كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

جامعة حلوان
HELWAN UNIVERSITY

Graduation Project

[Medical Diagnosis System for Brain Cancer by Using Deep Learning Techniques]

Supervised by:
[Dr. Ahmed Hesham Mostafa]

January 2024

HELWAN UNIVERSITY
Faculty of Computers and Artificial Intelligence
Computer Science Department

[Medical Diagnosis System for Brain Cancer by Using Deep Learning Techniques]

A graduation project dissertation by:

[Muhammed Abdulrahim Ibrahim Muhammed (201900698)]

[Mostafa Essam Abdulfattah Abu-Shama (201900824)]

[Ahmed Ashraf Abdelmonem Mahmoud (201900016)]

[Omar Mohamed Kamel Abdelmalek (201900530)]

[Ashraqat Mohamed Abu Al-Ela Mohamed (201900156)]

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computers & Artificial Intelligence, at the Computer Science Department, the Faculty of Computers & Artificial Intelligence, Helwan University

Supervised by:

[Dr. Ahmed Hesham Mostafa]

January 2024

جامعة حلوان
كلية الحاسبات والذكاء الإصطناعي
قسم علوم الحاسب

[نظام التشخيص الطبي لسرطان الدماغ باستخدام تقنيات التعلم العميق]

رسالة مشروع تخرج مقدمة من:

[محمد عبد الرحيم ابراهيم محمد (201900698)]

[مصطفى عصام عبد الفتاح ابو شامه (201900824)]

[احمد اشرف عبدالمنعم محمود (201900016)]

[عمر محمد كامل عبدالمالك (201900530)]

[اشرقت محمد ابوالعلا محمد (201900156)]

رسالة مقدمة ضمن متطلبات الحصول على درجة البكالوريوس في الحاسبات والذكاء الإصطناعي، بقسم علوم الحاسب، كلية الحاسبات والذكاء الإصطناعي، جامعة حلوان

تحت إشراف:

[د. أحمد هشام مصطفى]

يناير 2024

Contents

Chapter 3: CNN & Transfer Learning

Chapter 4: Models Results and Graphs

Chapter 5: Mobile Application, Project Planning and Analysis

Acknowledgment:

In the first place, we would like to take this opportunity to express our appreciation to our supervisor **Dr. Ahmed Hesham** for his support, guidance, and encouragement throughout our graduation project.

Here we are writing the last pages of our 4 years story, the journey was not easy, and we passed by many challenges, but what made all of this much easier is our families support and trust, we reached this point and was able to finish our essential 16 years of learning because of them, and we want to thank them from the bottom of our hearts.

Finally, we would also like to extend our thanks to our faculty for providing a suitable environment that led us to represent the best image that Computer Science and Artificial Intelligence graduates of Helwan University are meant to represent.

Abstract:

Brain tumors, as complex and diverse as they are, present a formidable challenge in the field of medicine. These abnormal growths within the brain can manifest in various forms, locations, and severities, making their early detection and diagnosis a critical factor in successful treatment. The rapid advancements in artificial intelligence (AI) have opened exciting possibilities for transforming the way we approach brain tumor detection, offering valuable assistance to healthcare professionals in their quest to provide timely and effective care to patients.

Traditionally, the identification of brain tumors has heavily relied on the expertise of radiologists and neurologists who manually analyze complex medical imaging data, including magnetic resonance imaging (MRI) and computed tomography (CT) scans. However, this process can be time-consuming and subject to human error, potentially leading to delayed diagnoses or misinterpretations.

One of the most significant advantages of AI in brain tumor detection is its ability to enhance the speed and precision of diagnosis. With AI assistance, doctors can access real-time insights, potentially leading to earlier detection and intervention. Furthermore, AI algorithms can assist in the triage process, prioritizing cases that require urgent attention and optimizing the allocation of resources within healthcare facilities.

AI's role in brain tumor detection is not limited to just analyzing imaging data. It can also aid in treatment planning by providing valuable information about tumor characteristics, assisting in surgical navigation, and even predicting patient outcomes. This holistic approach to brain tumor management ensures that healthcare professionals have the support and tools they need to deliver the best possible care to their patients.

In this ever-evolving landscape, the synergy between AI and medical expertise is poised to revolutionize the way we detect and treat brain tumors. By harnessing the power of artificial intelligence, healthcare professionals can leverage cutting-edge technology to enhance their diagnostic capabilities, ultimately improving patient outcomes and advancing our understanding of these complex conditions.

Keywords: Convolutional neural network (CNN), deep learning, transfer learning, fine tuning.

Chapter 1: Introduction

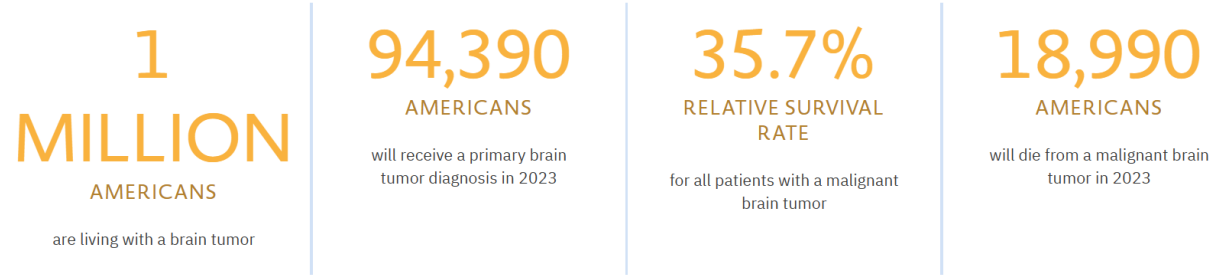
1.1 Overview

In the medical field of image classification, particularly in the context of brain tumors, has garnered significant attention in recent years. Brain Cancer is the most daisies that no cure for it and it's hard to detect, So we decided to make an AI Model to help the doctors around the world to be able to know if the patient infected or not, so we think implementing an AI Model help to detect the brain tumor within an Image of his MRI brain that helps to detect and classify the tumor quickly, saving time and effort.

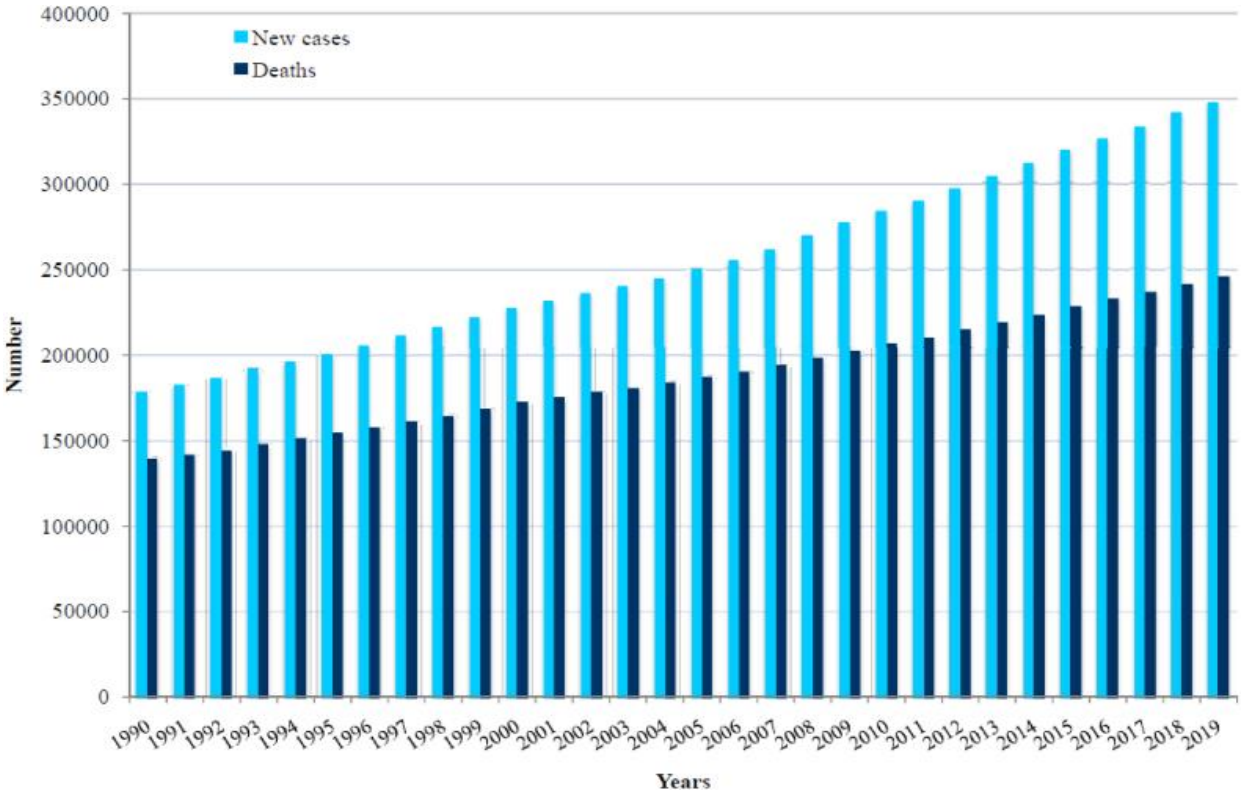
Problem:

There are more than 100 distinct types of primary brain tumors, each with its own spectrum of presentations, treatments, and outcomes. More than any other cancer, brain tumors can have lasting and life-altering physical, cognitive, and psychological impacts on a patient's life. And, despite years of research, brain cancer survival rates have remained little changed in recent years, even while survival rates for many other cancers have been significantly improved.

The following statistics and facts provide a snapshot of the burden primary brain tumors cause to Americans of all walks of life. Additional references are noted at the bottom of the page, but for a majority of the information listed below the figures stem from population statistics sampled during the 2015-2019 time period and analyzed in 2022 and reflect the latest available data at that time.



Brain cancer is a serious issue in the global burden of diseases. This observational research aimed to assess trends of the brain cancer incidence and mortality in the world in the period 1990–2019.



Worldwide, 347,992 new cases of brain cancer were recorded in 2019: it was diagnosed in 187,491 (54%) males and 160,501 (46%) females. In 2019, the total number of deaths from brain cancer worldwide was 246,253 (138,605 males and 107,648 females).

1.2 Objectives

- Using deep learning algorithms to implement a model helps to detect and classify Brain Cancer.
- Develop a CNN model for accurate brain tumor for classification MRI images.
- Evaluate the model's performance on a dataset comprising glioma, meningioma, and pituitary tumors.
- Implement effective image preprocessing techniques to enhance the quality of input data.
- Design and implement a Flutter mobile application for users and link it with the deep learning model.
- Create a mobile application containing different users such as (admin, doctor, and patients), each of whom has specific functions within the system.
- Create a backend system that handles Firebase, Flutter UI, API, and deep learning model together.
- Create an API controller that facilitates seamless communication among different components of the system, ensuring efficient data flow and interaction.
- Get more information that helps to detect and classify the tumor.
- Give the users all advice that helps them.
- All the treatments and the knowledge used are by the help of a brain and neurology doctor.

