Unveiling the Power of Deep Learning for Precise Brain Tumour Classification: Exploring Advanced Architectures and Optimized Hyperparameters

Ansa Baby
22540050
Manchester Metropolitan
University

Abstract— **The proper classification of brain tumours is critical for diagnosis, treatment planning, and patient care. In recent years, deep learning has emerged as a powerful tool for image classification tasks, with the potential to improve the precision and accuracy of brain tumour classification. This study aims to demonstrate the effectiveness of deep learning in achieving accurate brain tumour classification and investigate the impact of hyperparameter tuning on deep learning algorithms by examining advanced architectures and optimised hyperparameters. In the experiment, a self-designed model called ResNeXt, which is a variation of the ResNet architecture incorporating the concept of "cardinality" using grouped convolutions, was used along with other models such as VGG, AlexNet, and ResNeSt In addition, an investigation into hyperparameter optimisation techniques is conducted in order to determine the best setup for improved model performance. The impact of architectural complexity and hyperparameter optimisation on classification accuracy is thoroughly investigated, shedding light on how hyperparameters affect deep learning algorithms.**

This study's findings shed light on how hyperparameter tuning can have a significant impact on the performance and efficiency of deep learning models in brain tumour classification. Furthermore, these findings provide important information about how deep learning techniques can be used to improve the diagnosis and treatment of brain tumours. This research contributes to the development of more precise and effective methods for brain tumour classification, ultimately leading to better patient outcomes by advancing the field of medical image analysis.

Keywords— Deep learning, Brain tumour classification, Comparative study, Advanced architectures, Hyperparameter tuning, Training strategies, Performance evaluation, medical imaging, Visualization techniques.

I. Introduction

The prevalence of brain tumours adds to the disease's burden and makes accurate diagnosis and treatment planning difficult[1]. The accurate and timely classification of brain tumours is critical for directing treatment plans and improving patient outcomes. Traditionally, brain tumour classification relied heavily on the subjective and time-consuming manual interpretation of radiological images. Recent advances in deep learning methods, on the other hand, have opened up new avenues for the automated and precise classification of brain tumours. The purpose of this study is to use deep learning techniques to classify images of brain tumours, with a focus on the four major tumour types: meningioma, pituitary, glioma, and no tumour[1]

Deep learning is a subset of machine learning. Deep learning, a subset of machine learning, has produced impressive results in a variety of image recognition and classification tasks[3]. We hope to build a reliable and precise brain tumour classification system by leveraging deep learning models' ability to directly learn intricate patterns and features from raw image data [1].To accomplish our classification goal, we will investigate various deep learning architectures, including well-known models like VGG, ResNet, and AlexNet[3].

These architectures have demonstrated their efficacy in a variety of image classification tasks, making them suitable candidates for our investigation. We can accelerate the training process and improve the model's ability to generalise to our specific brain tumour dataset by utilising the pre-trained weights of these architectures, which have been trained on massive datasets such as ImageNet[3].

Furthermore, hyperparameter optimisation will be a key component of our research[5]. To determine the optimal configuration that maximises classification accuracy, we will systematically investigate various hyperparameter settings such as learning rate, batch size, weight decay, and regularisation methods[5]. The model's performance must be fine-tuned in order to accurately recognise different tumour types and underlying patterns, which can be accomplished through hyperparameter optimisation. This study's findings have far-reaching implications in the fields of medical image analysis and brain tumour diagnosis[2].

We can help medical professionals diagnose patients more quickly and accurately by developing an accurate and automated deep learning-based classification system, resulting in more focused treatment plans and better patient outcomes[2]. Furthermore, the findings of this study have the potential to advance the field of medical imaging research and pave the way for new advances in deep learning methods for classifying brain tumours.

A. *Research Objective*

The primary goal of this study is to investigate the effectiveness of advanced deep learning architectures and optimised hyperparameters for achieving precise and accurate brain tumour classification. This study aims to improve the performance of brain tumour classification systems and contribute to the advancement of medical image analysis in the field of neuro-oncology by exploring and comparing different deep learning models and tuning their hyperparameters.

B. *Report Structure*

Section II: Dataset Description,
Section III: Related Work
Section IV: Methodology,
Section IV.E: Model Architecture
Section IV.F: Model Training & Epoch,
Section V: Effect of Hyperparameter Tuning,
Section VI: Regularization & Overfitting - Early Fitting,
Section VII: Global & Local Minima

II. Dataset Description And Analysis

A. *Dataset Description*

The dataset used in this study consists of 7,023 human brain MRI images divided into four categories: glioma, meningioma, no tumour, and pituitary. The Data was divided into two subfolders- Testing and Training. This dataset contains a diverse and extensive collection of brain MRI images, allowing deep-learning models to learn intricate patterns and features associated with each tumour type. We hope to develop accurate and reliable deep-learning models from this dataset that will significantly contribute to the field of medical image analysis and improve the diagnosis and treatment of brain tumours.

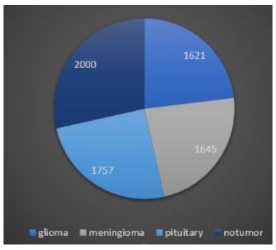


Figure 2 Dataset distribution

B. *Data Splitting*

The entire data frame is then divided into test, train and val using train test split, allowing for efficient model development and evaluation. ***Size:*** The training subset contains 5,712 images, which provides a large number of samples for the models to learn from. The testing subset contains 1,311 images, allowing for a thorough evaluation of the model's performance. Finally, the validation subset

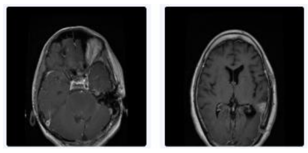


Figure 5 Glioma Sample image

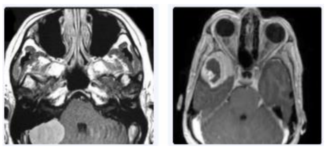


Figure 4 Meningioma sample image

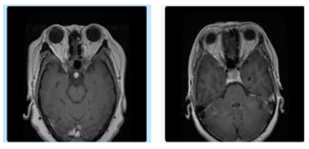


Figure 6 Pituitary sample image

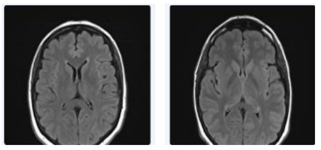


Figure 3 No Tumor sample image

contains 1,408 images, making it easier to fine-tune and optimise the model's hyperparameters. The distribution of images across the training, testing, and validation subsets is as follows: glioma (972, 324, 325), meningioma (986, 328, 331), no tumour (1,200, 400, 400), and pituitary (972, 324, 325) respectively. Figures above show the sample images in the dataset.

Attributes: The main attribute of interest in this dataset is the brain MRI images themselves. Each image represents a two-dimensional representation of the brain, capturing its internal structure and potential tumour presence. The images are grayscale or colour images, depending on the specific dataset format.

Deep Learning Tasks: This dataset is well-suited to a variety of deep learning tasks involving image classification and analysis. It can specifically be used for image recognition, image segmentation, and object detection. Deep learning models can learn intricate patterns and features from raw image data, allowing for accurate brain tumour classification.

Noise & Outliers: In the context of brain MRI images, noise refers to unwanted artefacts or distortions caused by factors like imaging hardware or patient motion. Outliers represent rare or abnormal cases that deviate significantly from typical brain MRI images. While noise and outliers can impact deep learning models' performance, the dataset used in this study has not shown any noticeable noise or outlier characteristics. This could be attributed to the careful acquisition and preprocessing of the MRI images, ensuring high-quality data. However, to ensure robustness, noise detection and reduction techniques, such as denoising filters or outlier detection algorithms, can be implemented as a preventive measure during the data preprocessing stage.

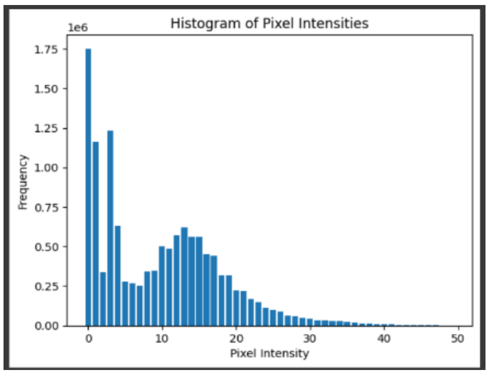


Figure 1 Pixel Intensity Graph

Statistical analysis: In the statistical analysis of the dataset, the means and standard deviations of the pixel intensities provide insights into the brightness and contrast of the brain MRI images. For this dataset, the calculated means are tensor([-0.0407, -0.0375, -0.0304]), and the standard deviations are tensor([0.0240, 0.0246, 0.0244]). These values help understand the distribution of intensities across the colour channels. Additionally, a histogram plot was generated to visualize the distribution of pixel intensities, highlighting any patterns or variations. The histogram provides a visual representation of the frequency of different intensity levels in the dataset.[Figure 1 Pixel Intensity Graph]

Justification for Dataset Selection: This dataset was chosen because of its large size, a diverse range of brain tumour types, and availability of labelled data. The dataset's size allows for effective deep-learning model training, and the inclusion of various tumour types ensures comprehensive coverage and generalisation.

Deep Learning Descriptive Data Analysis: In the context of deep learning, descriptive data analysis entails exploring and comprehending the characteristics of the dataset, such as its size, attributes, distribution, and statistical properties. This analysis helps in gaining insights into the dataset's composition, identifying potential challenges or biases, and guiding preprocessing steps or model selection

C. *Data Augmentation*

Hypothesis: Introducing data augmentation techniques to the training dataset will improve the deep learning models' performance and generalisation capabilities for brain tumour classification. The augmented dataset will provide additional variations and increase the diversity of the training samples by applying various transformations and modifications to the images, such as resizing, random noise addition, and standardisation. This increased variability will assist the models in learning more robust and discriminative features, reducing overfitting and improving the models' ability to classify brain tumour images accurately.

