecture

224×224 with an RGB color channel. This configuration has been chosen based on empirical evidence that suggests that this input shape yields better results. In addition, the Xception model has been trained for a total of 30 epochs. During each epoch, the model processes a batch size of approximately 178, computed as 5713//32. The validation steps, computed as 1311//32, are set to approximately 41. This approach has been shown to achieve satisfactory results in machine learning tasks involving image classification.

```
xception.evaluate(test)
```

```
41/41 [==============================] - 7s 164ms/step - loss: 0.1034 - accuracy: 0.9733
[0.10340148955583572, 0.9733028411865234]
```

Figure 4.22: Screenshot of Evaluating the Xception Model

After training it for 30 epochs, 99% training accuracy and 97% validation accuracy have been achieved.

4.2.3 ResNet50 Model Building and Training

The same methodology has been applied to load the ResNet50 Model as VGG-16 and Xception model. The ResNet50 model architecture was built by adding a flatten layer that converts the data into a 1-dimensional array for inputting to the next layer. The 'he_uniform' scaling initializer was used, and batch normalization was applied to the layer before or after the activation function in the Dense layer. The fully connected dense layer was used to

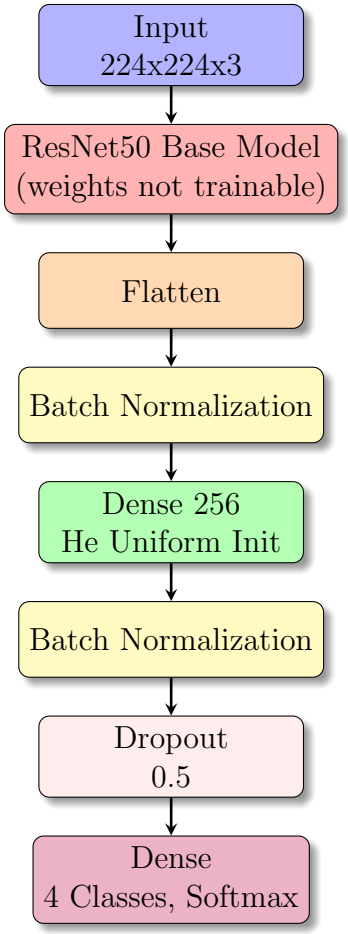

Figure 4.23: ResNet50 Model Architecture with Custom Layers

change the vector's dimension, receiving input from all neurons of its previous layer. Categorical cross-entropy was chosen as the loss function to quantify the difference between two probability distributions. The Adam optimizer was used to minimize the function and solve optimization problems. The same methodology as the previous two pre-trained models was applied to train the ResNet50 Model, fitting and training it for 30 epochs. Let's assess the accuracy by evaluating the validation data.

```
resnet50.evaluate(test)
```

```
41/41 [==============================] - 6s 131ms/step - loss: 0.1685 - accuracy: 0.9481
[0.168480783700943, 0.9481312036514282]
```

Figure 4.24: Screenshot of Evaluating the ResNet50 Model

Following this, 99% training accuracy and 94% validation accuary achieved by the ResNet50 model.

4.2.4 Inception-V3 Model Building and Training

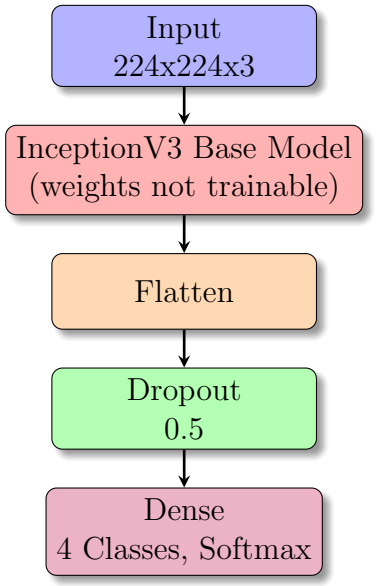

Figure 4.25: InceptionV3 Model Architecture with Custom Layers

After training the model for 30 epochs, 99% training accuracy and 96% validation accuracy were achieved. 4.26 illustrates the evaluation of Inception-V3 model.

```
inceptionV3.evaluate(test)
```

```
41/41 [==============================] - 6s 135ms/step - loss: 0.1237 - accuracy: 0.9664
[0.12365180253982544, 0.966437816619873]
```

Figure 4.26: Screenshot of Evaluating the Inception-V3 Model

5 Experiment

5.1 Enviornment Setup

Preparation of the environment is crucial for the successful implementation of the model, involving the importation and installation of necessary packages and libraries. The model is constructed, trained, and executed using the TensorFlow backend developed by Google Brain Wadhai et al. (2018). TensorFlow, an open-source framework widely utilized for building deep learning models and neural networks, offers a comprehensive API. Notably, prominent companies such as Airbnb, Uber, Snapchat, and Dropbox have integrated TensorFlow into their software libraries, underscoring its widespread adoption. Keras, a renowned neural network library, is compatible with the TensorFlow backend and is lauded for its intuitive interface and rapid prototyping capabilities. It supports both CPU and GPU and provides two primary model types: Sequential and Model. The "Model" in Keras represents a linear stack of layers, akin to the Sequential Model Keras Documentation (2019). Moreover, Keras offers diverse optimization methods like RMSprop, Adam, and SGD. Unlike many TensorFlow libraries written in C++ and CUDA, Keras libraries are Python-based, making them preferred for deep learning applications in Python Heller (2019). Environment setup necessitates the importation of essential packages and libraries. Core libraries such as Keras, TensorFlow, sklearn, and matplotlib are commonly employed. Within these libraries, packages like cv2, os, pyplot, layers, recall_score, f1_score, accuracy_score, train_test_split, Keras pre-processing, sklearn metrics, Keras sequential, and Keras optimizers are included. These packages facilitate various tasks such as automatic file operations, mathematical computations, and image processing operations like resizing, rotation, display, and cropping. Additionally, they provide tools for data visualization, efficient looping iterators, iterator handling, dataset splitting into training and testing subsets, and generation of confusion matrices and accuracy scores.