1.3 Purpose

Make it easy for any brain tumor patient or doctor to use our project to detect if the patient has the tumor or not, without missing a lot of time or effort and that will solve the problem of overcrowding in brain and neurology clinics and saves a lot of time spent on waiting.

The final goal is to assist in early detection and classification of brain tumors, potentially leading to timely medical interventions without human error.

1.4 Scope

- The project aims to help Brain Tumor Patients and doctors.
- Early Detection of Tumor Disease without human error.
- The doctor can use the Model from home or which place without being lost in time.
- Comparison of the proposed CNN model's performance with existing methods, showcasing its effectiveness in achieving high accuracy and precision.
- Generate the best accuracy of the deep learning model.
- Provide a brief comparison of at least two algorithms.
- Real-time monitoring and analysis by model for MRI images.
- Efficient communication and data flow between different components of the system through an API controller.

1.5 Model Planning

In this section, we delve into the project planning process, which encompasses the essential steps and considerations for successful project execution. We define clear project objectives, establish realistic timelines and milestones, allocate necessary resources, and identify key deliverables. Despite the absence of financial costs, the project demands a high level of effort, focus, and speed to achieve the desired outcomes. We emphasize the importance of effective project planning in ensuring smooth project implementation.

1.5.1 Feasibility Study

One of the most critical problems facing the world population, especially children, the elderly, and people with chronic diseases must be taken care of too much. That is because the Brain Tumor is hard to detect in early-stage people so, this model support doctors and whether the patient is infected with tumor or not by showing the patient's MRI.

Cost: The model has no cost because it is completely free. Without paying any cost. **Accessibility:** Doctors will be able to access it anytime and there will be no time constraints.

Time: Save users time instead of spending a lot of time at the Doctor's clinic. The user can use it from home or which place without wasting time and the user can chat with doctor by our mobile app.

1.5.2 Estimated Cost

Although the project does not incur direct financial costs, we recognize the investment of effort, time, and resources required for its execution. In this section, we provide a detailed estimation of the required resources, including personnel, equipment, and data collection efforts. This estimation helps in planning and allocating resources effectively throughout the project.

Hardware Materials:
Personal laptop or personal computer with medium specifications connected to the internet.

Software Materials (Technologies and Tools):

- Python Language
- Jupyter Notebook
- Colab Notebook
- Flask Framework
- Dart Language
- Flutter Framework
- Firebase & Database
- Android Studio & VS Code

1.6 Analysis and Limitation

Difficulty:
Complex Anatomy: The brain is a highly intricate organ with various regions responsible for diverse functions. Tumors can develop in any part of the brain, making their detection and localization complex.

Subtle Symptoms: In the early stages, brain tumor symptoms can be subtle and easily attributed to other common conditions, such as headaches or fatigue. This can lead to delayed diagnosis.

Symptom Variability: Brain tumor symptoms can vary widely depending on the tumor's size, location, and type. Some patients may experience seizures, while others may have cognitive changes or motor skill issues.

Time:
If any doctor wants to use this model to check brain cancer MRI of patients, whether the patients have cancer or not, it will take few seconds but with limited number of users.

1.7 Need for the Models

The Essential Role of Models in Brain Tumor Classification: A Digital Gateway via Medical Imaging.

In the quest to diagnose and manage brain tumors effectively, the need for specialized models in conjunction with medical imaging techniques is paramount. These models serve as digital gateways, bridging the gap between raw medical imaging data, such as MRIs scans, and precise diagnosis. Here's why these models are indispensable.

The traditional methods of brain tumor classification based on manual inspection of MRI images are time-consuming and subject to human error, so we need models.

1.8 Analysis of the Model

In analyzing the existing systems related to brain cancer, we identify their limitations and drawbacks. By examining current technologies, methodologies, and approaches, we gain insights into the gaps and deficiencies that necessitate the development of a new system. This analysis provides a foundation for designing a solution that addresses the identified limitations and enhances the overall effectiveness of medical field for brain cancer classification.

1.8.1 User Requirements

Understanding the needs and expectations of the system's users is vital for designing a user-centric solution. In this section, we identify and analyze the specific requirements and preferences of different user groups, such as admin, doctor, and users (patients).

Users want to know whether he is infected by Tumor or not, and that is by uploading an MRI on the patient's own as a basic requirement.

1.8.2 System Requirements

Building upon the user requirements, we define the system requirements in detail. This includes capturing domain-specific requirements, such as accurate deep learning and integration with existing medical systems. Additionally, we outline the functional requirements that specify the desired features and functionalities of the system, such as real-time, data storage, report generation, integration with APIs. Furthermore, we consider the non-functional requirements, including usability, robustness, reliability, performance, flexibility, efficiency, and availability, to ensure the system's overall effectiveness and user satisfaction.

What does your computer need in software and hardware?

Of course, the operating system, and since our application code will be written in many languages, frameworks, and tools. As we referred to them previously.

Chapter 2: Related Work and Literature Review

2.1 Introduction

Brain tumors afflict thousands of individuals worldwide, causing significant morbidity and, in many cases, fatalities. The challenges of efficient diagnosis and the shortage of data are major hurdles in the effective management of this condition. Current diagnostic methods for brain tumors often entail invasive procedures or time-consuming analyses, leading to delayed detection and potentially unfavorable patient outcomes.

One prevalent technique for brain tumor detection involves the use of medical imaging such as magnetic resonance imaging (MRI) and computed tomography (CT) scans. However, obtaining accurate results can be hindered by the complexities of brain anatomy and the subtleties of tumor appearance in these images. Consequently, many brain tumor cases remain undetected for extended periods, exacerbating the disease's progression and complications.

Recent advancements in artificial intelligence (AI) and deep learning have provided a ray of hope in addressing these challenges. Deep learning algorithms, trained on extensive datasets of brain scans, are gaining popularity for their ability to detect brain tumors rapidly and accurately in medical images. These AI models can analyze intricate brain structures, identify anomalies, and distinguish between benign and malignant tumors with remarkable precision.

Nonetheless, the availability of comprehensive brain tumor datasets remains a limiting factor in training robust deep learning models. Small datasets may lead to overfitting, compromising the model's generalizability. In response, researchers are exploring transfer learning as a strategy to address data scarcity. Transfer learning leverages knowledge gained from source tasks with abundant data to enhance the performance of models on the target task—diagnosing brain tumors based on MRI images.

This approach typically involves pretraining deep learning models on diverse, data-rich tasks and then fine-tuning them on the relatively small dataset specific to brain tumor diagnosis. By doing so, researchers aim to bridge the gap between the data-deficient target task and the wealth of information available in source tasks.

2.2 Related Works

Paper 1: Brain Tumor Classification Using Convolutional Neural Network

1. Introduction

The introduction to the paper sets the stage by underlining the critical nature of brain tumor diagnosis within the realm of medical science. It begins by citing statistics from the American Cancer Society, projecting the alarming number of brain tumor diagnoses and potential fatalities in the United States each year. Survival rates are highlighted, emphasizing their variance according to tumor type and patient age.

The central focus is on the menacing aspect of brain tumors, their distinction into primary and secondary forms, and the profound impact on the brain's normal functioning. The section underscores the gravity of the issue, introducing the paper's core objective: the development of a Convolutional Neural Network (CNN) model for precise brain tumor classification.

Medical imaging techniques, such as magnetic resonance imaging (MRI), play a pivotal role in detecting brain tumors. Nevertheless, the intricate intricacies of brain anatomy and the subtle characteristics of these tumors can pose a considerable challenge for human interpretation. Even with advanced imaging technology, identifying small or early-stage brain tumors can be elusive to the human eye. This is where the integration of machine learning and artificial intelligence (AI) becomes crucial.

Deep learning algorithms, particularly Convolutional Neural Networks (CNNs), exhibit remarkable capabilities in processing intricate medical images, uncovering complex patterns within high-dimensional data, and providing invaluable assistance in diagnosing and planning treatment for brain tumors.

- We propose a Deep Learning techniques especially CNN approach that can effectively predict Brain Cancer MRI.

2. Materials and Methods

A. Dataset Collection and Description:

This subsection delves into the specifics of the dataset used for the study. Comprising 3064 T1-weighted contrast-enhanced images from 233 patients, the dataset encompasses three distinct types of brain tumors: glioma, meningioma, and pituitary tumor. Each image file is rich in information, including labels, patient IDs, image data, tumor borders, and tumor masks.

B. Preprocessing:

The preprocessing methodology is meticulously detailed, encompassing steps such as resizing images to (112 x 112) dimensions, application of a Gaussian filter with a (5 x 5) kernel, and the crucial application of histogram equalization. The division of the dataset into training, validation, and test sets is justified, specifying the proportions allocated to each.

3. Patients and Data Acquisition

This section serves as a supplementary detailing of the dataset, emphasizing its relevance and contribution to the study. Patient demographics, the number of images per tumor type, and the dataset's origin are expounded upon.

The brain tumor dataset retrieved from comprises 3064 T1-weighted contrast-enhanced images. The dataset is collected from 233 patients with three kinds of brain tumor: meningioma (708 images), glioma (1426 images), and pituitary tumor (930 images).

4. Deep Convolutional Neural Networks

A. Proposed Methodology:

The proposed CNN model's architecture is elucidated in detail. Commencing with the preprocessing steps, the section progresses to the intricate layers of the CNN, explicitly detailing the convolutional layers (C1, C2, C3), subsampling layers (S1, S2), and fully connected layers. The utilization of the Rectified Linear Unit (ReLU) activation function and the incorporation of dropout layers to mitigate overfitting are meticulously explained. Finally, another dense layer with (SoftMax) activation and 3 nodes for the classification.

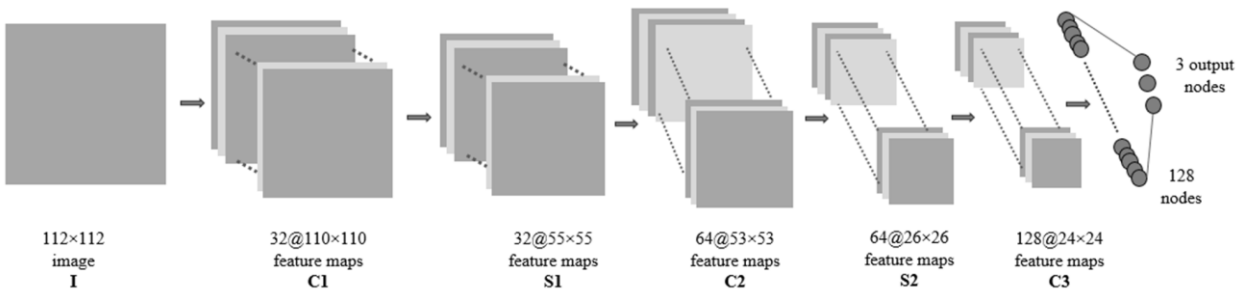
TRAINABLE PARAMETERS FOR CNN MODEL

Layers	Parameters
Conv2D (C1)	320
Conv2D (C2)	18,496
Conv2D (C2)	73,856
Dense (1)	9,437,312
Dense (2)	387
Total	9,530,371

B. CNN Architecture:

The CNN architecture is presented with granularity, specifying the dimensions of convolution kernels, max-pooling windows, and dropout rates. The cumulative number of trainable parameters in each layer is provided, offering insight into the model's complexity.