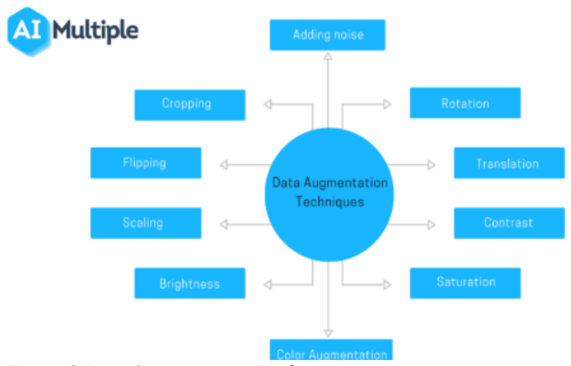


Figure 8 Data Augmentation Techniques

The focus of this research's data augmentation section was on improving the dataset's diversity and generalisation capabilities through various transformations. Using the transforms, the training images were transformed in a series of ways. Function for composing. The images were first resized to a consistent dimension of 32x32 pixels to ensure size consistency across the dataset. Random noise was added to the images to introduce variability and promote robust learning. Tensors, the fundamental data structure used in deep learning frameworks, were created from the transformed images. To ensure standardised input, each image channel was normalised using specific mean and standard deviation values, namely [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225][6].
Similar transformations, such as resizing, tensor conversion, and standardisation, were applied to the testing images, with the same mean and standard deviation values as the training images[6]. ***Analysis:*** These data augmentation techniques, when applied to both training and testing images, play an important role in reducing overfitting, improving model performance, and allowing deep learning models to capture the underlying patterns and features of brain tumour images [6]. By adding diverse and standardised variations to the dataset, the models become more robust, adaptable, and capable of accurate classification and diagnosis.

III. Related Work

Deep learning algorithms have been used to detect and classify brain tumours, which is an active area of research. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are two deep learning algorithms that have shown promise in this domain. In addition, the utilization of quantum machine learning has also emerged as

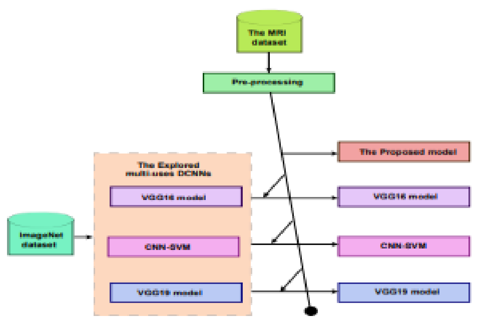


FIGURE 4. VGG16, VGG19 and CNN-SVM model were pre-trained with ImageNet dataset before the training process with the used MRI dataset, however, the proposed model were trained only with the used MRI images dataset.
Figure 7 Background work Model -2

a potential approach for improving accuracy and efficacy in brain tumour detection.

Musallam et al. [7] investigate the use of deep learning algorithms in the detection of brain tumours using magnetic. resonance imaging (MRI). The study proposes the use of CNNs, specifically the U-Net architecture, for segmenting brain tumours from MRI scans. The U-Net architecture, which consists of an encoder-decoder structure with skip connections, has shown promising results in capturing spatial features and reconstructing tumour segmentation masks. The study does, however, highlight some limitations, such as the need for large, annotated datasets and potential challenges in the interpretability and explainability of the CNN models.

Garg et al.[8] explores the application of various machine learning models, including CNNs, random forests, neural networks, K-nearest neighbours (KNN), and decision trees,

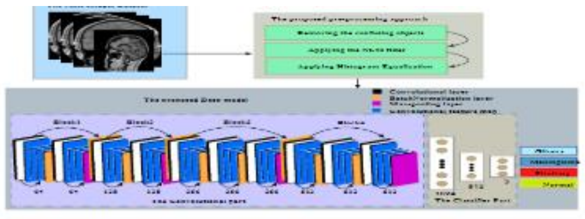


FIGURE 1. The MRI images dataset is entered into the proposed pre-processing steps. After that, it is entered into the proposed architecture in the training phase and in the testing phase to classify them as glioma, meningioma, pituitary, and normal.

Figure 9 BackGround work Model -1

for brain tumour detection. A hybrid ensemble classifier called KNN-RF-DT is also proposed. The study evaluates these models using 2556 brain tumour images, with feature extraction techniques including SWT, PCA, and GLCM. The results show high precision (97.73%), specificity (97.60%), sensitivity (97.04%), and reliability (97.41%). However, limitations such as rapid tumour growth, difficulty in tumour segmentation due to fuzzy borders, and difficulties in accurate feature extraction and selection are identified. To optimize the performance of deep learning algorithms, various techniques can be employed. The optimisation of parameters such as learning rate and batch size, known as hyperparameter tuning, can improve the model's performance and generalisation. Regularization techniques, such as L1 or L2 regularization and dropout, can prevent overfitting and enhance generalization. Transfer learning, which uses pre-trained models on large-scale datasets, can also speed up convergence and improve performance even with limited training data.

Obeidavi et al. [9] used a dataset of 2000 MR images to develop a CNN-based residual network for the early detection of brain tumours. They used the BRATS 2015 MRI dataset, and the results for the residual networks were encouraging. The proposed model's accuracy was 97.05%. They also took other metrics into account, achieving a mean accuracy of 97.05%, a global accuracy of 94.43%, a mean IoU of 54.21%, a weighted IoU of 93.64%, and a mean BF score of 57.027%. During training, one hundred epochs were used to improve performance.

Using 3D MR images, ***Khalil et al. [10]*** proposed a modified two-step dragonfly algorithm for brain tumour segmentation. Variations in tumour size and structure pose the greatest challenges in identifying and segmenting the early stages of brain tumours. To overcome these challenges, the researchers employed a two-step dragonfly algorithm to precisely extract the original contour point. They used the BRATS 2017 3D MR brain tumour dataset to obtain the results using the proposed model. They achieved about 5% higher accuracy than the previous researchers, who conducted a nearly identical study. To validate their findings, they also applied a variety of techniques, including fuzzy C-means, SVM, and random forests. To evaluate their results, they considered the metrics of accuracy, precision, and recall. They obtained an accuracy of 98.20%, a recall of 95.13%, and a precision of 95.13% after evaluating their proposed mode

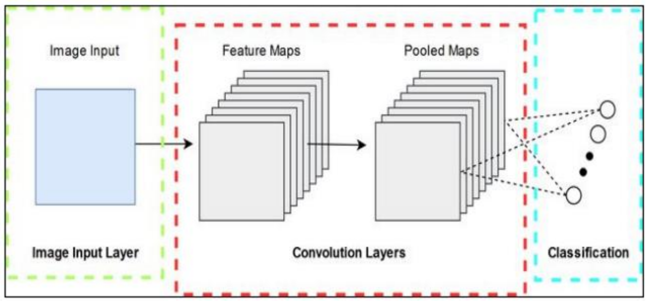


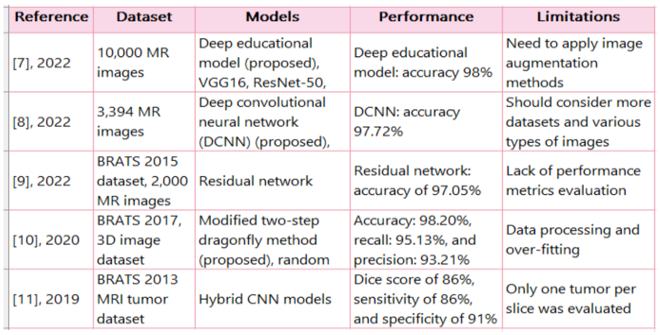
Figure 10 Single Convolution Layer

Sajid et al. [11] introduced a hybrid CNN model to detect brain tumours using BRATS MR images. The analysis and validation were performed on the effectiveness of a unique two-phase training method and sophisticated regularization approaches, such as dropout.

Their suggested hybrid model combined two- and three-path networks, which enhanced the model's performance. The model may be effective for a variety of segmentation tasks, according to the capacity analysis of the CNNs, and better performance may be obtained with more training instances. After examining their model, they discovered that their Dice score was 86%, their sensitivity was 86%, and their specificity was 91%.

An analytical review of previous work highlights the limited availability of annotated brain tumour datasets with diverse acquisition characteristics, affecting the generalization capability of deep learning models. Furthermore, the interpretability and explainability of deep learning models, particularly CNNs, remain difficult to achieve. Understanding and interpreting these models' learned representations and decision-making processes is critical for their practical application in clinical settings.

Given the limitations and challenges identified in the existing literature, the goal of this project is to contribute by rigorously justifying and critically evaluating two deep learning algorithms applicable to the chosen brain tumour dataset. CNNs and RNNs, for example, will be optimised using techniques such as hyperparameter tuning, regularisation, and possibly transfer learning. By addressing the limitations of previous work and leveraging the strengths of deep learning algorithms, the project aims to improve the accuracy and efficacy of brain tumour detection and classification, ultimately contributing to advancements in the field.



Reference	Dataset	Models	Performance	Limitations
[7], 2022	10,000 MR images	Deep educational model (proposed), VGG16, ResNet-50,	Deep educational model: accuracy 98%	Need to apply image augmentation methods
[8], 2022	3,394 MR images	Deep convolutional neural network (DCNN) (proposed),	DCNN: accuracy 97.72%	Should consider more datasets and various types of images
[9], 2022	BRATS 2015 dataset, 2,000 MR images	Residual network	Residual network: accuracy of 97.05%	Lack of performance metrics evaluation
[10], 2020	BRATS 2017, 3D image dataset	Modified two-step dragonfly method (proposed), random	Accuracy: 98.20%, recall: 95.13%, and precision: 93.21%	Data processing and over-fitting
[11], 2019	BRATS 2013 MRI tumor dataset	Hybrid CNN models	Dice score of 86%, sensitivity of 86%, and specificity of 91%	Only one tumor per slice was evaluated

Figure 11 Literature Review Result

IV. Methodology

This study's methodology aims to investigate advanced deep learning architectures and optimised hyperparameters for accurate brain tumour classification. This included the following steps: dataset description, model selection, hyperparameter tuning, training procedure, evaluation metrics, experimental setup, results presentation and discussion, and model parameter change considerations. Each step was carefully designed and executed to ensure a systematic and rigorous approach to the research.