5.2 Data Description

Our collection is mostly composed of two datasets: ***figshare*** and ***kaggle***. This combined dataset, which comprises 7022 images from human brain MRI scans, is divided into four categories: pituitary, meningiomas, gliomas, and no tumor. Within a single folder, each photo is arranged into sub-folders based on its unique classification. To make testing and training easier, we now need to separate the data into separate files for each.

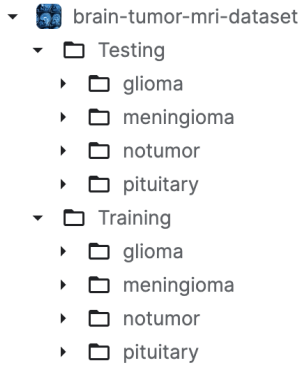

Figure 5.1: Screenshot of Dataset folder hierarchy

The dataset that is used to train the model during the development stage is known as the training dataset, and it is an essential part of machine learning. On the other hand, the test dataset is the one that is reserved for assessing the model's performance and is not altered during the training process. To guarantee the model's generalizability and avoid overfitting to the training dataset, evaluation must be conducted using an independent test dataset.

Plotting an image of each brain tumor type:

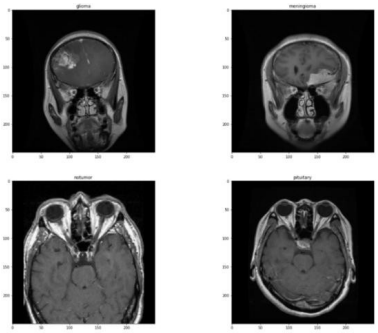

Figure 5.2: Example of Brain MRI images from the Dataset

Table 5.1: Number of Images for Each Tumor Type

Tumor Type	Number of Images (Testing)	Number of Images (Training)
Glioma	300 images	1321 images
Meningioma	306 images	1339 images
No tumor	405 images	1595 images
Pituitary	300 images	1457 images

5.3 Techniques of the Data Pre-processing

5.3.1 Image Ratio

The CNN model's input layer demands that the images be of the size (224, 224). Nonetheless, as figure 5.3 shows, there are variances in the images' height and breadth, as well as in the dimensions of the black corners. Unfortunately, some broad photos may appear deformed when resized. Let's look at the ratio distributions (ratio = width/height) to address this.

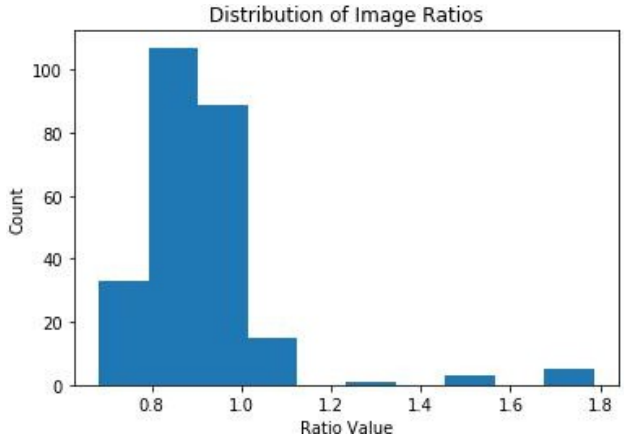

Figure 5.3: Histogram of image distributions

By examining Figure 5.3, we can see that there is a significant variation in image sizes. The majority of images have a ratio value between 0.7 and 1.1, indicating that they are taller. However, a small portion of images have ratio values between 1.2 and 1.8, indicating that they are wider. Since the CNN model and other pre-trained models only accept input of 224 by 224 pixels, the data must be pre-processed, including image cropping, before being fed into the model.

5.3.2 Cropping the MRI Images

To standardize the data, the initial step involves cropping the images. To accomplish this, we have utilized the technique outlined in the "pyimagesearch" blog. This method is effective in

identifying the pertinent region within the image. Subsequently, the utmost Northern, Eastern, Western, and Southern (x, y) coordinates are identified along the contour, as detailed below.

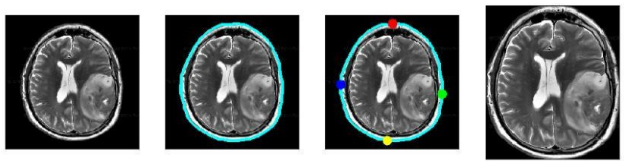

Figure 5.4: Cropping of Brain MRI Images

- Obtain the primary picture from the dataset.
- Find the brain's most pronounced shape.
- Note that the contour is represented by a Numpy array of (x, y) coordinates, then plot the extreme points on each of the four contour edges.
- Use those extreme points to crop the image.
- For each picture in the collection, repeat these steps.

5.4 Data Augmentation

The photos have been cropped, and are now ready to be entered into the model. However, because the existing dataset is too tiny, it must be augmented using data augmentation techniques before building the model. The photos are resized, rotated, flipped, zoomed, and subjected to other random modifications to make the most out of the limited data samples. This guarantees that the model will never see the same picture twice. To specify the required parameters for this procedure, refer to Keras The Keras Blog (2016) and its "ImageDataGenerator" class.
The number of images is greatly expanded once the method is performed to each and every image in the training dataset, creating a final dataset that is appropriate for training models. The code implementation of Data Augmentation is shown in 5.5, along with the output that shows the number of augmented images.