The model updates its parameter during the training with forward propagation and backward propagation algorithm.



5. Evaluation Metrics

The evaluation metrics employed for assessing the CNN model's performance are exhaustively detailed. A comprehensive analysis includes the confusion matrix, precision, recall, f1-score, support, and Receiver Operating Characteristic (ROC) curves. The achieved testing accuracy of 94.39% and average precision of 93.33% are highlighted as indicative of the model's effectiveness.

$$Precision = \frac{True\ Positive}{True\ positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ positive + False\ Negative}$$

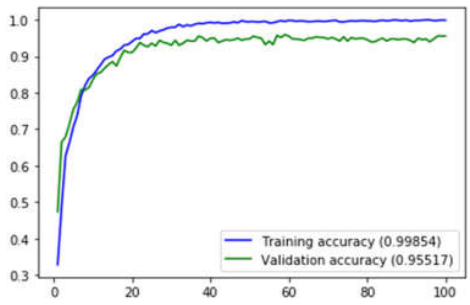
$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

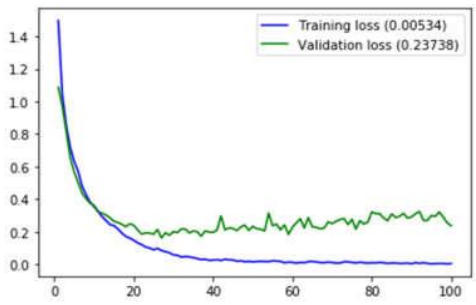
PERFORMANCE MEASURE INDICES

Class	Precision	Recall	F1-Score	Support
Class 1 (Glioma)	0.88	0.85	0.87	103
Class 2 (Meningioma)	0.94	0.95	0.94	226
Class 3 (Pituitary)	0.98	0.99	0.99	171

6. Results

The analysis section provides a visual representation of the training and validation sets' loss and accuracy across 100 epochs. The ROC curve is presented, illustrating the model's ability to discern between different tumor classes. The confusion matrix and performance measures per class offer a nuanced understanding of the model's strengths and areas of proficiency.





7. Conclusions

Brain tumor classification is very crucial in the domain of medical science. In this paper, we concentrated on developing a CNN classifier which classifies among three important tumor classes (glioma, meningioma, pituitary). Initially, the proposed system preprocesses the image data. The preprocessing includes filtering images using Gaussian filter and applying histogram equalization technique on the filtered images. Then the system classifies the images using the CNN model. The number of parameters of the model is too high, and the model is trained on a significantly small amount of data. Hence, there is a possibility of overfitting. To prevent the overfitting, a regularization technique, i.e. dropout regularization, is used on the model. It helps the model to concentrate on the most prominent patterns during the training phase. Therefore, there is a better chance of generalization which keeps the model stable. The model ends up with the accuracy of 94.39% and an average precision of 93.33%. Hence, the CNN classifier will be very significant in the medical field and in saving precious lives.

Paper 2: Transfer Learning Networks with Skip Connections for Classification of Brain Tumors

1. Abstract

This article presents a transfer learning model via convolutional neural networks (CNNs) with skip connection topology, to avoid the vanishing gradient and time complexity, which are usually common in transfer learning networks. Three pretrained CNN architectures, namely Alex-Net, VGG16 and Google-Net are employed to equip with skip connections. The transfer learning is implemented through fine-tuning and freezing the CNN architectures with skip connections based on magnetic resonance imaging (MRI) slices of brain tumor dataset. Furthermore, in the preprocessing, a frequency-domain information enhancement technique is employed for better image clarity. Performance evaluation is conducted on the transfer learning networks with skip connections to obtain improved accuracy in brain MRI classifications.

2. Introduction

The study delves into the realm of brain tumor classification using transfer learning networks enriched with skip connections. Brain tumors, including glioma, meningioma, and pituitary tumors, emerge from uncontrolled cell division and abnormal tissue growth, posing serious health risks. Timely detection is crucial, as these tumors exhibit variations in size, shape, and contrast.

Deep learning, specifically convolutional neural networks (CNN) like Alex-Net, VGG16, and VGG19, has become a staple in brain image analysis for tumor classification. Despite their efficiency, traditional DL models face challenges such as vanishing gradient and time complexity.

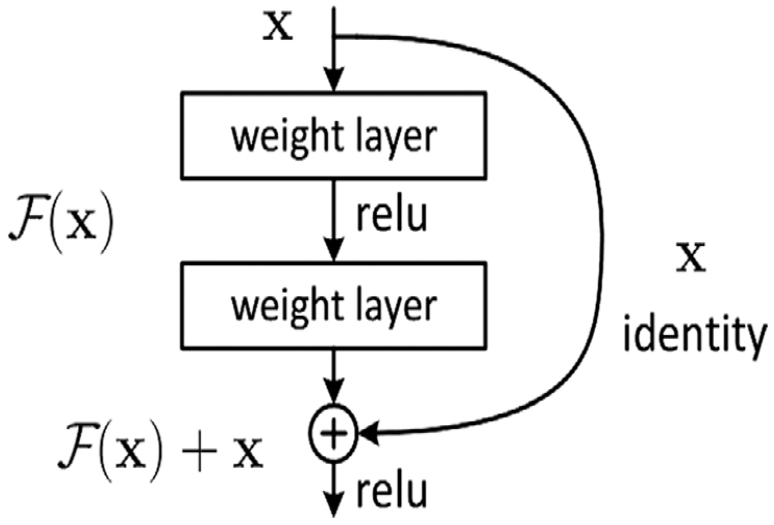
To overcome these hurdles, the study introduces a transfer learning (TL) model employing CNN with skip connection topology. Three pretrained CNN architectures Google-Net, Alex-Net, and VGG16 are harnessed, incorporating skip connections to alleviate issues like vanishing gradient and time

complexity. Adjustments are made to input datasets and the number of kernels for output networks.

The preprocessing stage includes the application of a frequency-domain information enhancement technique to enhance image clarity. TL is executed through fine-tuning and freezing CNN architectures with skip connections, utilizing magnetic resonance imaging (MRI) slices from a brain tumor dataset on Fig share. Performance evaluation focuses on common tumor classifications, such as meningioma, glioma, and pituitary tumors, showcasing the effectiveness of the proposed approach in improving accuracy and efficiency in brain MRI classifications.

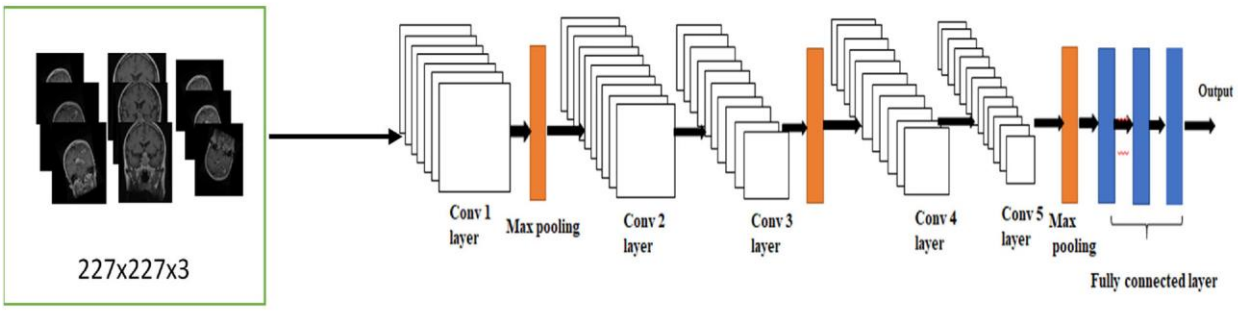
3. TL Networks with Skip Connections

The idea behind skip connections is to allow the gradient to flow more easily through the network during training. By adding the input of a layer to the output, the model has a shortcut or "skip" that enables the gradient to bypass certain layers. This helps in mitigating the vanishing gradient problem and allows for more effective training of very deep networks.

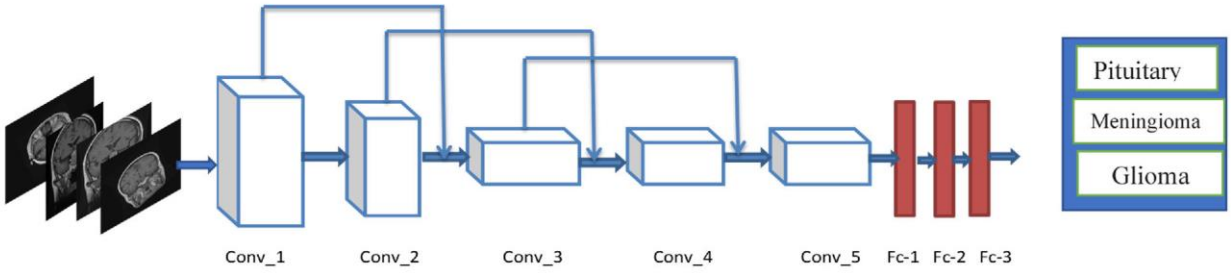


3.1 Alex-Net Model

Alex-Net, the 2012 ISLVRC winner, is a pioneering deep neural network with five convolutional and three fully connected layers. In this study, a modified Alex-Net with skip connections is utilized for brain tumor detection, adapting the architecture for three classes. The preprocessing involves Z-score normalization and contrast enhancement using adaptive histogram equalization. The modified architecture differs in skip connections, dataset classes, frequency domain changes, and kernel variations. Transfer learning is applied by fine-tuning the pretrained Alex-Net, specifically setting the final fully connected layer to accommodate three brain tumor classes.