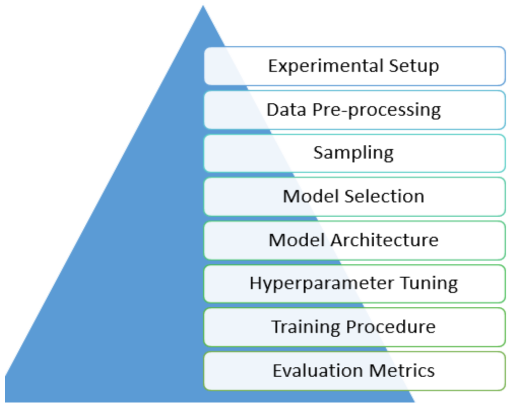


Figure 12 Methodology Steps

A. *Experimental Setup*

The tests were run on a machine equipped with an NVIDIA Tesla T4 GPU as part of the experimental setup. For training and evaluating the VGG19 architecture, the GPU provided efficient computational power. The CPU, an x86_64 processor, supported Python code execution and other necessary computations. The system's total memory capacity was 15.36GB, allowing it to handle large datasets and model parameters. The experiments were carried out on a Linux operating system version 5.15.107+, which provided a stable and reliable environment for neural network experiments. Python 3.10.11, as well as the TensorFlow and Keras libraries, were used to implement and train the VGG19 model. The combination of these hardware and software components provided an optimal setup for carrying out the experiments efficiently and accurately.

B. *Data- Pre-processing*

An extensive data pre-processing pipeline was used in this study to prepare the brain tumour dataset for deep learning model training. The raw MRI images were transformed, including resizing to a consistent resolution, cropping to remove unnecessary background, and augmenting the dataset with techniques such as rotation, flipping, and scaling. These transformations aimed to improve the generalisation capability of the trained models by increasing the diversity and variability of the dataset. Additionally, normalisation techniques were used to standardise the image pixel values, ensuring that the input data had a zero mean and unit the training process and the prevention of any bias towards specific intensity ranges.

C. *Sampling*

The training data was sampled using a Data Loader with a batch size of 32 and random shuffling, ensuring diverse samples in each batch during training.

D. *Model Selection*

Three different models have been implemented in this project to achieve the research goal: A self-designed architecture of ResNeXt Model, a proposed VGG11, VGG19, ResNet50, and Alex Net. These models were chosen based on their demonstrated efficacy in a variety of computer vision tasks, including image classification. Exploring and evaluating the performance of these models on brain tumour classification can provide valuable insights into their suitability and effectiveness for precise brain tumour classification.

E. *Model Architecture*

Experiments were conducted on two deep learning architectures, including at least one self-designed architecture called ResNeXt. ResNeXt is a variation of the ResNet architecture that incorporates the concept of cardinality, which was introduced as part of its design. The ResNeXt model was specifically designed by me to explore its performance and compare it with other established architectures in the experiments.[18]

1) *RetNeXt – Self-designed Architecture*

Hypothesis: It is hypothesized that ResNeXt, an extension of the ResNet architecture with grouped convolutions, will offer enhanced performance compared to other models. By introducing the concept of "cardinality," ResNeXt allows for more diverse feature representations and an improved capacity to learn intricate patterns in brain tumour images.

The model’s name, ResNeXt, contains Next. It means the next dimension, on top of the ResNet. This next dimension is called the “cardinality” dimension.[19] The ResNeXt architecture is a ResNet extension that incorporates the concept of "cardinality" as a key design element. In ResNet, each residual block uses standard convolutional layers, while in ResNeXt, each residual block is composed of ResNeXtBlocks, which incorporate grouped convolutions and multiple parallel branches[18]. The number of parallel pathways or branches within a block is referred to as its cardinality. It enables the model to capture different aspects of the input data by simultaneously learning diverse feature representations.[18]

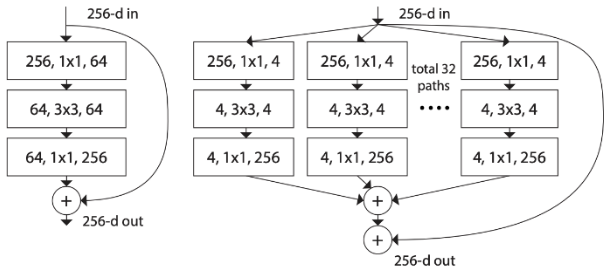


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

the ResNeXt model encompasses the overall structure of the ResNeXt architecture and the ResNeXtBlock is used within the residual blocks of the ResNeXt model.

The ResNeXtBlock class defines the residual block with grouped convolutions and cardinality. It is responsible for performing the grouped convolutions and aggregating the outputs from parallel branches. This class is used within the ResNeXt model.[18]

The ResNeXt class represents the overall ResNeXt model architecture. It is composed of multiple layers and blocks.

- Input Layer: No specific class definition is used here. It represents the input layer where images with the specified dimensions are passed into the model.
- Convolutional Layers: The initial convolutional layers are defined using the nn.Conv2d class, and they are responsible for generating feature maps. In the provided code, the self.conv1 variable represents the first convolutional layer of the ResNeXt model.
- Residual Blocks with Cardinality: The ResNeXt blocks are used within layer1, layer2, layer 3, and layer4 of the ResNeXt model. These layers are created using the _make_layer method, which uses the ResNeXtBlock class to construct the desired number of residual blocks. The block parameter in _make_layer specifies the usage of the ResNeXtBlock class.
- Feature Maps: The output from the residual blocks is passed through the layer self.layer1, self.layer2, self.layer3, and self.layer4, which represent the feature maps of the model.
- Global Average Pooling: The global average pooling operation is performed using the nn.AdaptiveAvgPool2d class, which reduces the spatial dimensions of the feature maps to a single value per feature map.
- Fully Connected Layers: The flattened features from the global average pooling are passed through the fully connected layers defined by self.fc, which performs the necessary transformations.
- Output Layer: The final output layer is defined by self.fc and produces the model's predictions with the specified number of classes.[18]

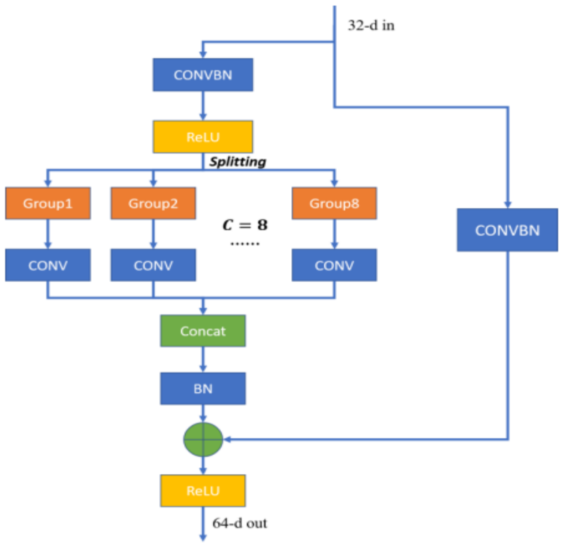


Figure 13 General structure of a ResNeXt block with cardinality = 8

With a cardinality of 32, the ResNeXtBlock is a critical component of the ResNeXt architecture. ResNeXt improves the model's representational power and ability to capture diverse feature representations by introducing the concept of cardinality. The second convolutional layer in the ResNeXtBlock performs grouped convolutions with a cardinality of 32, which means it divides the input channels into 32 groups and performs convolutions within each group separately. This enables the model to learn and combine data from multiple parallel pathways, resulting in a more diverse and rich set of features. In this architecture, the use of 32 as the cardinality value represents the level of parallelism and complexity in capturing feature representations.

2) *AlexNet*

Hypothesis : It is hypothesized that AlexNet, with its unique architecture comprising convolutional layers, max pooling, and fully connected layers, will demonstrate competitive performance for brain tumour classification. AlexNet was one of the pioneering deep learning models that achieved remarkable success in image classification tasks, and it is expected to leverage its ability to capture complex image features for accurate tumour classification.