There is a tumor in this sample photograph. Let's apply data augmentation to it to observe the results of the method. Several augmented images were produced after the sample image was enhanced; these are displayed in figure 5.7.

```
Found 5712 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
```

Figure 5.5: Screenshot showing the number of images to be augmented

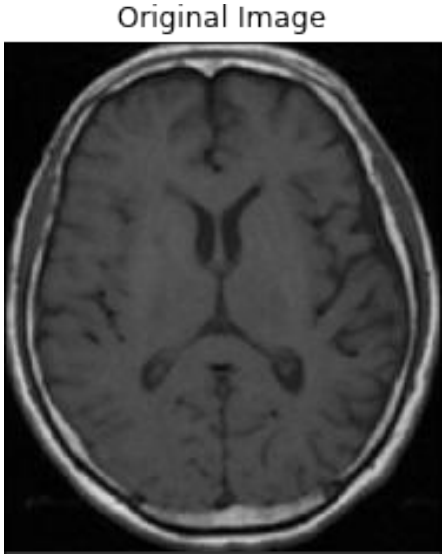

Figure 5.6: Sample of original Brain MRI Image

5.5 Model Performance

Based on the specified criteria of input data size, model complexity, accuracy, model loss, accuracy graph, and confusion matrix, five deep learning models underwent evaluation to identify and categorize brain cancers. The experiments were conducted utilizing the Python programming language on a desktop system equipped with a Shinelon Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz processor, 32 GB RAM, and an NVIDIA GeForce GTX 1050ti GPU. The datasets employed in the experiments are listed in Table 5.2. The parameters and number of convolutional layers for each model are detailed in Table 5.1. Comparative analysis revealed that the suggested CNN model boasts the fewest layers and network parameters among the pre-trained models considered. Notably, the Xception model stands out with 53 times more network parameters and three times greater depth compared to the suggested model. While the architectures of the VGG-16, Xception, and Inception-V3 models align closely with that of the VGG-16, they exhibit greater depth. Despite this, the suggested CNN model maintains lower complexity in terms of parameter count relative to other pre-trained models. Specifically, the ResNet50 model possesses three times as many relevant parameters as the suggested CNN model. Conversely, the Xception and Inception-V3 models demonstrate comparable quantities and depths of network parameters.

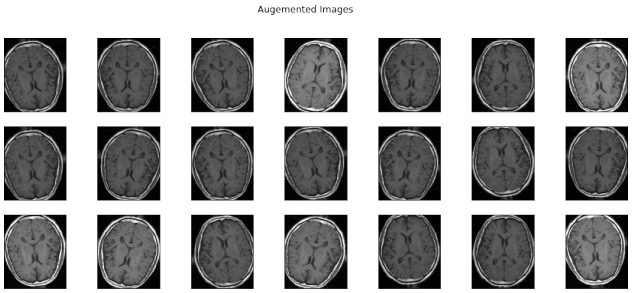

Figure 5.7: Augmented images

Table 5.2: Comparison of the Models in Terms of Convolutional Layers and Parameters

All the Model	Number of Convolutional Layers and Number of Parameters	
Proposed CNN model	5	17,706,224
VGG-16	13	14,815,044
Xception	71	21,262,892
ResNet50	48	49,681,540
Inception-V3	48	22,007,588

From the previous chapter, after training all the models, very good results have been achieved.

Table 5.3: Performance Metrics of all the Models presented

Models	Training Accuracy	Validation Accuracy	Training Time (estimated)
Proposed CNN model	99%	98%	46sec 29sec
VGG-16	96%	94%	31sec 26sec
Xception	99%	97%	29sec 26sec
ResNet50	95%	94%	30sec 25sec
Inception-V3	99%	96%	32sec 25sec

In the medical field, accuracy is of the utmost importance as it can greatly impact a patient's life. To aid in this, various pre-trained models were utilized to create a CNN model from scratch. Upon examining table 5.3, it is evident that the proposed CNN model yields significantly higher accuracy in both training and validation compared to the other pre-trained models. While these models do provide good results, they are not sufficient in detecting brain tumors, which is a critical illness in today's world. Thus, it is imperative to prioritize the highest level of accuracy.

Moreover, the loss function in machine learning gauges a model's capacity to forecast the

desired result, or "ground truth." The suggested CNN model outperforms other pre-trained models in terms of prediction accuracy, as table 5.4 illustrates.

Table 5.4: Loss and Accuracy of all the Models presented

Models	Loss	Accuracy
Proposed CNN model	0.0518	98%
VGG-16	0.1696	94%
Xception	0.0929	97%
ResNet50	0.2147	94%
Inception-V3	0.1173	96%

5.6 Plot the Performance and Result

There are two key measures that determine how well a model is performing: Model Accuracy and Model Loss. The `matplotlib` library was utilized to display the graphs showing these metrics. **Model Accuracy**: This is the main measure utilized for assessing classification models. The following formula determines the model's accuracy:

$$\text{Accuracy} = \left(\frac{\text{Total Accurate Predictions}}{\text{Number of Predictions Overall}} \right) \tag{12}$$

Accuracy improves as the model's loss decreases, indicating that the model is better at making correct predictions. Conversely, as the loss increases, the accuracy typically decreases.

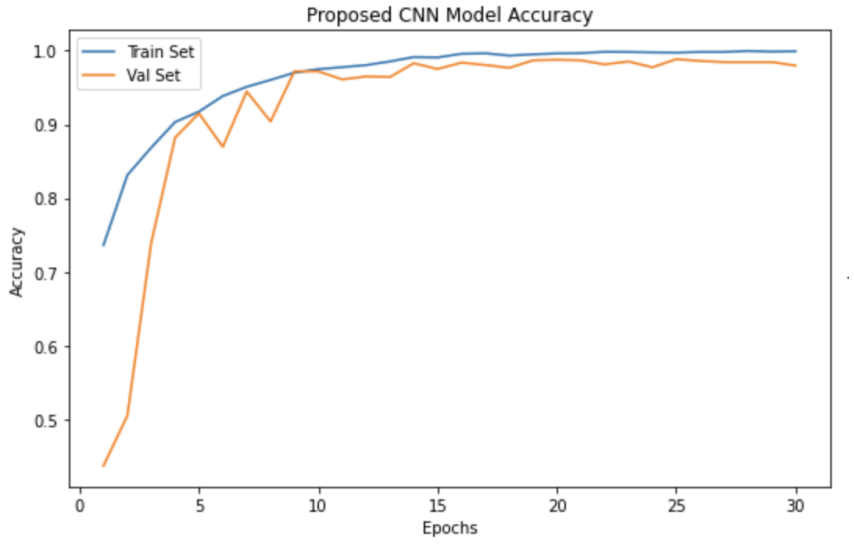

Figure 5.8: Proposed CNN model accuracy graph

By observing figure 5.8, the validation accuracy started from a meager number. After five epochs, it catches the training accuracy. The training accuracy and validation accuracy is closer to each other, and it does not overfit after running the model for 30 epochs.