(A) AlexNet architecture



(B) AlexNet with skip connection

TABLE 1 Summary of convolutional layers in AlexNet

Convolutional layer number	Number of filters	Filter size	Stride	Padding
Convolutional layer-1	96	3 × 3	4	0
Convolutional layer-2	256	3 × 3	1	2
Convolutional layer-3	384	3 × 3	1	1
Convolutional layer-4	384	3 × 3	1	1
Convolutional layer-5	256	3 × 3	1	1

Layer type	Feature map	Kernel size + stride	Activation
Input image	1	—	—
Convolution	96	3 × 3 _ stride 4	relu
Max pooling	60	3 × 3 _ stride 2	relu
Convolution	256	3 × 3 _ stride 1	relu
Max pooling	256	3 × 3 _ stride 2	relu
Convolution	384	3 × 3_ stride 1	relu
Convolution	256	3 × 3 _ stride 1	relu
Max pooling	256	3 × 3 _ stride 2	relu
FC	—	—	relu
Softmax	—	—	Softmax

TABLE 2 Summary of layers in AlexNet

Evaluation Metrics and Results:

The performance of an Alex-Net with skip connections by examining individual alternate layers. Incremental convolutional layers result in improved accuracy, with conv_5 showing the highest performance. Dropout layers eliminate overlapping feature maps, enhancing overall performance. Intermediate layers (conv_1, conv_3, conv_5, and two fully connected layers) are separately analyzed for brain tumor classification, utilizing features learned from these layers with a support vector machine classifier. Increasing the number of convolutions positively impacts the fully connected layer FC_6, yielding high performance metrics. The study observes that extracting features from conv_3, representing mid-level features, contributes to improved performance. The training progress, illustrated in Figure 11, demonstrates the model achieving 100% accuracy after 200 epochs with a reduced loss of 0.003.

$$Acc = \frac{True\ positive\ (TP) + True\ negative(TN)}{Total\ number\ of\ tested\ images}$$

$$sensitivity = \frac{TP}{TP + False\ negaitve\ (FN)}$$

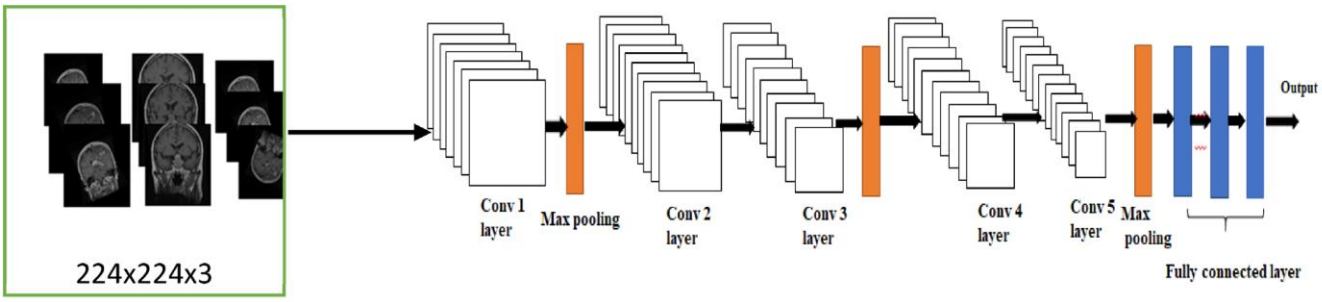
$$specificity = \frac{TN}{TN + False\ positive\ (FP)}$$

TABLE 5 Performance of AlexNet with skip connection

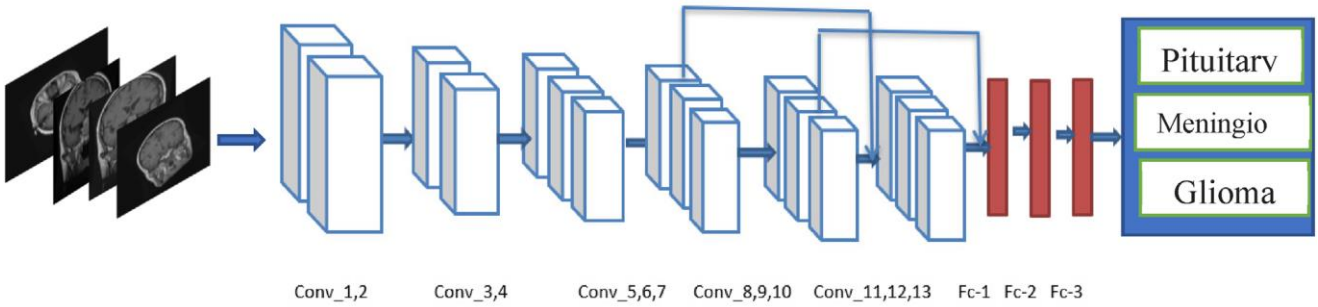
Layers	Accuracy (%)	Sensitivity (%)	Specificity (%)	F-score (%)	Precision (%)
Conv_1	50	55	78	59	56
Conv_3	63.5	70.2	84	72.5	72
Conv_5	95	94.2	98	93	94
FC_6	96.1	95.2	98.2	94.5	95
FC_7	92	91	96	92.2	91

3.2 VGG16 Model

The VGG16 architecture, comprising 16 layers, employs a structure similar to Alex-Net with convolutional, max-pooling, and ReLU layers. The number of filters ranges from 64 to 512, with three fully connected layers. Filter window size is consistently 3 × 3 across all convolutional layers. VGG16 has higher layer parameters compared to Alex-Net, with varying filter sizes in different layers. The computational complexity is elevated due to two stages with 512 filters each. For transfer learning, the final fully connected layer is fine-tuned for three output classes. The rectified linear unit is used as the activation function. VGG16 with skip connection is illustrated in Figure 5, maintaining an input dataset size of 224 × 224. Table 3 details the layer count in the VGG16 network. The architecture's pyramid structure aids in feature extraction and enhances classification accuracy.



(A) VGG16 architecture



(B) VGG16 with skip connection

TABLE 3 Summary of layers in VGG16

Layer type	Feature map	Kernel size + stride	Activation
Input image	1	—	—
Convolution	64	3 × 3 _ stride 4	relu
Convolution	64	3 × 3 _ stride 4	relu
Max pooling	64	3 × 3 _ stride 2	relu
Convolution	128	3 × 3 _ stride 4	relu
Convolution	128	3 × 3 _ stride 4	relu
Max pooling	256	3 × 3 _ stride 2	relu
Convolution	256	3 × 3_ stride 1	relu
Convolution	256	3 × 3 _ stride 1	relu
Convolution	256	3 × 3 _ stride 1	relu
Max pooling	256	3 × 3 _ stride 2	relu
Convolution	512	3 × 3 _ stride 1	relu
Convolution	512	3 × 3 _ stride 1	relu
Convolution	512	3 × 3 _ stride 2	relu
Max pooling	512	3 × 3 _ stride 1	relu
Convolution	512	3 × 3 _ stride 1	relu
Convolution	512	3 × 3 _ stride 2	relu
Convolution	512		
Max pooling	256	3 × 3 _ stride 2	relu
FC	—	—	relu
Softmax	—	—	Softmax

Evaluation Metrics and Results:

The analysis of the VGG16 architecture reveals a 32.5% accuracy difference between the first and last convolutional layers, indicating that initial layers learn from visible image variations, while later layers capture both visible and invisible variations, yielding higher accuracy. Freezing different layers in VGG16 shows that the final convolutional layer (conv_13) provides the best performance. Final fully connected layers outperform other layers, achieving 97.2% accuracy. Training progress in Figure 14 demonstrates 100% accuracy after 200 epochs with a loss reduction to 0.003. Confusion matrix computations result in a true positive rate of 33.33%.

TABLE 6 Performance of VGG16 with skip connection

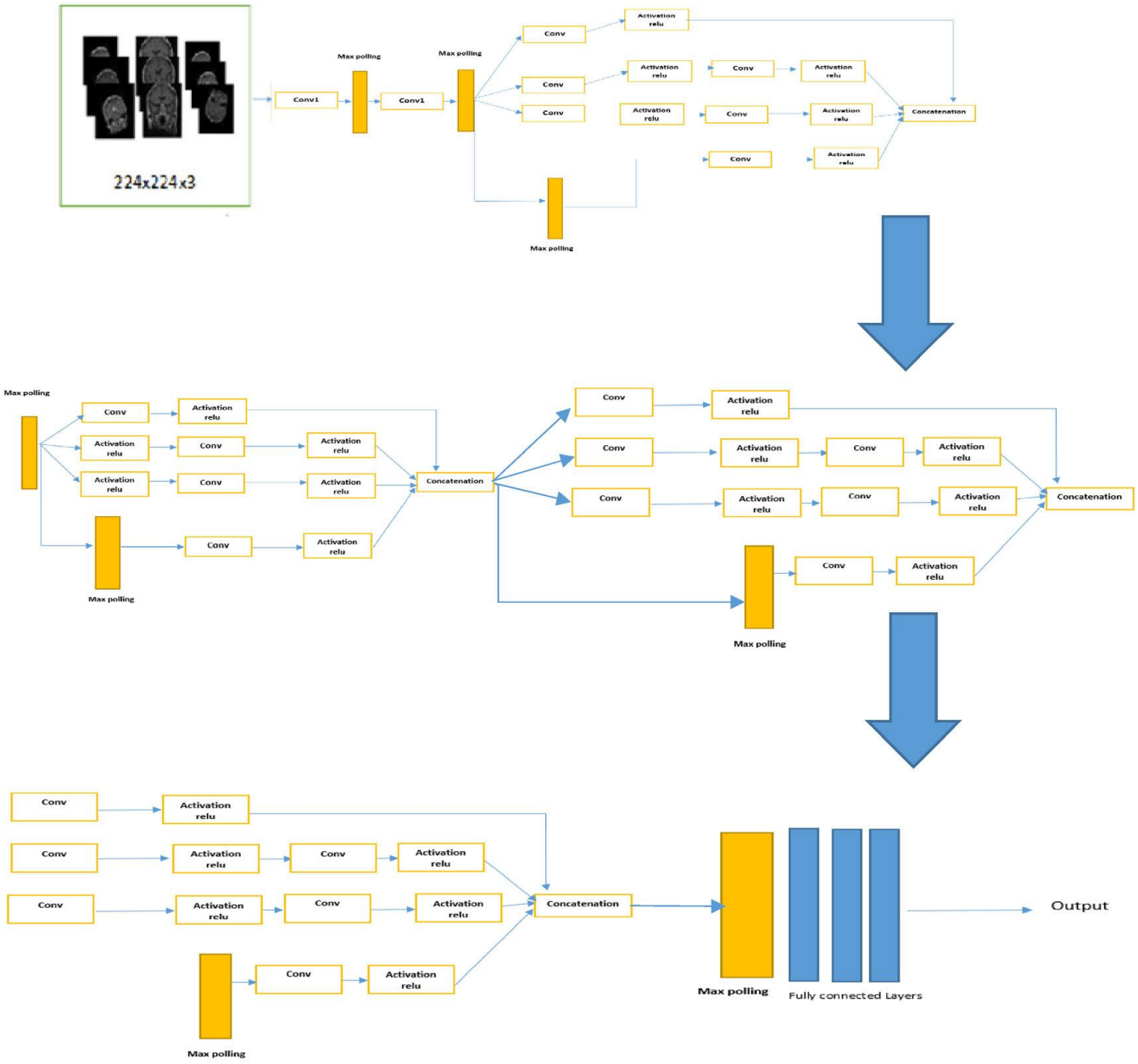
Layers	Accuracy (%)	Sensitivity (%)	Specificity (%)	F-score (%)	Precision (%)
Conv_1	50	55	78	59	56
Conv_3	63.5	70.2	84	72.5	72
Conv_5	65.3	73.2	85	73.5	74.5
Conv_7	68.5	75	88	76.7	77.2
Conv_9	72.5	78.5	88.4	79.2	81.2
Conv_11	77.3	82	91.5	82.2	84
Conv_13	82.5	88.2	94	89.5	91
FC_1	96.5	96.2	95.2	92.5	92.5
FC_2	98.9	96.9	97	94.2	94.8

3.3 Google-Net Model

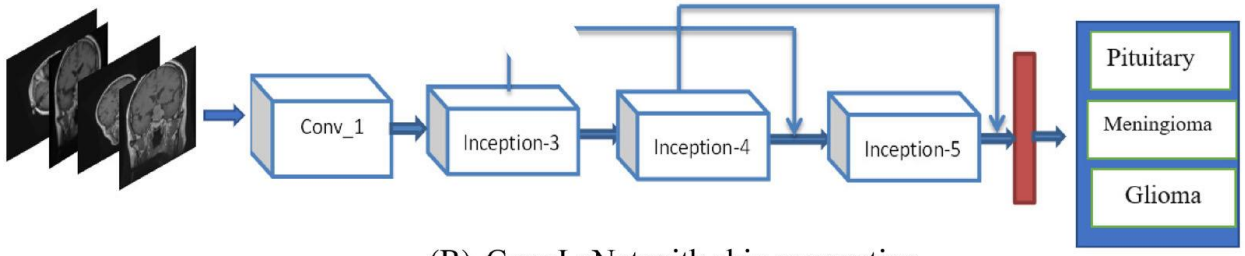
Google-Net with skip connection is a network architecture that addresses the issue of vanishing gradients by adding previous layer connections to the current layers. This ensures that there is no feature loss, and the present layer's features are enhanced by adding previous layer features. Google-Net consists of nine inception modules, each containing one max-pooling layer and six convolutional layers, with four layers dedicated to dimension reduction. Relu activation is applied to all fully connected layers, and dropout regularization is used. With 6.8 million parameters, Google-Net outperforms VGG and Alex-Net, and it surpasses Alex-Net on the ILSVRC dataset.