AlexNet emerged as the champion of the ImageNet large-scale visual recognition challenge in 2012, as proposed by Alex Krizhevsky and his colleagues. It introduced a deeper architecture compared to the previous LeNet-5 model.[17] The feature extractor and classifier are the two main components of the AlexNet architecture. The feature extractor is made up of several convolutional layers, which are then followed by the max pooling and ReLU activation

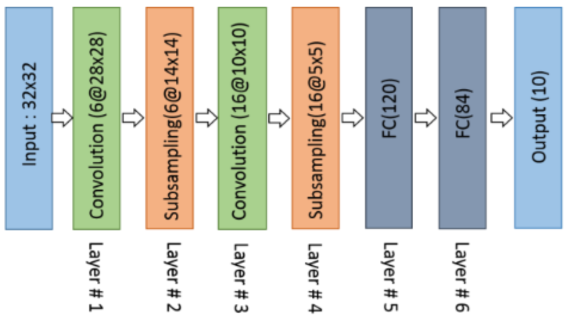


Figure 14 AlexNet

functions. The network receives a three-channel image as input. The first convolutional layer consists of 64 filters with kernel sizes of 3x3 and strides of 2, reducing the spatial dimensions of the input. It is followed by a max pooling layer with a 2x2 kernel. This pattern of convolutional layer, max pooling, and ReLU activation is repeated several times, increasing the number of filters gradually. The final max pooling layer reduces the spatial dimensions even further.[14]

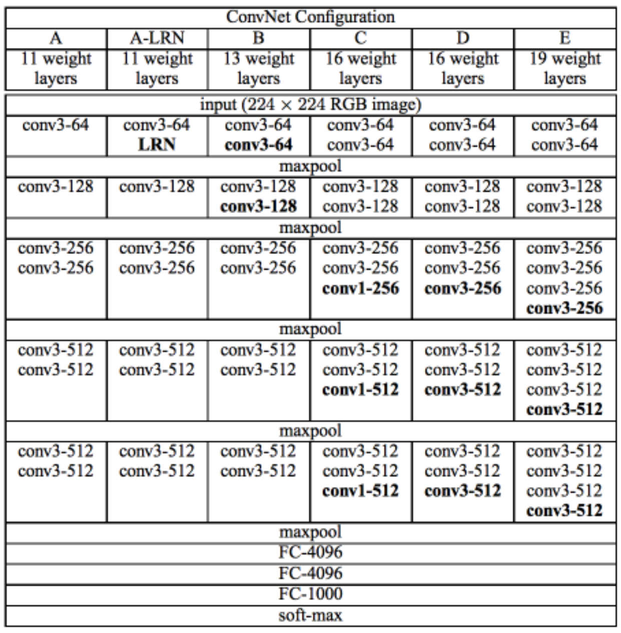
The classifier is a fully connected neural network that performs classification on the output of the final convolutional layer. It begins with dropout regularisation, which aids in the prevention of overfitting. The output of the final convolutional layer is flattened into a 1D vector and passed through fully connected layers activated with ReLU. From 4096 to the desired output dimension, the number of units in the fully connected layers decreases. The output predictions are generated by the final layer.[14]

AlexNet is distinguished by its deep convolutional layers, maximum pooling, and ReLU activations, which are followed by fully connected layers for classification. This architecture was widely used and had a significant impact on the advancement of deep learning for image classification tasks.[14]

3) *VGG:*

Hypothesis: It is hoped that VGG, with its deeper architecture and smaller convolutional filters, will be able to capture more intricate features from brain tumour images, resulting in greater accuracy. VGG's stacked convolutional layers extract hierarchical representations of the input, allowing for better tumour discrimination.

Simonyan and Zisserman's[12] VGG system is a deep convolutional neural network architecture that has gained popularity for its simplicity and effectiveness in image classification tasks. The VGG system's main feature is its deep structure, which enables it to learn intricate features and capture fine-grained details from images. VGG's fundamental architecture consists of a stack of convolutional layers with small receptive fields (3x3 filters), followed by max-pooling layers for spatial downsampling. This pattern of convolutional and pooling layers is repeated several times, resulting in a deep network with many parameters. For classification, the VGG system also employs fully connected layers with ReLU activation.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	**conv3-64**	conv3-64	conv3-64	conv3-64
maxpool					
conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128
		conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	**conv3-256**	conv3-256
					conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 15 VGG 11, 16 & 19 versions comparison

The number of layers and their configurations are flexible. The number of layers and their configurations can differ, with VGG11 being a specific configuration made up of 11 convolutional layers. The VGG system has achieved impressive performance on various image classification benchmarks by leveraging this deep architecture, demonstrating its effectiveness in learning, and representing complex visual patterns.

- ***VGG 11***

The VGG11 layer architecture is a variant of the VGG network, which is a deep convolutional neural network architecture designed for image classification tasks. The VGG11 architecture is made up of convolutional layers, max-pooling layers for downsampling, and fully connected layers for classification. [14]

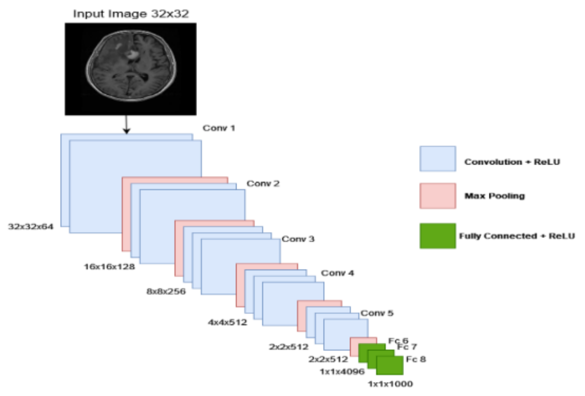


Figure 16 VGG11 Architecture

The VGG11 layer architecture:

- ***Input layer***

The input layer accepts a 3-channel image (typically RGB) of size 224x224 pixels as input.

- ***Convolutional Layers:***

The VGG11 architecture begins with two convolutional layers, each with 64 filters. Each convolutional layer extracts features at different spatial scales by applying a set of learnable filters to the input image. The filters have a small 3x3 receptive field and a stride of one, which means they move across the input image one pixel at a time. Following the convolutional layers is a Rectified Linear Unit (ReLU) activation function, which introduces nonlinearity into the network.[14]

- ***Max Pooling Layers:***

The Max pooling layer is added after every two convolutional layers to downsample the feature maps. The maximum pooling layers have a kernel size of 2x2 and a stride of 2, which reduces the spatial dimensions of the feature maps by half while retaining the most important features.
Layers that are completely connected:
Fully Connected Layers: The output of the last max pooling layer is flattened into a 1D vector and classified using fully connected layers. The VGG11 architecture has three fully connected layers, each with 4096 units. Except for the last, each fully connected layer is followed by a ReLU activation function. Dropout is used after the first two fully connected layers to prevent overfitting. Dropout randomly sets a portion of the input units to zero during training, which aids in model regularisation. The final fully connected layer generates the network's final output, which corresponds to the predicted probabilities for each class.
The output layer in the VGG11 architecture has the same number of units as the number of classes in the classification task, i.e., 4 here.
The VGG11 architecture has demonstrated strong performance on various image classification tasks, including the classification of brain tumour images, by stacking multiple convolutional and fully connected layers.[14]

- **VGG19**

The VGG19 layer architecture is another variant of the VGG network, which is a deep convolutional neural network architecture designed for image classification tasks.
The VGG network architecture is divided into five sections, each of which includes convolutional layers followed by max pooling. Block 1 employs two convolutional layers with 64 filters and 3x3 kernels, followed by ReLU activation. Following that, a 2x2 max pooling operation is performed. Block 2 includes two convolutional layers with 128 filters, as well as ReLU activation and max pooling. Block 3 consists of four convolutional layers each with 256 filters, ReLU activation, and maximum pooling. Block 4 uses four convolutional layers and 512 filters, as well as ReLU activation and max pooling. Block 5 brings the architecture to a close with four convolutional layers, ReLU activation, and max pooling. To ensure consistent spatial dimensions, adaptive average pooling is used.[14]
Fully connected layers are used for classification after the convolutional blocks. The first fully connected layer is made up of 4096 units that are activated by ReLUs and have a dropout probability of 0.5. The second fully connected layer includes 4096 units as well as ReLU activation and dropout. The output layer is the final fully connected layer, which contains the same number of units as the desired output classes. [14]
VGG11 has 11 layers, whereas VGG19 has 19 layers. With increasing numbers of filters in each block, both configurations follow a pattern of convolutional layers followed by max pooling. VGG19, on the other hand, has

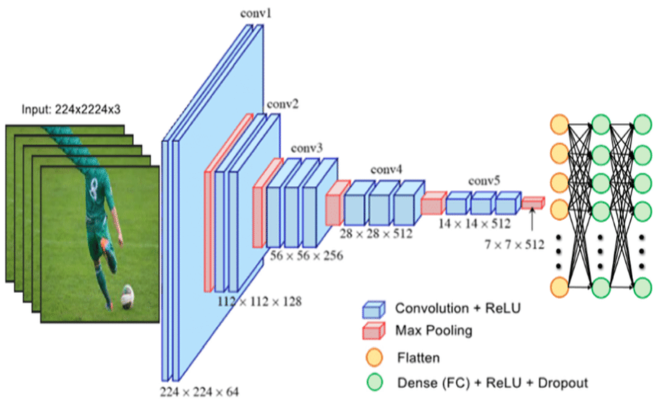
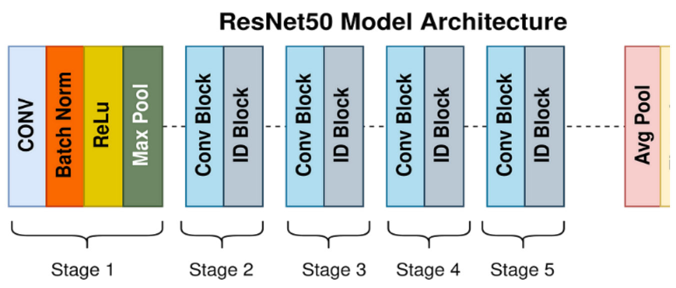


Figure 17 VGG19 Architecture

more convolutional layers and filters than VGG11. This increased depth enables VGG19 to capture more intricate features while also increasing computational complexity. Overall, VGG19 is a deeper and more complex network than VGG11, which may enable better representation learning at the expense of higher computational requirements.[14]

4) ***ResNet***

ResNet-50 is a deep convolutional neural network architecture that has received a lot of attention in the field of computer vision. It solves the problem of training very deep networks by introducing skip or residual connections. These connections allow the gradient to flow directly through the network, avoiding vanishing gradients and allowing for the training of deep networks with improved performance.[14]



The ResNet-50 architecture is implemented in the provided code using the ResNet class and the Bottleneck block. The ResNet class accepts as input a configuration, resnet50_config, which specifies the number of blocks and channels in each stage of the network. The ResNet-50 Bottleneck block is made up of three convolutional layers and is responsible for reducing spatial dimensions while increasing the number of channels. It also has a residual connection, which enables the network to learn residual mappings.[14]