Model Loss: For this assignment, a multi-class classification model with a loss function and two or more output labels had to be developed. The output label is a one-hot encoding with values in the range of 0 to 1. If the output label is in integer form, it can be converted to categorical encoding using the Keras.utils to_categorical method.The loss function works to improve the performance of the machine learning model by serving as a learning tool. If there is a large discrepancy between the expected and actual results, the loss function will yield a very high number. By observing figure 5.9, the validation loss has started from a

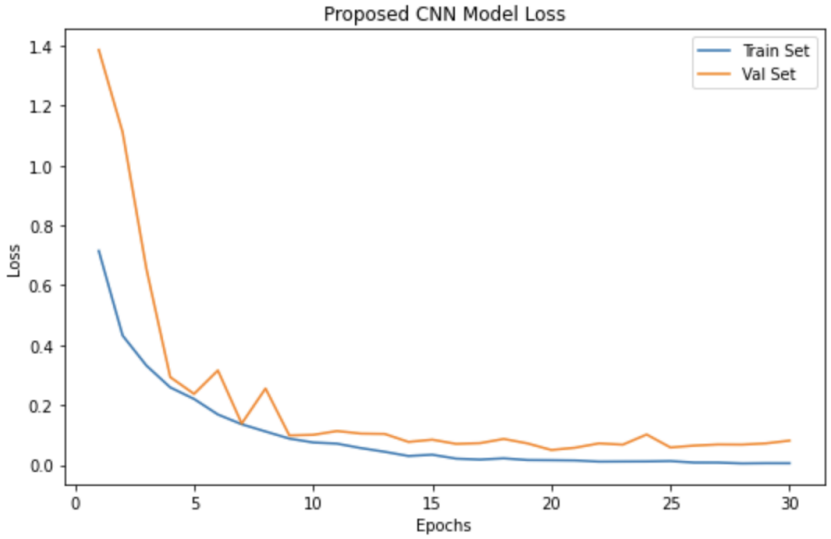

Figure 5.9: Proposed CNN model loss graph

very high number, but it reduces epoch by epoch, and it is also closer to the training loss after training the model for 30 epochs.

5.7 Confusion Matrix when tested on a Testing dataset

A confusion matrix is also known as an error matrix, which helps to speed up statistical data processing and make it easier to interpret results through clear data visualization. In order to assess the performance of the implementation algorithms used in the experiment, confusion matrices were created for each model, such as the suggested CNN model, using the sklearn library. The actual level is represented by the y-axis, while the expected level is represented by the x-axis.

Now we have the confusion matrix we obtained for the suggested CNN model in figure

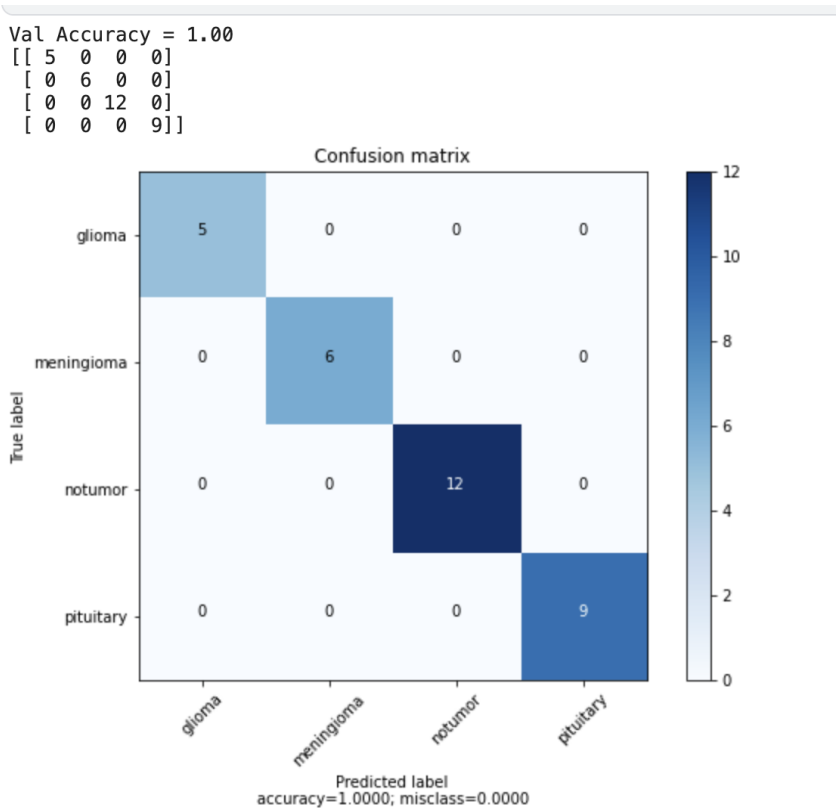

Figure 5.10: Confusion Matrix of proposed CNN model

5.10. The model used Testing data to generate a confusion matrix, achieving a validation accuracy of 100%. It successfully categorizes tumors into glioma, meningioma, no tumor, and pituitary. During each iteration, 32 images are classified from the training data. Running the loop 100 times, occasional incorrect predictions were made, primarily involving meningioma and pituitary tumors. Differentiating between meningioma and glioma remains challenging due to their benign nature. The confusion matrix of misclassifications is depicted in Figure 5.11.