TABLE 4 Summary of layers in GoogLeNet

Layer type	Feature map	Kernel size + stride	Activation
Input image	1	—	—
Convolution	64	3 × 3 _ stride 4	relu
Max pooling	64	3 × 3 _ stride 4	relu
Convolution	60	3 × 3 _ stride 2	relu
Max pooling	128	3 × 3 _ stride 4	relu
Inception (3a)	128	3 × 3 _ stride 4	relu
Inception (3b)	256	3 × 3 _ stride 2	relu
Max pooling	256	3 × 3_ stride 1	relu
Inception (4a)	256	3 × 3 _ stride 1	relu
Inception (4b)	256	3 × 3 _ stride 1	relu
Inception (4c)	256	3 × 3 _ stride 2	relu
Inception (4d)	512	3 × 3 _ stride 1	relu
Inception (4e)	512	3 × 3 _ stride 1	relu
Max pooling	512	3 × 3 _ stride 1	relu
Inception (5a)	512	3 × 3 _ stride 1	relu
Inception (5b)	512	3 × 3 _ stride 2	relu
Avg pooling	256	3 × 3 _ stride 2	relu
FC	—	—	relu
Softmax	—	—	Softmax



(A) GoogLeNet architecture



(B) GoogLeNet with skip connection

Evaluation Metrics and Results:

performance of Google-Net with skip connections for brain MRI abnormality detection. Google-Net, characterized by nine inception modules, shows notable feature learning improvements, with the last module achieving a high accuracy of 98.7%. The learning variations between inception blocks are comparatively small, with an accuracy progression difference of 16%. Google-Net consistently outperforms Alex-Net and VGG16 in terms of training progress and validation performance, reaching a minimum loss of 0.0005. The confusion matrix illustrates accurate classifications across diverse classes, yielding an overall accuracy of 98.5%. Notably, Google-Net's ROC values on original and augmented image data, at 97.9 and 98.6 respectively, surpass those of other networks, indicating superior performance. The study underscores the effectiveness of Google-Net with skip connections in enhancing feature learning for brain MRI abnormalities, with potential implications for medical image applications.

TABLE 7 Performance of GoogLeNet with skip connection

Layers	Accuracy (%)	Sensitivity (%)	Specificity (%)	F-score (%)	Precision (%)
Inception 3a	80	85	88	89	86
Inception 3b	91.5	90.2	94	92.5	92
Inception 4a	89.8	90	93.4	90	90
Inception 4b	91.1	90.2	95.2	91.5	90.7
Inception 4c	92	91	96	92.2	91
Inception 4d	94	92.5	94.1	94.2	94
Inception 4e	95.5	94	95	95.5	94.5
Inception 5a	97	96.2	96.7	97.1	96
Inception 5b	98.7	97	97.8	98.2	97.7

TABLE 8 Comparison of the transfer learning models

Accuracy %/models		AlexNet	VGG16	GoogLeNet
[31]	No skip connection	93.33	96.25	97.1
[32]		97.5	95	95
[30]		97.39	98.69	98.04
Our TL model	CNN with skip connection	97.6	98.92	98.3

Abbreviations: CNN, convolutional neural networks; TL, transfer learning.

4. Conclusions

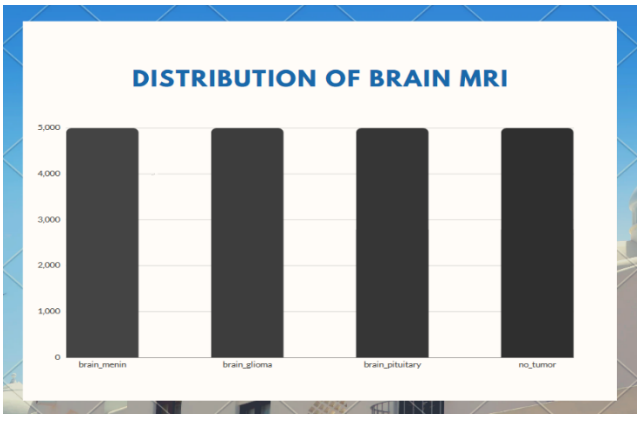
The article proposes a transfer learning (TL) model using convolutional neural networks (CNN) with skip connections to identify abnormalities in brain MRI scans, particularly for brain tumor classification. Three CNN architectures (Alex-Net, VGG16, and Google-Net) are employed, with VGG16 achieving the highest accuracy of 98.92% when incorporating skip connections. The use of skip connections helps mitigate vanishing gradient issues and reduces time complexity in TL networks. The study recommends incorporating a frequency-domain information enhancement technique during preprocessing for improved clarity in MRI images. Future work could include implementing ensemble classifiers to enhance classification accuracy and extending the model to a multiclass classifier for fine-tuning and freezing. The TL model's application may extend beyond brain tumor classification to other medical imaging domains such as other MRI images and skin cancer.

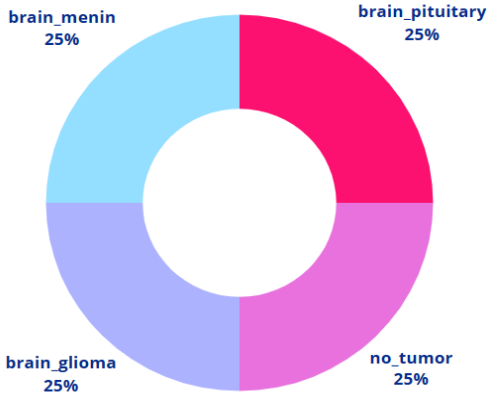
2.3 Details of Our Datasets

Overview: To create our Brain Cancer MRI (Magnetic Resonance Imaging) dataset, we collected some datasets, based on three different types of brain tumors (glioma, meningioma, pituitary) and another dataset does not contain any disease to train our model.

Number of Samples: 20000
[Training : Testing] Ratio → 16000 image : 4000 image → 80 % : 20 %
Diminution of images: (512 * 512)
Classes number: 4
Classes Labels: brain glioma, brain menin, brain pituitary, no_tumor

Analysis & Distribution Data:





2.4 Preprocessing

We are going to go through the steps of Image preprocessing needed to train, validate, and test the datasets.

What is Image Processing?

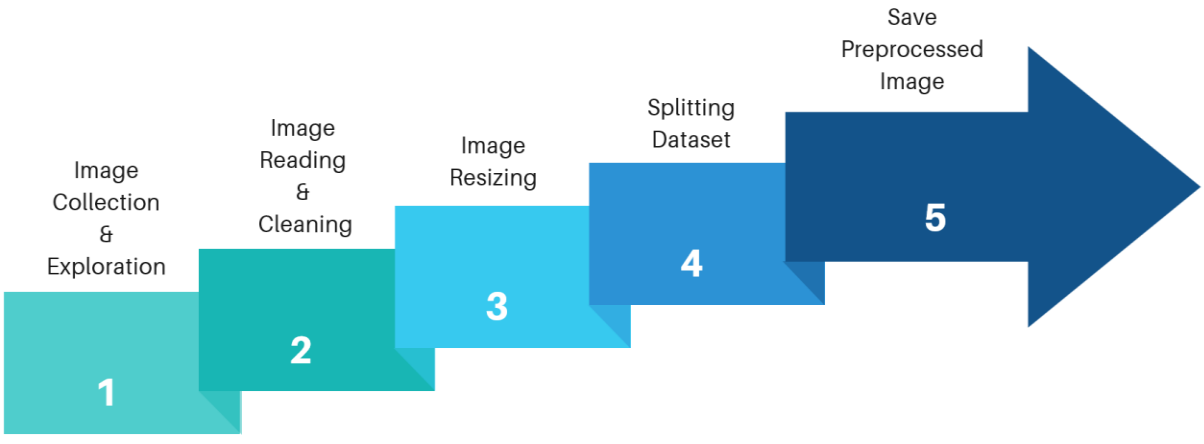
Image processing is divided into analog image processing and digital image processing.

Digital image processing is the use of computer algorithms to perform image processing on digital images. Digital image processing has many advantages over analog image processing. The aim of digital image processing is to improve the image data (features) by suppressing unwanted distortions.

An image is nothing more than a two-dimensional array of numbers (or pixels) Ranging between 0 and 255. It is defined by the mathematical function f (x, y) where X and y are the two coordinates horizontally and vertically.

Steps for our Image Preprocessing:

- Image Collection & Exploration
- Image Reading & Cleaning
- Image Resizing
- Splitting Dataset
- Save Preprocessed Images



Step 1: Image Collection & Exploration

We collect the MRI images from Kaggle and explore it.

Step 2: Image Reading & Cleaning

In this step, we store the path to our image dataset into a variable then we create a list to store all paths of images from this folder and we use function to clean anything except images.

Step 3: Image Resizing

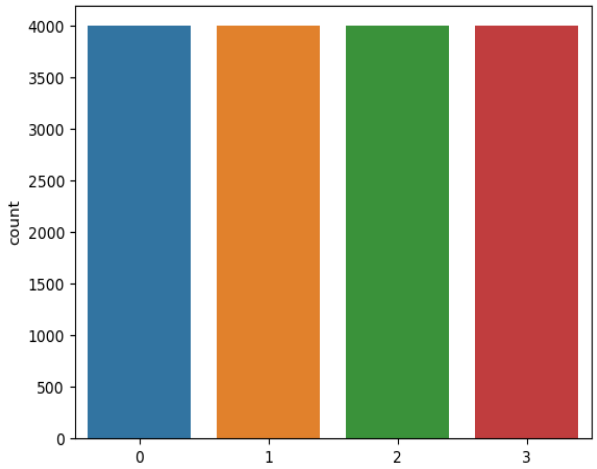
Many model architectures call for square input images, but few devices capture perfectly square images. Altering an image to be a square call for either stretching its dimensions to fit to be a square or keeping its aspect ratio constant and filling in newly created "dead space" with new pixels. Input images may be many sizes, and some may be smaller than the desired input size. So, we resize images dataset to (128*128) To train the model faster.