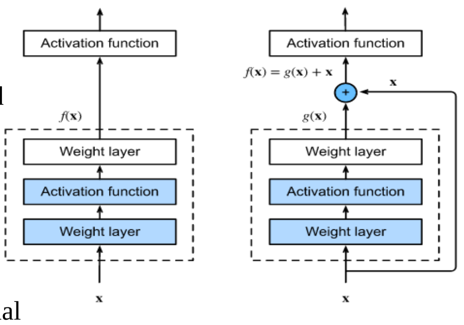


Figure 18 Residual Blocks of ResNet

The portion of a ResNet (left) within the dotted-line box directly learns the mapping f(x). The portion within the dotted-line box in a residual block (right) aims to learn the residual mapping g(x) = f(x) - x, where x represents the input to the block. By constructing the block in this manner, the network is aided in learning the identity mapping f(x) = x. When f(x) equals x, g(x) becomes zero, implying that only the upper layers' weights and biases, such as fully connected and convolutional layers, need to be adjusted. The solid line, also known as the residual or shortcut connection, directs the layer input 'x' to the addition operator, allowing the network to focus on learning the residual mapping[14]

F. *Model Training , Epoch & Cross Validation*

To train and evaluate a model, the provided code consists of functions and a training loop. A model architecture, accuracy calculation, training and evaluation functions, utility functions, and a training loop are all included. The calculate_accuracy function compares the predicted class to the ground truth labels to determine the accuracy of the model's predictions. The train function runs the training loop, optimising the parameters of the model based on the calculated loss and updating the gradients. The model's performance on a validation set is evaluated by the evaluate function. The epoch_time function computes the time elapsed between epochs. Finally, the get_predictions function retrieves the model's predictions, labels, and probabilities. The training loop iterates over a set number of epochs, measuring elapsed time, performing training and evaluation, and saving the best model based on validation loss.

a) Generating Ground Truth

I created a CSV file called 'ground_truth.csv' that contains code that generates ground truth labels. It loops through a test dataset, extracting image file paths and labels and writing them to a CSV file. The labels are converted into a binary format with four classes, and the resulting data is stored as a numpy array called 'y_test'. This generated ground truth can be analysed further or used for evaluation in machine learning tasks.

b) Cross Validation

a k-fold cross-validation strategy is implemented for deep learning model training. The dataset is split into training and validation sets using the KFold function from sklearn.model_selection. The model is trained for a specified number of epochs on each fold, and the best-performing model based on validation loss is saved. The average validation loss across all folds is calculated and printed as a measure of model performance.

G. *Model Evaluation & Comparison*

In order to assess the performance and compare the effectiveness of different models, a comprehensive evaluation process will be conducted. This evaluation will involve analysing various metrics and techniques to determine the strengths, weaknesses, and overall suitability of each model for the given task.[15]

My Evaluation Metrics Contain

- ❈ ROC & AUC Curve
- ❈ Confusion Matrix
- ❈ F - Measure Score Average
- ❈ Recall Average
- ❈ Precision Average
- ❈ Accuracy
- ❈ Accuracy Loss
- ❈ Predicted- Actual Labels Graph

ROC Curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation of a binary classification model's performance. At various classification thresholds, it plots the true positive rate (TPR) versus the false positive rate (FPR). The Area Under the ROC Curve (AUC-ROC) is a metric that quantifies the model's overall performance. For starters, they show the trade-off between the true positive rate and the false positive rate, allowing for an intuitive assessment of the model's performance across various threshold settings. Second, the AUC-ROC is a single scalar value that summarises the model's ability to differentiate between positive and negative instances, with a higher AUC-ROC indicating better performance.[15] Added some of the samples below.

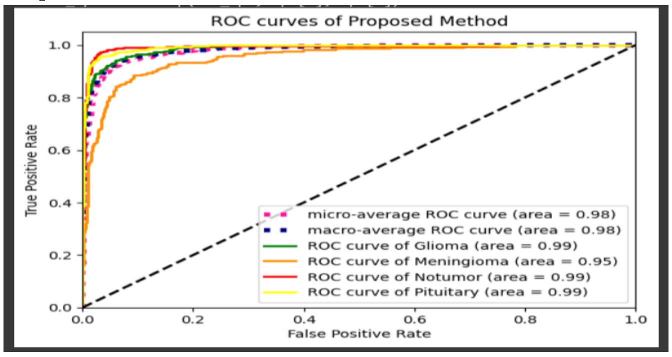
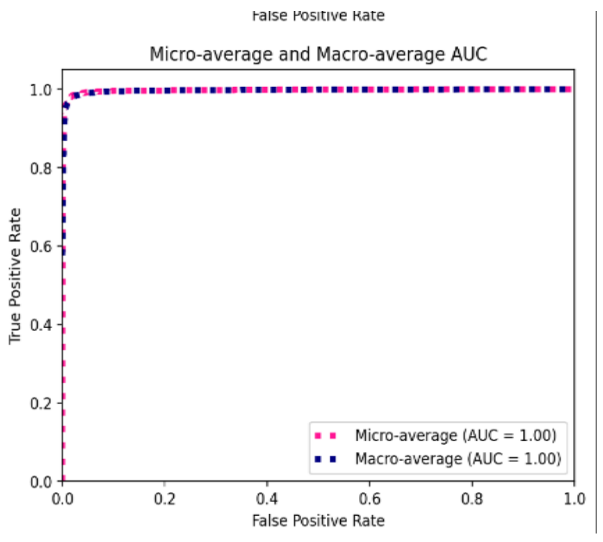
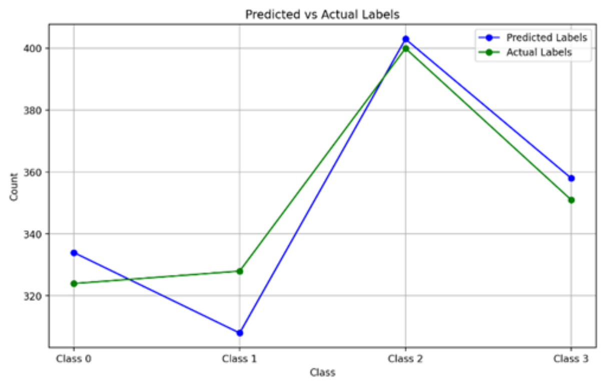


Figure 19 VGG 11 ROC Curve





Confusion Matrix

A confusion matrix is a tabular representation of a classification model's performance. True positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) are all calculated. TP denotes instances that were correctly classified as positive, FP denotes instances that were incorrectly classified as positive, FN denotes instances that were incorrectly classified as negative, and TN denotes instances that were correctly classified as negative. The confusion matrix provides valuable insights into the classification model's accuracy and effectiveness in distinguishing between different classes or categories by analysing these metrics.[15]

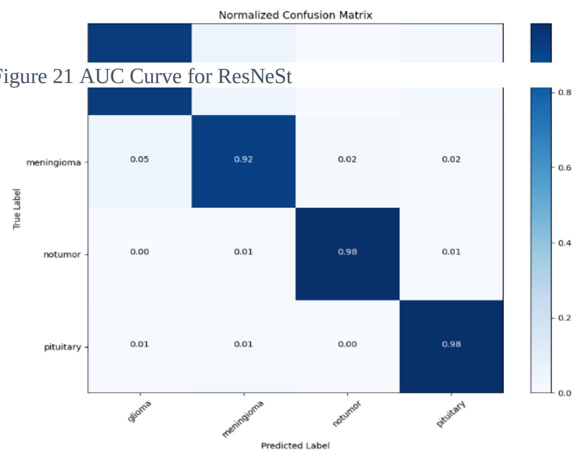
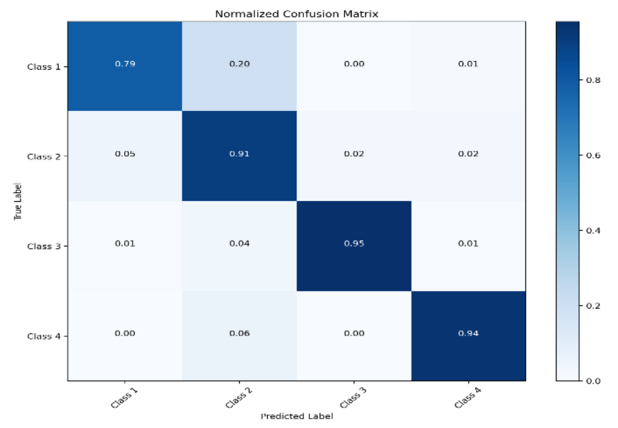
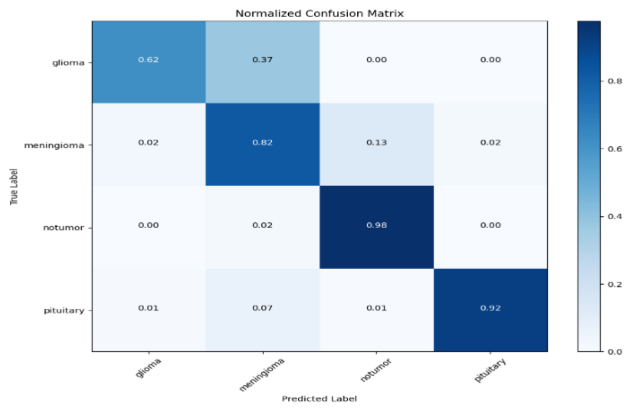
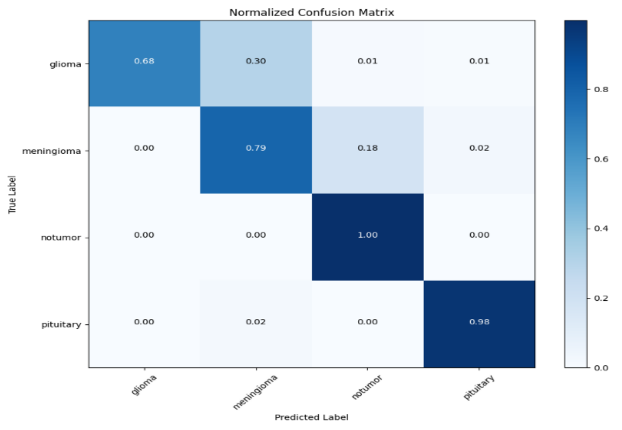