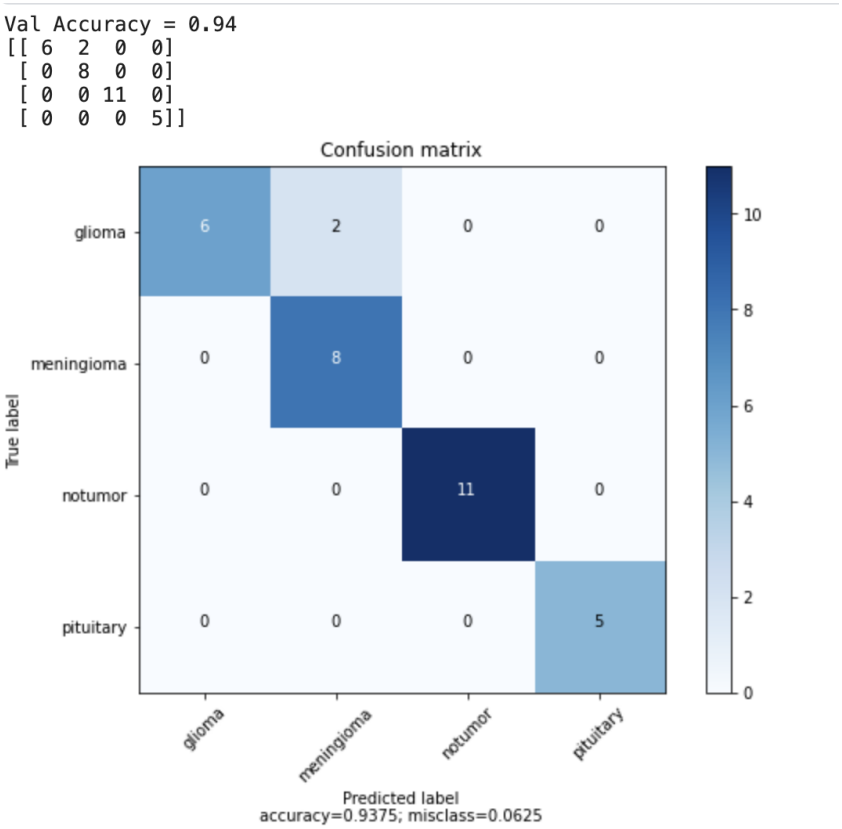

Figure 5.11: CNN model predicts wrong confusion matrix

5.8 Comparing the suggested CNN model with models that have already been trained

The suggested CNN model was compared in two ways to other pre-trained models, namely,

1. The accuracy and loss of the model graph and
2. confusion matrix

5.8.1 The Accuracy and Loss of the Model Graph

The model loss and accuracy graph for all the pre-trained models I used in this experiment have been plotted. The same approach has been used by using the matplotlib library to plot the chart. The graph of every model based on 30 epochs has been plotted. Figure 5.12 shows the comparison graph between the proposed CNN model and VGG-16 and the Xception model I used in this experiment based on the Model loss and accuracy. Figure 5.13 shows the comparison graph between the proposed CNN model and ResNet50, and the Inception-V3 model has been used in this experiment based on the Model loss and accuracy.

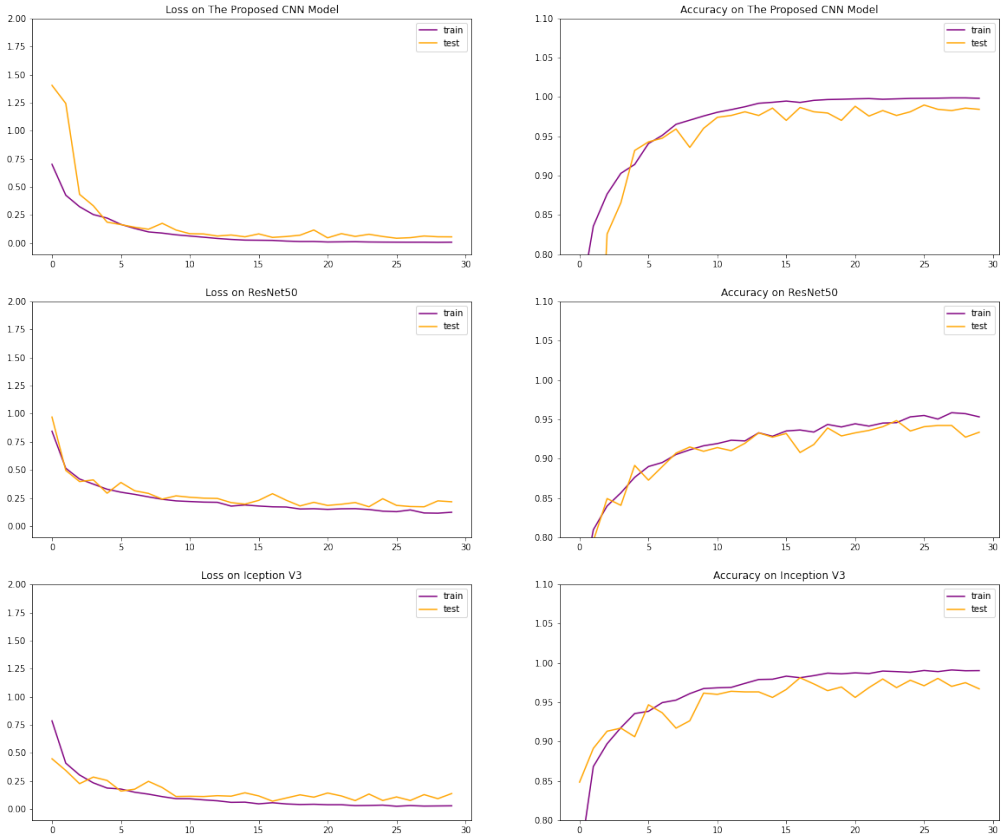

Figure 5.12: The suggested CNN model, VGG-16 model, and Xception model's model loss and accuracy graph