Original Size (512, 512, 3) - (Width, Height, No. RGB channels)
Resized (128, 128, 3)

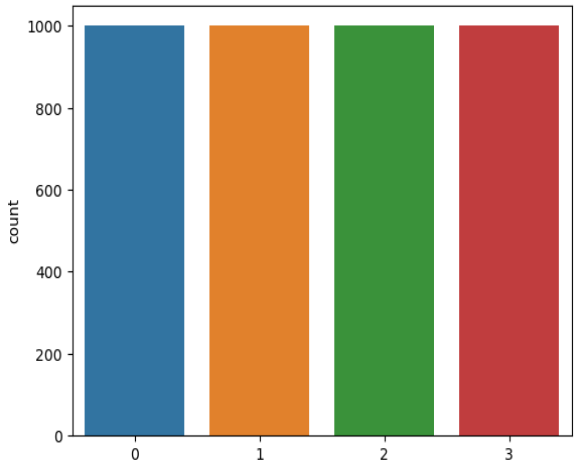
Step 4: Splitting Dataset

Then we split the dataset to training and testing (validation) to prepare data before feeding them into the neural network.

Training Data



Testing Data



These graphs show that the dataset is **balanced.**

Step 5: Save Preprocessed Images

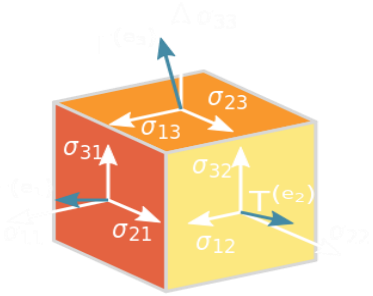
Finaly, we saved the training and testing dataset into two lists to fit it in with our model.

2.5 Literature Review

2.5.1 Layer used briefly in Traditional CNN

Now we will explain some of the CNN layers steps in detail first we get an image from the user then do multi-tasks of image processing to get an input image to the MRI model. After further processing image will be passed to the CNN layer. First, the image dimension will change to be ready for the second layer which is the convolutional layer, which changes among the layers until reaching the last layer. Let us talk about the basic layers used in detail.

2.5.2 CNN Layers



CNN is a class of artificial neural network Consists of:

1- Input layer
2- Hidden layer
3- Output layer

Input is a tensor with a shape: (number of inputs) x (input height) x (Input width) x (input channels)

A CNN architecture is formed by a stack of distinct layers input volume into an output volume through a differentiable function. A few distinct types of layers are commonly used.

1-Convolutional layer

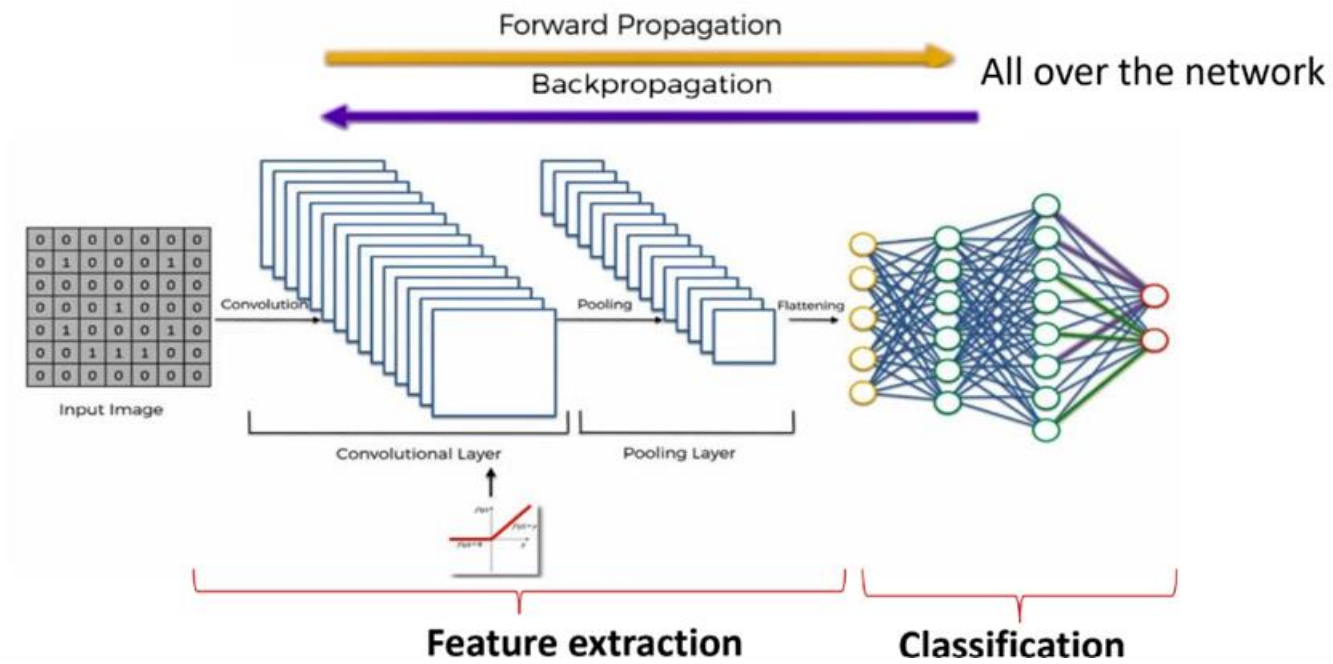
2-Pooling layer

3-ReLU layer

4-Fully connected layer

5-Loss Layer

CNN: Extraction & Classification



1. Convolutional Layer:

Local connectivity: Dealing with high-dimensional inputs such as images. It is impractical to connect neurons to all neurons in the previous volume because such a network architecture does not take the spatial structure of the data into account.

Spatial arrangement: Three hyperparameters control the size of the output volume of the convolutional layer: the depth, stride, and padding size.
Number of neurons that "fit"

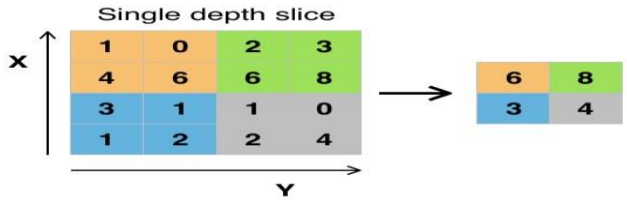
Input volume size W kernel field (filter) size K, Stride S, Padding P

$$\frac{W - K + 2P}{S} + 1.$$

2. Pooling Layer:

Max pooling is the most common. It partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum.

$$f_{X,Y}(S) = \max_{a,b=0}^{1} S_{2X+a,2Y+b}.$$



3. ReLU Layer:

It is an activation function what is the different between RELU & Sigmoid

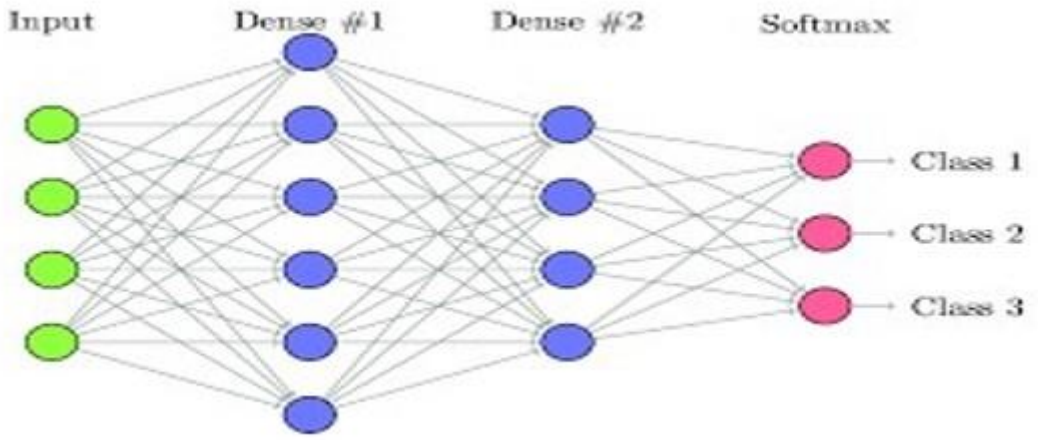
RELU does not have a negative value effectively removes negative values from an activation map by setting them to zero.

Sigmoid have a negative value.

ReLU is often preferred to other functions because it trains the neural network several times faster without a significant penalty to generalization accuracy.

4. Fully Connected Layer:

After several convolutional and max pooling layers, the final classification is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer.



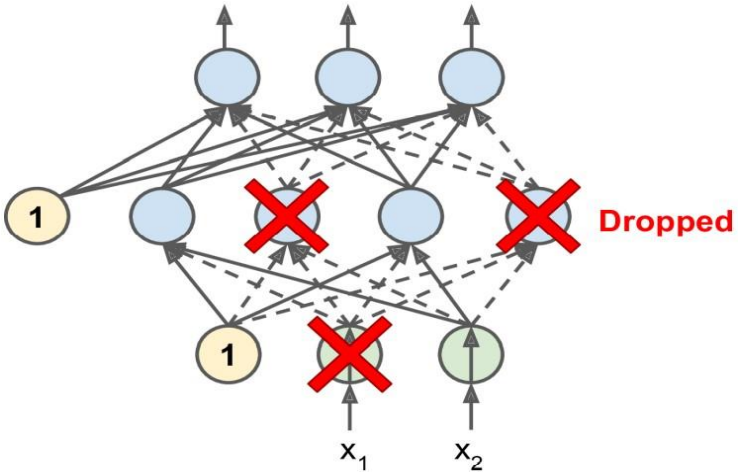
5. Loss layer:

Specifies how training penalizes the deviation between the predicted output of the network, and the true data labels.

The SoftMax loss function is used for predicting a single class of K mutually exclusive classes. [nb 3] Sigmoid cross-entropy loss is used for predicting K independent probability values in [0,1] {display style [0,1]}. Euclidean loss is used for regressing to real-valued labels $(-\infty, \infty)$.

2.5.3 Dropout

Dropout is a regularization technique used to prevent overfitting in deep neural networks. It involves randomly "dropping out" neurons, excluding them from the training process at each step with a set probability (dropout rate), typically 50%. This prevents neurons from co-adapting and relying too much on specific inputs, promoting a more robust network that generalizes better.



During training, each step produces a unique neural network among a vast number of possible networks (2^N, where N is the number of droppable neurons). Although these networks share weights, they are distinct. The final neural network can be viewed as an ensemble of these smaller networks.

To compensate for the increased connections during testing (due to dropout), the input connection weights are adjusted after training. This adjustment ensures that neurons receive signals within the expected range for optimal performance.