Figure 21 AUC Curve for ResNeSt

Figure 22 Predicted Vs Actual plot







From the above pictures, its proven that ResNet and ResNeXt have good prediction abilities. We need more analysis to come to a conclusion. *The square value in the diagonal shows the percentage of True positive values predidcted by the model. Darker the shade, more the true positive prediction.*

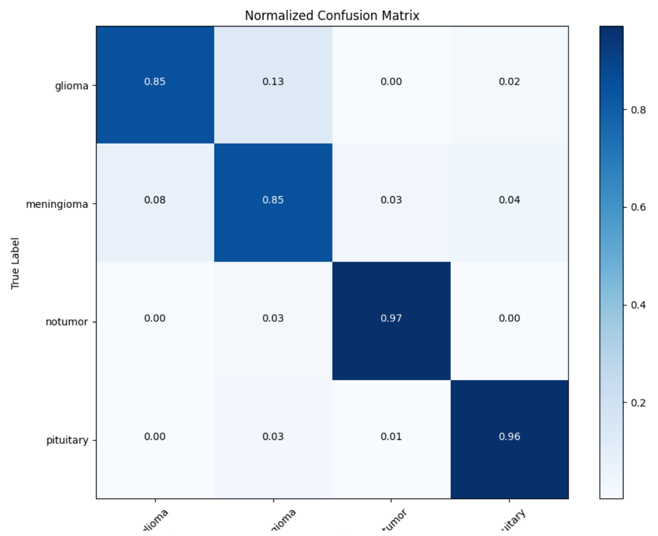


Figure 20 Confusion Matrix plots for AlexNet, VGG 11 and VGG19 Figrespectively

From Confusion Metrix, it is possible to find other evaluation matrix parameters like Precision, Recall and F-Measure. Other evaluation parameters are Accuracy Score and Accuracy loss.

Accuracy is a metric that measures the overall correctness of a model's predictions. It is calculated as the ratio of the number of correct predictions (True Positives and True Negatives) to the total number of predictions.

Accuracy = (Number of correctly predicted samples) / (Total number of samples)

Accuracy loss, also known as classification loss or cross-entropy loss, is a measure of the discrepancy between the predicted probabilities of the model and the true labels.

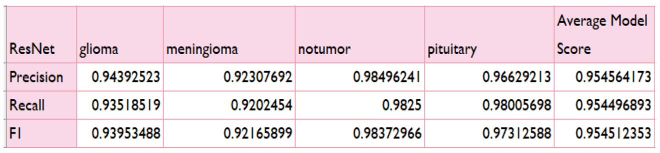
Accuracy Loss = (Sum of individual losses for all samples) / (Total number of samples)

Precision = TP / (TP + FP)
Recall = TP / (TP + FN)
F-measure = 2 * (Precision * Recall) / (Precision + Recall)

Where TP (True Positive)
FP (False Positive)
FN (False Negative)

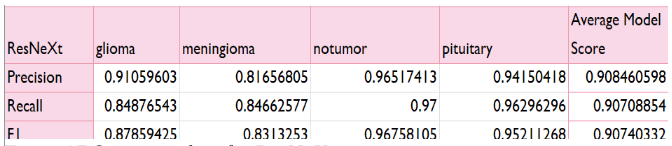
Precision is a metric that evaluates the model's ability to correctly identify positive instances among the instances predicted as positive. ***Recall***, also known as sensitivity or true positive rate, measures the proportion of actual positive instances that were correctly predicted by the mode. The ***F-measure, also called the F1 score***, is a combined metric that balances both precision and recall.

Since I have 4 classes of Brain Tumour, precision score, Recall and F- Measure will be an array of 4, which contains the scores per class. To get the score of the model, need to get the average.



ResNet	glioma	meningioma	notumor	pituitary	Average Model Score
Precision	0.94392523	0.92307692	0.98496241	0.96629213	0.954564173
Recall	0.93518519	0.9202454	0.9825	0.98005698	0.954496893
F1	0.93953488	0.92165899	0.98372966	0.97312588	0.954512353

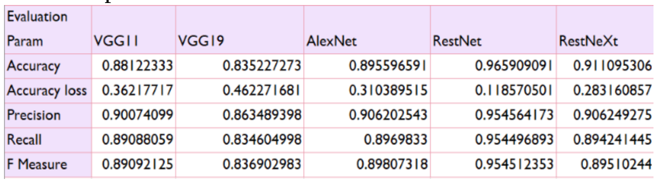
Figure 28 Score per class for ResNet



ResNeXt	glioma	meningioma	notumor	pituitary	Average Model Score
Precision	0.91059603	0.81656805	0.96517413	0.94150418	0.908460598
Recall	0.84876543	0.84662577	0.97	0.96296296	0.90708854
F1	0.87859425	0.8313253	0.96758105	0.95211268	0.90740332

Figure 27 Score per class for ResNeXt

The Complete evaluation Matrix of all models is.



Evaluation Param	VGG11	VGG19	AlexNet	RestNet	RestNeXt
Accuracy	0.88122333	0.835227273	0.895596591	0.965909091	0.911095306
Accuracy loss	0.36217717	0.462271681	0.310389515	0.118570501	0.283160857
Precision	0.90074099	0.863489398	0.906202543	0.954564173	0.906249275
Recall	0.89088059	0.834604998	0.8969833	0.954496893	0.894241445
F Measure	0.89092125	0.836902983	0.89807318	0.954512353	0.89510244

Figure 29 Evaluation Matrix

Based on the evaluation metrics provided, the ResNet model had the highest accuracy of 96.59%, indicating that it performed well in correctly classifying the dataset. The ResNeXt model also performed well, with an accuracy of 91.11%. VGG11 and VGG19, on the other hand, achieved lower accuracies of 88.12% and 83.52%, respectively.

When accuracy loss, which measures misclassifications, was considered, the ResNet model achieved the lowest value of 0.1185, indicating its ability to minimise errors. The highest accuracy loss was 0.3622 for VGG11, followed by 0.4623 for VGG19.

ResNet had the highest precision of 95.46%, which reflects the model's ability to correctly identify positive samples, followed by RestNeXt with 90.62%. AlexNet and VGG11 both had competitive precision values of 90.62% and 90.07%, respectively. ResNet and RestNeXt had the highest recall values of 95.45% and 89.42%, respectively, according to the recall metric, which measures the model's ability to correctly detect positive samples. VGG19 had the lowest recall value (83.46%).

When the F-measure, which balances precision and recall, was considered, ResNet had the highest value of 95.45%, followed closely by RestNeXt with 89.51%. AlexNet and VGG11 both had respectable F-measure values of 89.80% and 89.09%.

In summary, the ResNet model performed admirably across a wide range of evaluation metrics, including accuracy, accuracy loss, precision, recall, and F-measure. In terms of these evaluation measures, the ResNeXt model performed well as well, while VGG models performed relatively poorly.

V. Effect of Hyper Parameter Tuning

The ResNet model & RestNeXt models consistently outperform the other two models in terms of accuracy, precision, recall, and F-measure, according to the model evaluation. However, given our research goal of analysing the effect of hyperparameters on model performance, we chose a model with medium performance for hyperparameter testing. This method allows us to investigate the impact of various hyperparameter configurations on the model's predictive capabilities and gain insights into further optimising its performance.[5]

I repeatedly changed the below parameters and took the evaluation matrix and analysed the output.

1. The number of Epochs
2. Learning rate
3. Optimization Algorithm
4. Weight Decay
5. Loss Function

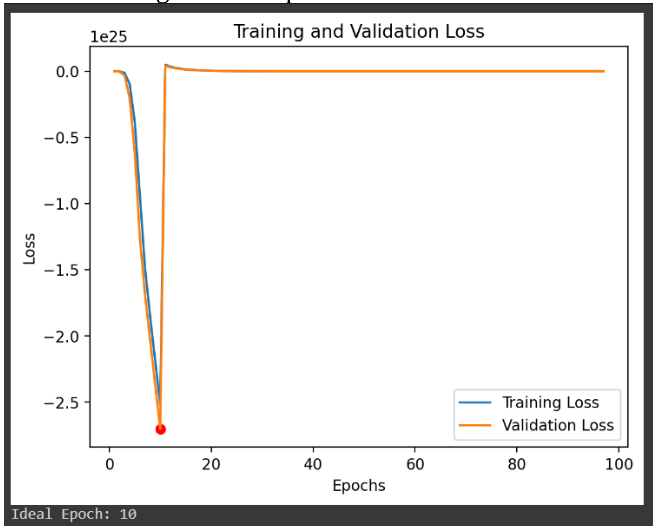
i. The number of Epochs

The ***hypothesis*** is that increasing the number of epochs used during training will improve model performance up to a point. Beyond that point, increasing the number of epochs may not improve the model's accuracy or convergence significantly. Training for too few epochs, on the other hand, may result in underfitting, whereas training for too many epochs may result in overfitting. We want to find the best balance between model performance and training time by adjusting the number of epochs.