Model Loss comparison: The accuracy increases with a reduced loss function. In the beginning, epochs proposed CNN model's loss function is higher than other pre-trained models, but when it trains through more epochs, the loss function decreases enormously than other models. After running it for 30 epochs, 0.0054 loss function has been achieved for the proposed model, VGG-16 achieved 0.1182, Xception achieved 0.0186, ResNet50 achieved 0.1181, and Inception-V3 achieved 0.0219 loss function. Moreover, the proposed CNN model's training loss and testing loss was much closer, but it did not over-fit because the training loss is lower than the testing loss.

Model Accuracy comparison: Accuracy depends on the loss function. From the previous comparison, the proposed CNN model achieved the lowest loss function. Moreover,

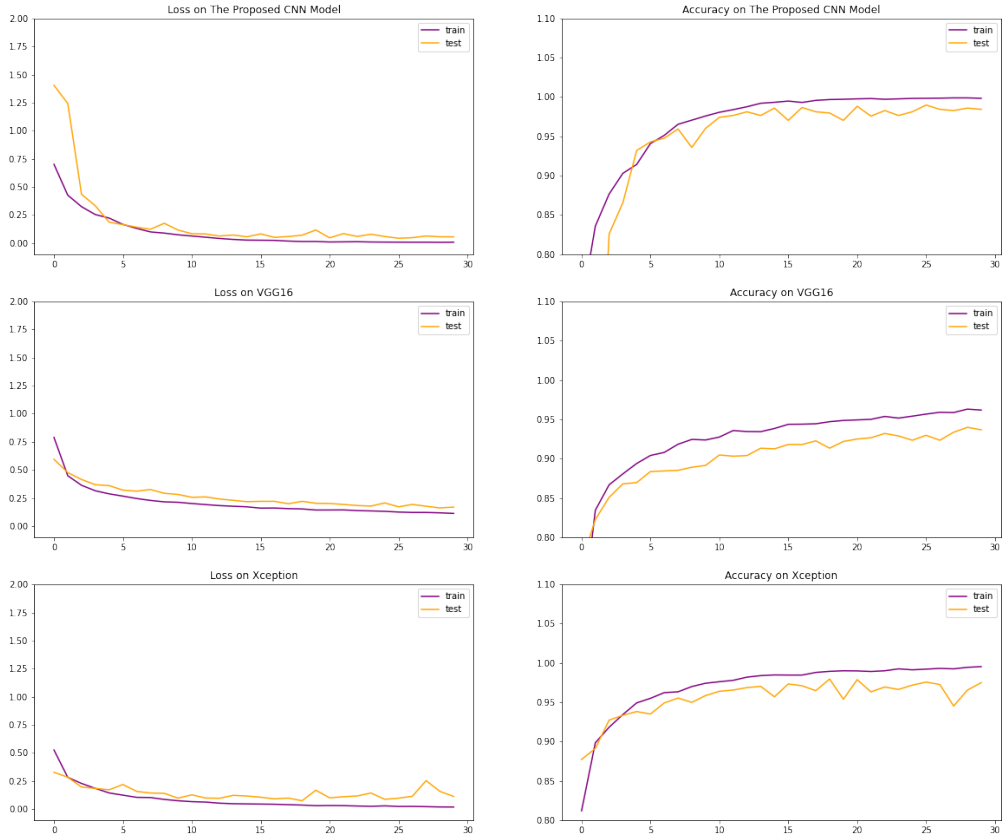

Figure 5.13: The suggested CNN, ResNet50, and Inception-V3 models' model loss and accuracy graphs

because of this, it achieved the highest accuracy on the proposed CNN model. For the proposed CNN model training and testing, accuracy is much closer, but the testing accuracy will never overcome the training accuracy; else, it will face the over-fitting problem. By observing the graph, the proposed CNN model provides better performance.

5.8.2 Confusion Matrix for Transfer Learning Models

we extend the analytical framework established for the proposed convolutional neural network (CNN) model to evaluate the performance of several pre-trained models, namely VGG-16, Xception, ResNet50, and Inception-V3. The evaluation focuses on the models' ability to classify brain tumor images, utilizing confusion matrices as a visual tool to assess each model's performance based on validation accuracy.

Figure 5.14 presents the confusion matrix for the VGG-16 model. Analysis of this matrix reveals that the VGG-16 model occasionally struggles to differentiate between the absence of a tumor and the presence of a meningioma. Specifically, the model achieves a validation accuracy of 94%, with a misclassification rate of 0.0625.

Similarly, Figure 5.15 displays the confusion matrix for the Xception model, which shows that three photos were incorrectly identified by the model. Notably, it misclassified cases of glioma as pituitary tumors or meningiomas. Such errors may cause misunderstandings in medical environments. The Xception model yielded a misclassification rate of 0.0938 and a validation accuracy of 91%.

The ResNet50 model's confusion matrix is shown in Figure 5.16. This matrix, which shows two cases of misclassification out of 32 studied photos, highlights a particular difficulty in differentiating between gliomas and meningiomas. The misclassification rate was found to be 0.0625, and the model attained a validation accuracy of 94%.

Lastly, the Inception-V3 model's confusion matrix, illustrated in Figure 5.17, shows that this model also faced difficulty in differentiating between glioma and meningioma in three out of 32 cases. The Inception-V3 model attained a validation accuracy of 91%, with a misclassification rate of 0.0928.