It's crucial to evaluate the model's performance without dropout to avoid misleading comparisons between training and validation losses. If overfitting is observed, increasing the dropout rate may be beneficial, while decreasing it could address underfitting. Adjusting dropout rates for layer sizes and considering partial dropout after the last hidden layer are additional strategies. Although dropout may slow down convergence, proper tuning often results in a significantly improved model, justifying the additional time and effort invested.

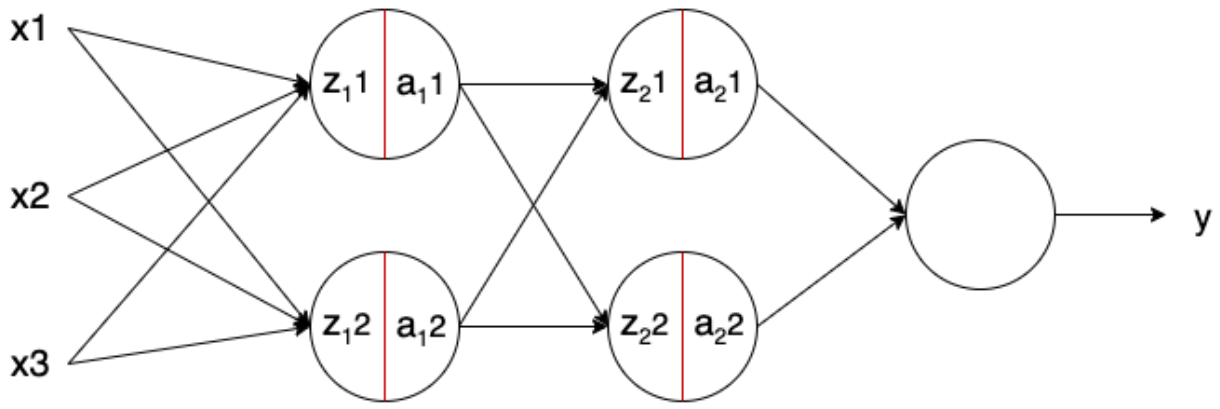
2.5.4 Batch Normalization

Batch Norm is a normalization technique done between the layers of a Neural Network instead of in the raw data. It is done in mini batches instead of the full data set. It serves to speed up training and use higher learning rates, making learning easier.

Following the technique explained in the previous section, we can define the normalization formula of Batch Norm as:
being the mean of the neurons' output and the standard deviation of the neurons' output.

How Is It Applied?

In the following image, we can see a regular feed-forward Neural Network: are the inputs, the output of the neurons, the output of the activation functions, and the output of the network:



Batch Norm – in the image represented with a red line – is applied to the neurons' output just before applying the activation function. Usually, a neuron without Batch Norm would be computed as follows:

$$z = g(w, x) + b; \qquad a = f(z)$$

Being $g()$ the linear transformation of the neuron, the weights of the neuron, the bias of the neurons, and the activation function. The model learns the parameters and. Adding Batch Norm, it looks as:

$$z = g(w, x); \qquad z^N = \left(\frac{z - m_z}{s_z}\right) \cdot \gamma + \beta; \qquad a = f(z^N)$$

being the output of Batch Norm, the mean of the neurons' output, $s_z$the standard deviation of the output of the neurons, and \gamma and \beta learning parameters of Batch Norm. Note that the bias of the neurons (b) is removed. This is because as we subtract the mean m_z, any constant over the values of z – such as b – can be ignored as it will be subtracted by itself.

The parameters beta and gamma shift the mean and standard deviation, respectively. Thus, the outputs of Batch Norm over a layer result in a distribution with a mean \beta and a standard deviation of \gamma. These values are learned over epochs and the other learning parameters, such as the weights of the neurons, aiming to decrease the loss of the model.

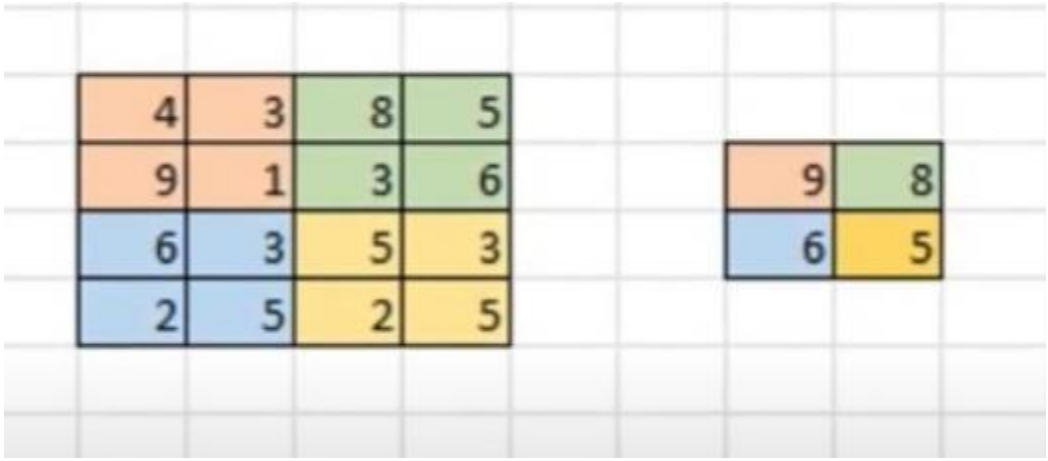
Implementation in Python:

Implementing Batch Norm is quite straightforward when using modern Machine Learning frameworks such as Keres, TensorFlow, or Pytorch. They come with the most used methods and are up to date with state-of-the-art.

2.5.5 Max Pooling 2D

Max pooling is a type of operation that is typically added to CNN's following individual Convolutional layers.
When added to a model max pooling reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer.

How max pooling works:



so here on the Left we have some sample input of size four by four and now we're going to use the same two by two filter size with a stride of two to do max pooling on this input so our first two by two region is here in orange and we can see the max value of this region is nine so we store that over here in our output then we slide over by two pixels and we see the max value in the green region is eight and we stored that over here in our output as well since we've now reached the edge we move back over to the far left and go down by two pixels here the max value in the blue region is six and we store that here in our output and finally we move to the right by two and see the max value of the yellow region is five so we store that over here in our output as well now we've just gone through the complete process of max pooling on this sample 4x4 input and the resulting output is this 2x2 block here so our input dimensions were again reduced by a factor of two.

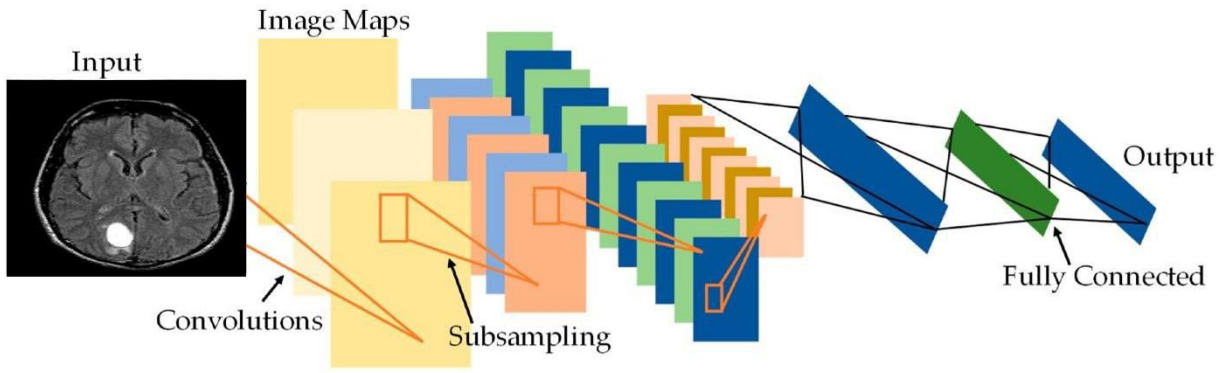
Why add max pooling to our network?

- max pooling is reducing the resolution of the given output of a convolutional layer the network will be looking at larger areas of the image at a time going forward which reduces the number of parameters in the network and consequently reduces computational load and max pooling help to reduce overfitting.

Chapter 3: CNN & Transfer Learning

3.1 What is CNN?

CNNs are a type of deep learning algorithm that are used to process data with a grid like topology. CNNs are a type of deep learning algorithm that is used to process data that has a spatial or temporal relationship. CNNs are like other neural networks, but they have an added layer of complexity since they use a series of convolutional layers. Convolutional layers are an essential component of Convolutional Neural Networks (CNNs).

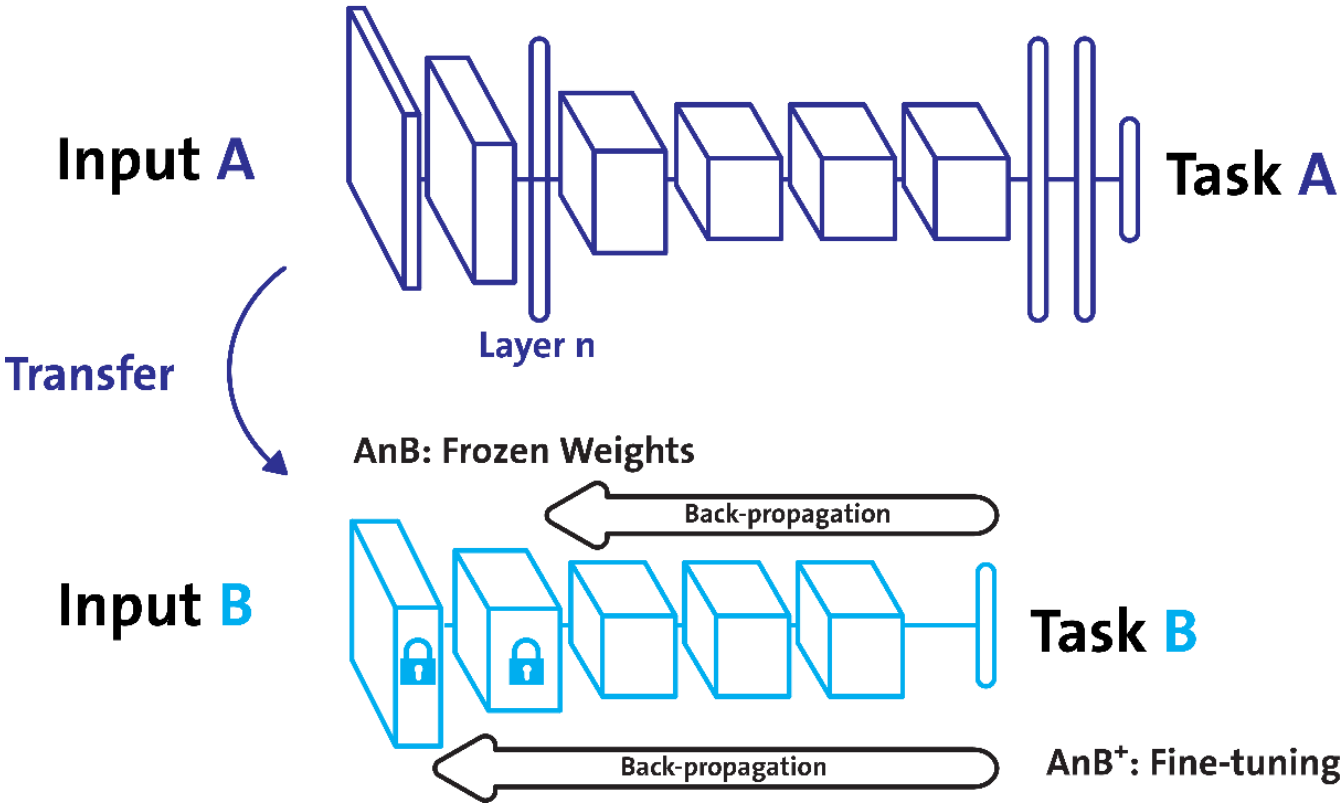


3.2 What Is Transfer Learning?

Transfer learning is a machine-learning method where the application of knowledge obtained from a model used in one task can be reused as a foundation point for another task.