A series of experiments were carried out in order to validate the hypothesis regarding the impact of the number of epochs on model performance. [5][5]

The experiments involved training the model with varying numbers of epochs, ranging from a small number to a large number. The validation loss metric was used to assess the model's performance. The findings supported the hypothesis, as increasing the number of epochs resulted in a significant reduction in validation loss at first, indicating improved performance. However, once a certain threshold was reached, the reduction in validation loss became marginal, indicating that the model had already converged.[5]

A Number of Epochs vs. Validation Loss graph was examined to determine the optimal number of epochs. This graph revealed a pattern in which the validation loss decreased rapidly at first but then began to level off after a certain number of epochs. The ideal epoch number was determined to be the point at which increasing the epochs did not result in significant improvements in validation loss.



```
Ideal Epoch: 10
```

ii. Learning rate

The ***hypothesis*** for the learning rate experiment was that changing the learning rate during training would affect the model's accuracy. Finding an optimal learning rate was expected to improve performance, whereas extremely high or low learning rates were expected to impede convergence and result in lower accuracy.[5]

To put the hypothesis to the test, the model was trained with various learning rates, including 0.1, 0.01, 0.001, 1e-3, and 5e-3. After each training run, the model's accuracy was assessed. Changing the learning rate had a noticeable effect on the model's accuracy, according to the results.

The model's accuracy decreased when the learning rate was set to 0.1 and 0.01, compared to the baseline. This is due to an excessively fast learning rate, which causes the model to overshoot optimal values and struggle to converge effectively.

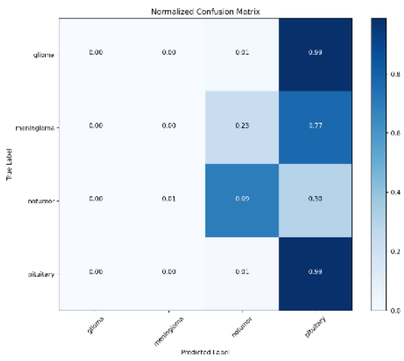


Figure 30 Reduced Accuracy due to change in LR

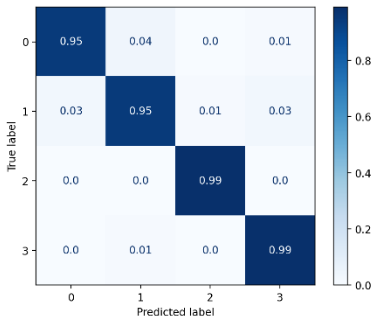
The model's accuracy improved when the learning rate was set to 0.001, 1e-3, and 5e-3, compared to the baseline. These learning rates enabled more precise updates to the model's parameters, resulting in improved convergence and accuracy. Among the learning rates tested, 1e-3 and 5e-3 produced the best results, indicating that they were closer to the optimal learning rate for the given model and dataset. This finding supports the hypothesis that determining an optimal learning rate can improve model accuracy. [5]

Finally, the experiments on changing the learning rate confirmed the hypothesis that changing the learning rate has a significant effect on the model's accuracy. It was possible to improve the model's performance and achieve higher accuracy by selecting an appropriate learning rate, such as 1e-3 or 5e-3.

iii. Optimization Algorithm

The ***hypothesis*** for the experiment with different optimisation algorithms was that using a different optimisation algorithm would affect the model's accuracy. It was expected that switching from the default optimisation algorithm(*Adam*) to *Stochastic Gradient Descent (SGD)* would improve accuracy.

The experiment consisted of training the model using the default optimisation algorithm before switching to SGD. After each training run, the accuracy was assessed. When SGD was used as the optimisation algorithm, the results showed a significant increase in accuracy. SGD, which is well-known for efficiently handling noisy and large-scale datasets, optimised the model's parameters by updating them based on randomly selected subsets of the training data. Because SGD is stochastic, it allows for faster convergence and better solutions, resulting in higher accuracy. The experiment confirmed that altering the optimisation algorithm, specifically, SGD had a significant impact on accuracy. The model achieved high accuracy by utilising SGD, outperforming the default optimisation algorithm.



iv. Weight Decay

The weight decay experiment's ***hypothesis*** was that using weight decay regularisation would improve the model's performance by reducing overfitting. The goal was to see how different weight decay values affected the model's accuracy.[5]

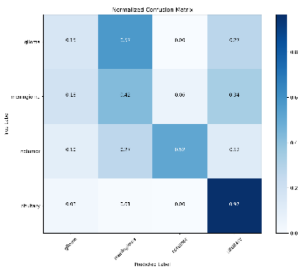


Figure 31 Reduced Accuracy due to wrong weight decay

During the experiment, the model was trained with weight decay values ranging from 0.001 to 0.01. After each training run, the accuracy was assessed. In contrast to the hypothesis, increasing the weight decay value resulted in a decrease in accuracy.
Weight decay regularisation adds a penalty term to the loss function, which encourages the model to use smaller weights. By reducing the model's complexity, this regularisation technique helps to prevent overfitting. However, in this particular experiment, increasing the weight decay value appeared to be beneficial.[5]

v. *Loss Function - Negative Log Likelihood (NLL) loss function*

The experiment's hypothesis was that using the Negative Log Likelihood (NLL) loss function instead of the CrossEntropyLoss loss function would improve the model's accuracy. The goal was to see if the loss function selection affected the model's performance.[5]
The model was trained using the NLL loss function during the experiment, and its accuracy and loss were assessed. Surprisingly, the accuracy decreased significantly, with an accuracy of 0.235 and a large negative value for the loss (-2.7e+25).
When the target variable has a categorical distribution, the NLL loss function is commonly used. It computes the negative log-likelihood of each class's predicted probabilities. However, in this experiment, the NLL loss function performed poorly when compared to the CrossEntropyLoss.

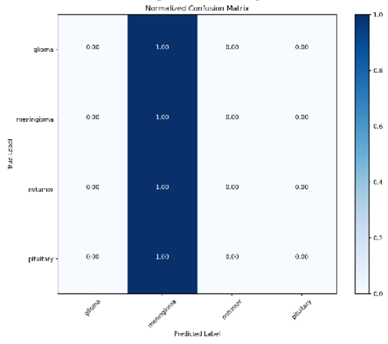


Figure 33 Wrong prediction due to wrong loss function

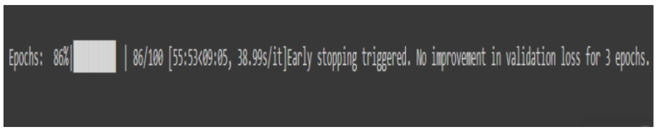
Finally, the experiment did not support the hypothesis that using the NLL loss function would improve the accuracy of the model. The findings suggest that the loss function should be carefully chosen based on the problem domain and model architecture.[5]

VI. Regularisation & Overfitting – An early stopping approach

Hypothesis: Early stopping is used to prevent overfitting and improve the model's generalisation capability. We can identify the point at which the model begins to overfit by monitoring the validation loss during training and stopping the training process at that point.

Early stopping is a regularisation technique used during neural network training. The goal is to keep the model from constantly improving on the training data while performing poorly on unobserved data. We can detect when the model's performance on the validation data begins to deteriorate by monitoring the validation loss, which is calculated on a separate validation set. This indicates that the model is overfitting and is failing to learn generalizable patterns.

Early stopping is implemented in the provided code snippet by keeping track of the best validation loss achieved thus far. The model's state is saved if the validation loss decreases. If the validation loss does not improve after a set number of epochs (determined by the patience parameter),



```
Epochs:  86%|████  | 86/100 [55:53<09:05, 38.99s/it]Early stopping triggered. No improvement in validation loss for 3 epochs.
```

Figure 32 Early Stopping

early stopping is triggered. This means that the training process is terminated early because the model's performance has not improved significantly.

Early stopping was triggered after 86 epochs due to no improvement in validation loss for three consecutive epochs. The training and validation losses over the last few epochs suggest that the model's performance has plateaued, and that additional training may not result in significant improvements.

This implies that the model has reached a point where it can no longer learn meaningful patterns from the data and is most likely overfitting. Early stopping prevents the model from wasting computational resources and time on additional training that does not improve performance.

VII. Global & Local Minima

Based on the values calculated for local and global minima, it appears that both are the same, which is 0.9187864065170288. This indicates that the optimisation process has reached a stable point in the parameter space and that the model's performance has plateaued. Because both the local and global minima have the same value, it indicates that the model has achieved a relatively high level of accuracy.
In terms of inference, it is possible to conclude that the model learned the patterns and features in the data effectively, resulting in consistent and accurate predictions. Minor changes in the optimisation process have no significant effect on the model's performance. This is a positive result, indicating that the model has been well-optimized.

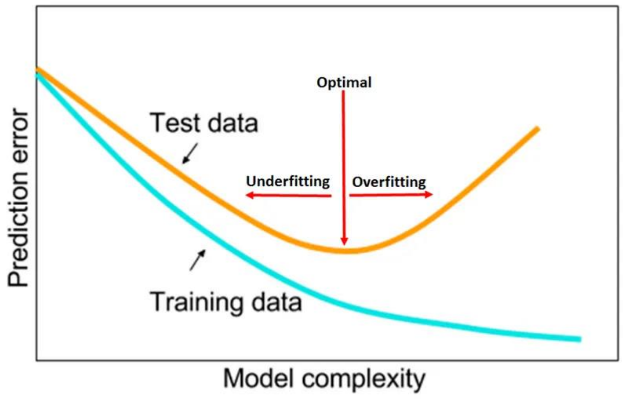


Figure 34 Global Minima

VIII. Conclusion

The experiment was designed to demonstrate the power of deep learning for precise brain tumour classification. The research looked into advanced architectures and hyperparameter optimisation to improve the performance of deep learning models in this context. The researcher used several models, including VGG, AlexNet, ResNet, and their own model, ResNeXt. ResNeXt is a ResNet architecture variant that incorporates the concept of "cardinality" through the use of grouped convolutions. This enabled a thorough examination of the effect of architectural complexity on classification accuracy. After analysing the results, it was discovered that deeper models, such as ResNet and ResNeXt, outperformed shallower architectures, such as VGG and AlexNet, in terms of accuracy. ResNet had the lowest accuracy loss of 0.1186, indicating superior generalisation and less overfitting when compared to the other models. VGG19, on the other hand, had the highest accuracy loss of 0.4623, indicating a relatively poorer performance in this regard. The effects of varying the number of epochs, learning rate, optimisation algorithm, weight decay, and loss function were also investigated. Among these parameters, the number of epochs and the optimisation algorithm produced promising results, while the others performed poorly.