These findings underscore the nuanced performance of each model in the context of brain tumor classification. While each model demonstrates high overall accuracy, the detailed examination of confusion matrices reveals specific areas where model performance can be further optimized, particularly in distinguishing between closely related tumor types. This analysis not only highlights the strengths and limitations of each pre-trained model but also underscores the importance of continuous refinement and validation within the field of medical image analysis.

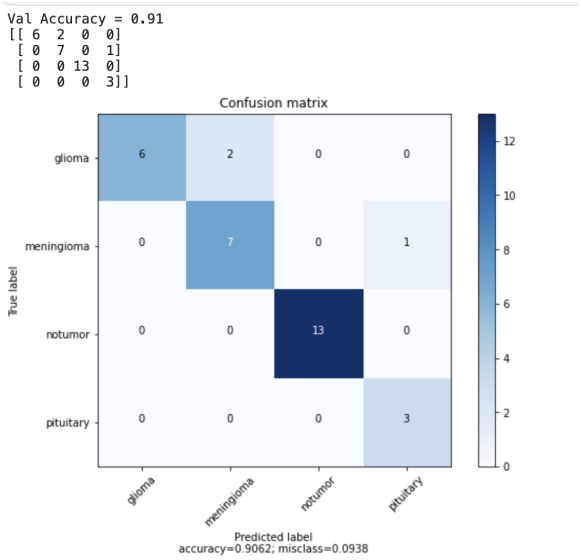

Figure 5.14: The VGG-16 model's confusion matrix

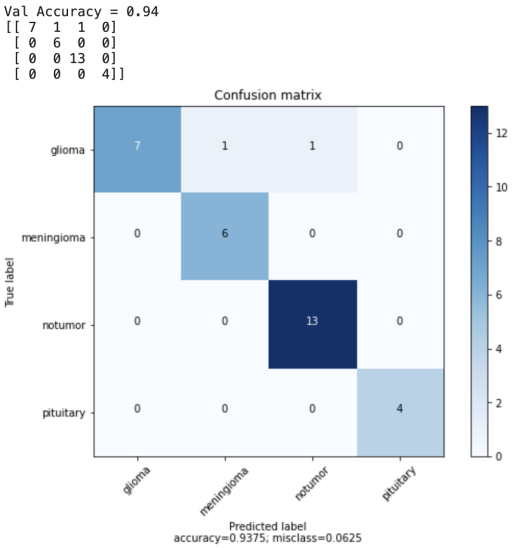

Figure 5.15: The Xception Model's Confusion Matrix

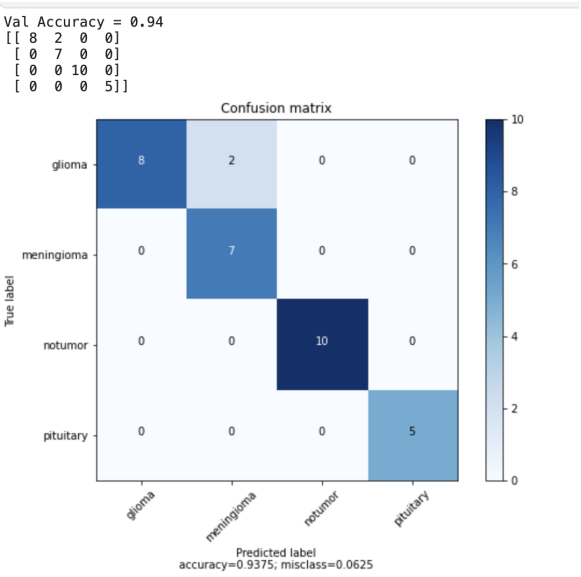

Figure 5.16: The ResNet50 Model's Confusion Matrix

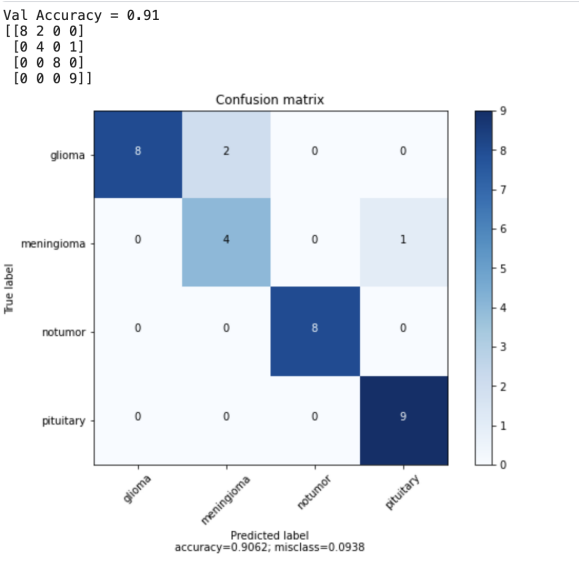

Figure 5.17: The InceptionV3 Model's Confusion Matrix

6 Conclusion

The research offers a novel method for classifying and identifying various types of brain tumors from MRI images. This inventive technique first defines the region of interest using picture edge recognition methods, then employs data augmentation to enhance the size of the training dataset. Next, a Convolutional Neural Network (CNN) model is proposed for precise and efficient classification of brain tumors. The results demonstrate that the suggested CNN model outperforms current pre-trained models such as VGG-16, Xception, ResNet-50, and Inception-v3, even with the use of a small dataset. It also attains flawless precision. Additionally, the CNN model has faster training times because of its lower parameter needs, which contribute to its higher computing efficiency.

The prognostic relevance of brain tumors in patients may be significantly impacted by the suggested system. Better pre-processing methods and thorough hyper-parameter tuning can lead to even greater gains in the model's efficiency. For issues requiring categorical classification, the suggested system is capable of classifying the tumor types pituitary, glioma, meningioma, and no tumor. To assess how well the suggested model performed in comparison to other pre-trained models, the model's accuracy and loss graphs as well as its confusion matrix were plotted.

In later studies, Ensemble Learning will be used to combine multiple models into a more effective meta-model. In order to attain the better prediction levels that are promised, this approach requires a high-performance Graphics Processing Unit (GPU) and a CPU, but it also requires a significant amount of processing power and time. Consequently, the proposed approach indicates promise for early identification of dangerous diseases in other clinical domains associated with medical imaging, such as breast and lung cancer, which have notable global mortality rates.

References

(2019). Center for biomedical image computing & analytics (cbica). http://braintumorsegmentation.org/. Accessed on 5 November 2019.

Abiwinanda, N., Hanif, M., Hesaputra, S., Handayani, A., and Mengko, T. R. (2019). Brain tumor classification using convolutional neural network. In *World Congress on Medical Physics and Biomedical Engineering*. Springer.

Akkus, Z., Galimzianova, A., Hoogi, A., Rubin, D., and Erickson, B. (2017). Deep learning for brain mri segmentation: State of the art and future directions. *J. Digit. Imaging*, 30:449–459.

American Association of Neurological Surgeons (2019). Classification of Brain Tumours. https://www.aans.org/en/Media/Classifications-of-Brain-Tumors. Accessed: 30 November 2019.

Basheera, S. and Ram, M. S. S. (2019). Classification of brain tumors using deep features extracted using cnn. *J. Phys.*, 1172:012016.

Cancer Treatments Centers of America (2019). Brain Cancer Types. https://www.cancercenter.com/cancer-types/brain-cancer/types. Accessed: 30 November 2019.

Carlo, R., Renato, C., Giuseppe, C., Lorenzo, U., Giovanni, I., and Domenico, S. (2019). Distinguishing functional from non-functional pituitary macroadenomas with a machine learning analysis. In *Mediterranean Conference on Medical and Biological Engineering and Computing*, pages 1822–1829. Springer.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1800–1807, Honolulu, HI.

Cinar, A. and Yldrm, M. (2020). Detection of tumors on brain mri images using the hybrid convolutional neural network architecture. *Med. Hypotheses*, 139:109684.

Das, S., Aranya, R., and Labiba, N. (2019). Brain tumor classification using convolutional neural network. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*.

Devkota, B., Alsadoon, A., Prasad, P., Singh, A. K., and Elchouemi, A. (2017). Image segmentation for early stage brain tumor detection using mathematical morphological

reconstruction. In *6th International Conference on Smart Computing and Communications (ICSCC)*, Kurukshetra, India.

George, D. N., Jehlol, H. B., and Oleiwi, A. A. H. (2015). Brain tumor detection using shape features and machine learning algorithms. *International Journal of Scientific & Engineering Research*, 6(12).

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Heller, M. (2019). What is keras? the deep neural network api explained. *Online Source*. Retrieved from https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html.

Hendrik, R. E. (2005). Glossary of mr terms. American College of Radiology.

Hinton, G. (2012). Neural networks for machine learning online course lecture 6a. Coursera. Available from: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

Keras Documentation (2019). Keras: The python deep learning library. https://keras.io/. Retrieved October 2019.

Khawaldeh, S., Pervaiz, U., Rafiq, A., and Alkhawaldeh, R. (2018). Noninvasive grading of glioma tumor using magnetic resonance imaging with convolutional neural networks. *J. Appl. Sci.*, 8:27.

Luo, L. et al. (2019). Adaptive gradient methods with dynamic bound of learning rate. In *Proc. of ICLR 2019*.

National Brain Tumor Society (2020). Quick brain tumor facts. https://braintumor.org/brain-tumor-information/brain-tumor-facts/.

National Health Service (2020). Brain Tumours. https://www.nhs.uk/conditions/brain-tumours/.

Peltarion Platform (2019). Categorical crossentropy loss function. https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy.

Preston, D. C. (2006). Magnetic resonance imaging (mri) of the brain and spine from basics. http://casemed.case.edu.

R. A. Novellines, M. D. (2004). *Squire's Fundamentals of Radiology*. UPR, six edition.

Robbins, H. and Munro, S. (1951). A stochastic approximation method. *Ann. Math. Stat.*, 22:400–407.

Romeo, V., Cuocolo, R., Ricciardi, C., Ugga, L., Cocozza, S., Verde, F., et al. (2020). Prediction of tumor grade and nodal status in oropharyngeal and oral cavity squamous-cell carcinoma using a radiomic approach. *Anticancer Res.*, 40:271–280.

Saha, S. (2018). A comprehensive guide to convolutional neural networks. Towards Data-science.

Sajjad, M., Khan, S., Khan, M., Wu, W., Ullah, A., and Baik, S. W. (2019). Multi-grade brain tumor classification using deep cnn with extensive data augmentation. *J. Comput. Sci.*, 30:174–182.

Simonyan, K. and Zisserman, A. (2014a). Very deep convolutional networks for large-scale image recognition. *arXiv*.

Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition.

Song, Y., Ji, Z., Sun, Q., and Yuhui, Z. (2016). A novel brain tumor segmentation from multi-modality mri via a level-set-based model. *Journal of Signal Processing Systems*, 87.

Stewart, B. and Wild, C. (2014). *World Cancer Report 2014*. IARC Nonserial Publ, Lyon, France.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Talo, M., Baloglu, U. B., Yldrm, O., and Acharya, U. R. (2019). Application of deep transfer learning for automated brain abnormality classification using mri images. *Cognitive Systems Research*, 54:176–188.

The Keras Blog (2016). Building powerful image classification models using very little data. Retrieved October 2019.

Wadhai, V., Chancalani, A., Kachwalla, M., Shinde, P., Katare, R., and Agarwal, A. (2018). Classification of brain mri images for cancer detection using deep learning. *International Journal of Advanced Research in Computer and Communication Engineering*, 7(4). ISO 3297:2007 Certified.

Özyurt, F., Sert, E., Avci, E., and Dogantekin, E. (2019). Brain tumor detection based on convolutional neural network with neutrosophic expert maximum fuzzy sure entropy. *Measurement*, 147.