Finally, the study demonstrated the utility of deep learning models, specifically ResNet and ResNeXt, for accurate brain tumour classification. The hyperparameter optimisation investigation shed light on the significant impact of architectural complexity and specific hyperparameters on classification accuracy. The findings could help to advance deep learning algorithms in medical image analysis and improve the accuracy of brain tumour classification systems.

Limitations & Future Work

The research had limitations due to memory constraints and the use of Google Colab, resulting in a reduced number of epochs during training. Future work should consider utilizing resources with larger memory capacities to improve convergence and performance. Additionally, exploring other medical imaging tasks, such as segmentation or detection, and investigating alternative hyperparameter optimization techniques would be beneficial. Furthermore, incorporating interpretability methods to enhance understanding and trust in the models' predictions is a valuable avenue for future research

IX. References

[1] aans.org."Brain Tumors". Accessed May. 10, 2023. [Online]. Available: https://www.aans.org/en/Patients/Neurosurgical-Conditions-and-Treatments/Brain-Tumors

[2] Saeedi, S., Rezayi, S., Keshavarz, H. et al. MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques. BMC Med Inform Decis Mak 23, 16 (2023). https://doi.org/10.1186/s12911-023-02114-6

[3] Raheleh Hashemzehi et al 2021 https://www.sciencedirect.com/science/article/abs/pii/S0208521620300784

[4] P Gokila Brindha et al 2021 . Detection of brain tumors from MRI images base on deep learning using hybrid model CNN and NADE, Biocybernetics and Biomedical Engineering.

[5] Towardsdatascience."Hyper-parameter Tuning Techniques in Deep Learning". Accessed May. 11, 2023. [Online]. Available: https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8

[6] Cem Dilmegani."Top Data Augmentation Techniques: Ultimate Guide for 2023".Accessed May. 11, 2023. [Online]. Available: https://research.aimultiple.com/data-augmentation-techniques/

[7] Musallam, A.S.; Sherif, A.S.; Hussein, M.K. A New Convolutional Neural Network Architecture for Automatic Detection of Brain Tumors in Magnetic Resonance Imaging Images. IEEE Access 2022, 10, 2775–2782

[8] Amin, J.; Sharif, M.; Haldorai, A.; Yasmin, M.; Nayak, R.S. Brain tumor detection and classification using machine learning: A comprehensive survey. Complex Intell. Syst. 2022, 8, 3161–3183

[9] Obeidavi, M.R.; Maghooli, K. Tumor Detection in Brain MRI using Residual Convolutional Neural Networks. In Proceedings of The 2022 IEEE International Conference on Machine Vision and Image Processing (MVIP), Ahvaz, Iran, 23–24 February 2022; pp. 1–5.

[10] Khalil, H.A.; Darwish, S.; Ibrahim, Y.M.; Hassan, O.F. 3D-MRI brain tumor detection model using modified version of level set segmentation based on dragonfly algorithm. Symmetry 2020, 12, 1256.

[11] Sajid, S.; Hussain, S.; Sarwar, A. Brain tumor detection and segmentation in MR images using deep learning. Arab. J. Sci. Eng. 2019, 44, 9249–9261.

[12] Karen Simonyan, Andrew Zisserman."Very Deep Convolutional Networks for Large-Scale Image Recognition ". Accessed May. 12, 2023. [Online]. Available: https://arxiv.org/abs/1409.1556

[13] Sovit Ranjan Rath."Implementing VGG11 from Scratch using PyTorch". Accessed May. 15, 2023. [Online]. Available: https://debuggercafe.com/implementing-vgg11-from-scratch-using-pytorch

[14] d2l, “Dive into Deep Learning,” 2023.

[15] Tavish Srivastava. "12 Important Model Evaluation Metrics for Machine Learning Everyone Should Know". Accessed May. 16, 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/s

[16] Brain Tumor MRI Dataset. https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/code

[17] AnalyticsVidhya “Introduction to The Architecture of Alexnet" ". Accessed May. 25, 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/

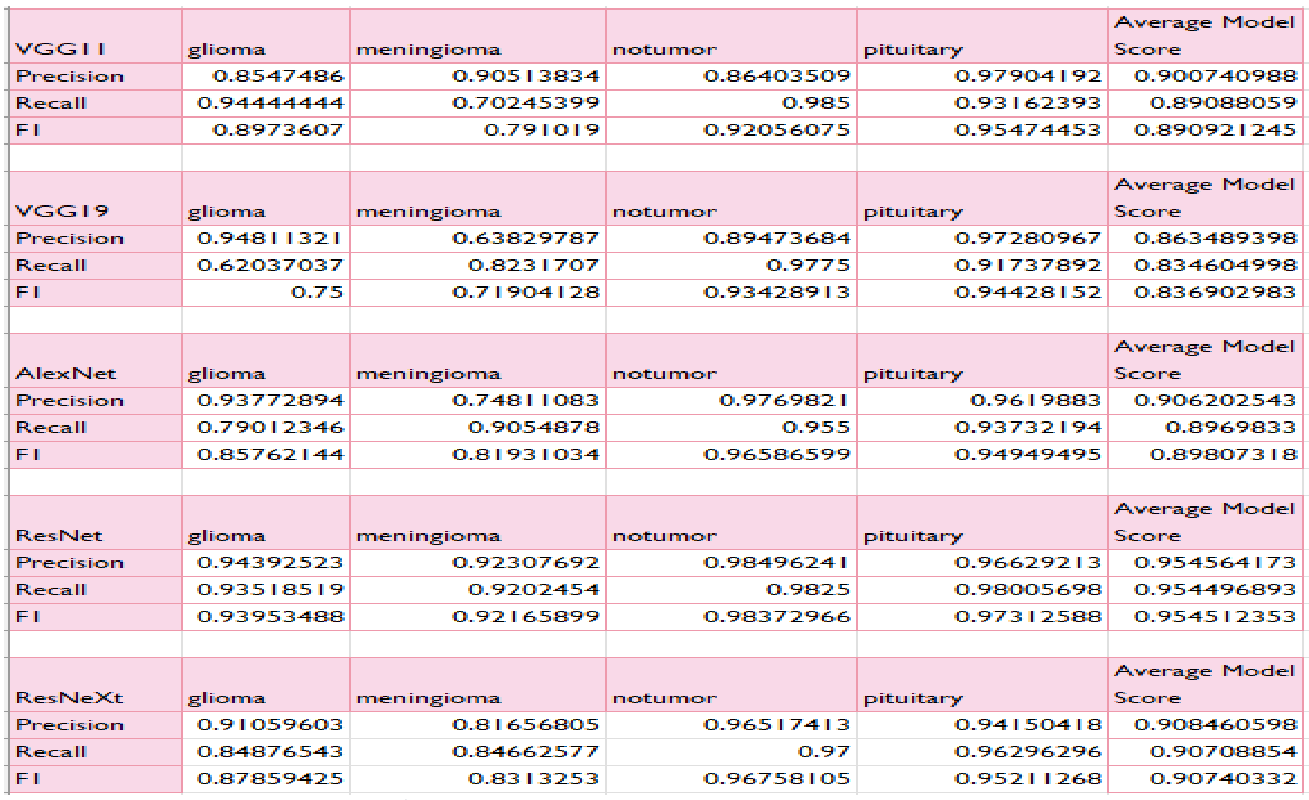
[18] Saining Xie et al, 2017, “ Aggregated Residual Transformations for Deep Neural Networks” Accessed May. 20, 2023. [Online]. Available: https://arxiv.org/abs/1611.05431v2

[19] Sik-Ho Tsang “Review: ResNeXt — 1st Runner Up in ILSVRC 2016 (Image Classification)”. Accessed May. 20, 2023. [Online]. Available: https://towardsdatascience.com/review-resnext-1st-runner-up-of-ilsvrc-2016-image-classification-15d7f17b42ac

X. Project Prototype

https://colab.research.google.com/drive/1nlJSi1_7uGuRCaLZXkjoShyxz4ri87mI?usp=sharing

XI. Appendix



VGG11	glioma	meningioma	notumor	pituitary	Average Model Score
Precision	0.8547486	0.90513834	0.86403509	0.97904192	0.900740988
Recall	0.94444444	0.70245399	0.985	0.93162393	0.89088059
F1	0.8973607	0.791019	0.92056075	0.95474453	0.890921245

VGG19	glioma	meningioma	notumor	pituitary	Average Model Score
Precision	0.94811321	0.63829787	0.89473684	0.97280967	0.863489398
Recall	0.62037037	0.8231707	0.9775	0.91737892	0.834604998
F1	0.75	0.71904128	0.93428913	0.94428152	0.836902983

AlexNet	glioma	meningioma	notumor	pituitary	Average Model Score
Precision	0.93772894	0.74811083	0.9769821	0.9619883	0.906202543
Recall	0.79012346	0.9054878	0.955	0.93732194	0.8969833
F1	0.85762144	0.81931034	0.96586599	0.94949495	0.89807318

ResNet	glioma	meningioma	notumor	pituitary	Average Model Score
Precision	0.94392523	0.92307692	0.98496241	0.96629213	0.954564173
Recall	0.93518519	0.9202454	0.9825	0.98005698	0.954496893
F1	0.93953488	0.92165899	0.98372966	0.97312588	0.954512353

ResNeXt	glioma	meningioma	notumor	pituitary	Average Model Score
Precision	0.91059603	0.81656805	0.96517413	0.94150418	0.908460598
Recall	0.84876543	0.84662577	0.97	0.96296296	0.90708854
F1	0.87859425	0.8313253	0.96758105	0.95211268	0.90740332

Figure 35 Individual class evaluations score of all models.