During transfer learning, the knowledge leveraged and rapid progress from a source task is used to improve the learning and development to a new target task. The application of knowledge is using the source task's attributes and characteristics, which will be applied and mapped onto the target task.

However, if the transfer method results in a decrease in the performance of the new target task, it is called a negative transfer. One of the major challenges when collaborating with transfer learning methods is being able to provide and ensure the positive transfer between related tasks while avoiding the negative transfer between less-related tasks.



3.3 Types of Transfer Learning

Inductive Transfer Learning: In this type of transfer learning, the source and target task are the same, however, they are still different from one another. The model will use inductive biases from the source task to help improve the performance of the target task. The source task may or may not contain labeled data, further leading onto the model using multitask learning and self-taught learning.

Unsupervised Transfer Learning: I assume you know what unsupervised learning is, however, if you do not, it is when an algorithm is subjected to being able to identify patterns in datasets that have not been labeled or classified. In this case, the source and target are similar, however, the task is different, where both data are unlabeled in both source and target. Techniques such as dimensionality reduction and clustering are well known in unsupervised learning.

Transudative Transfer Learning: In this last type of transfer learning, the source and target tasks share similarities, however, the domains are different. The source

domain contains a lot of labeled data, whereas there is an absence of labeled data in the target domain, further leading onto the model using domain adaptation.

3.4 Transfer Learning Pros and Cons

Transfer Learning Pros: Better base: Using a pre-trained model in transfer learning offers you a better foundation and starting point, allowing you to perform some tasks without even training.
Higher learning rate: Due to the model already having been trained on a similar task beforehand, the model has a higher learning rate.

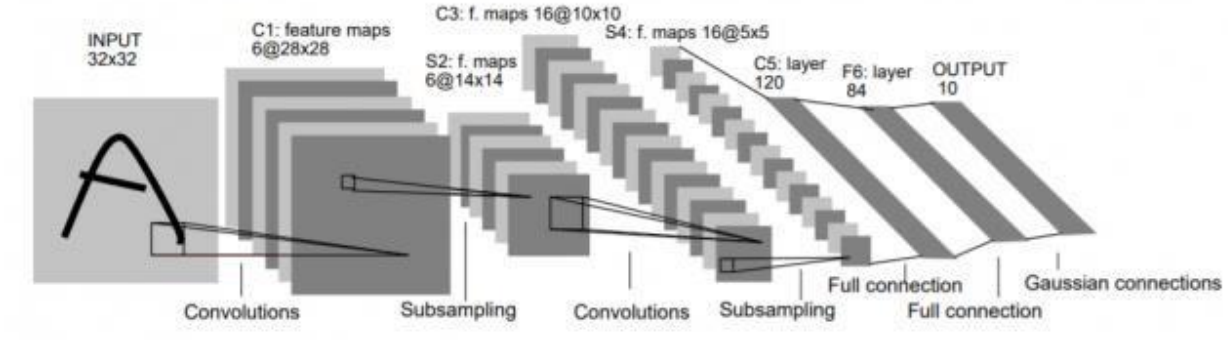
Higher accuracy rate: With a better base and higher learning rate, the model works at a higher performance, producing more accurate outputs.

Transfer Learning Cons and Problems: Transfer learning should be avoided when the weights trained from your source task are different from your target task. For example, if your previous network was trained for classifying cats and dogs and your new network is trying to detect shoes and socks, there is going to be a problem as the weights transferred from your source to the target task will not be able to give you the best of results. Therefore, initializing the network with pre-trained weights that correspond with similar outputs to the one you are expecting is better than using weights with no correlation.

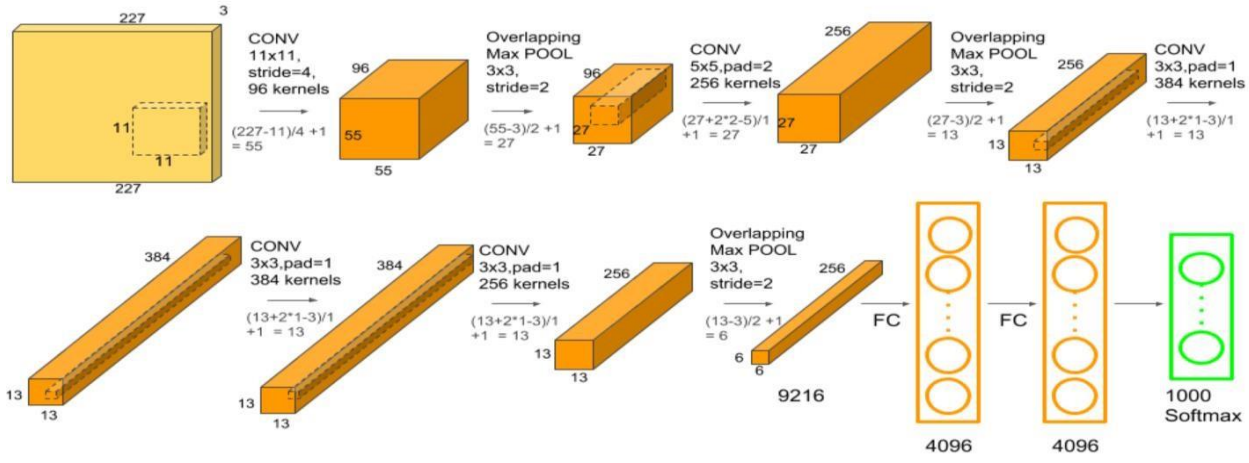
Removing layers from a pre-trained model will cause issues with the architecture of the model. If you remove the first layers, your model will have a low learning rate as it must juggle working with low-level features. Removing layers reduces the number of parameters that can be trained, which can result in overfitting. Being able to use the correct number of layers is vital in reducing overfitting, however, this is also a timely process. Negative transfer learning: negative transfer learning is when a previous learning method obstructs the new task. This only occurs if the source and target are not similar enough, causing the first round of training to be too far off. Algorithms do not have to always agree with what we deem as similar, making it difficult to understand the fundamentals and standards of what type of training is sufficient.

3.5 Different types of CNN Architectures

3.5.1 Le-Net: The Le-Net CNN is a simple yet powerful model that has been used for various tasks such as handwritten digit recognition, traffic sign recognition, and face detection. Although LeNet was developed more than 20 years ago, its architecture is still relevant today and continues to be used.

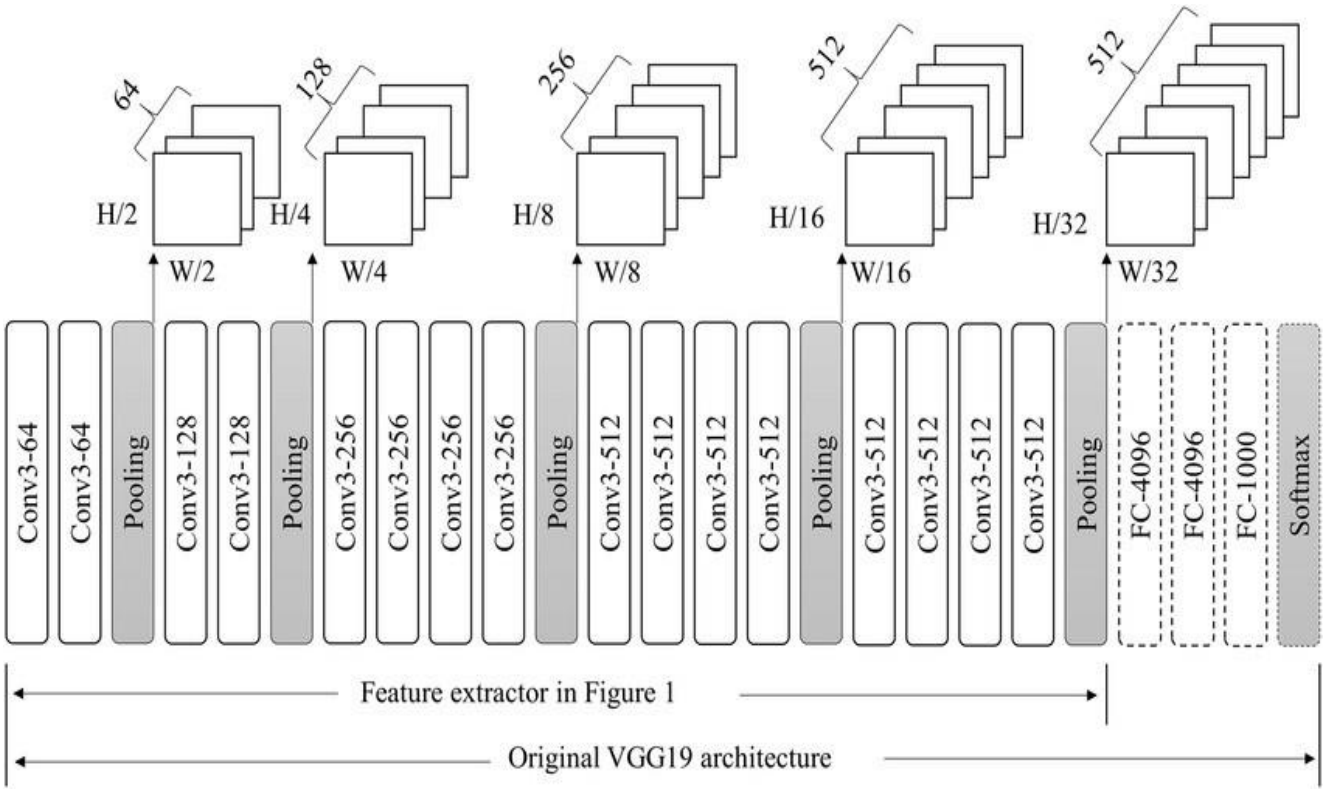


3.5.2 Alex-Net: The Alex Net architecture was designed to be used with large-scale image datasets and it achieved state-of-the-art results at the time of its publication. Alex-Net is composed of five convolutional layers with a combination of max pooling layers, 3 fully connected layers, and 2 dropout layers. The activation function used in all layers is ReLU. The activation function used in the output layer is SoftMax. The total number of parameters in this architecture is around sixty million.



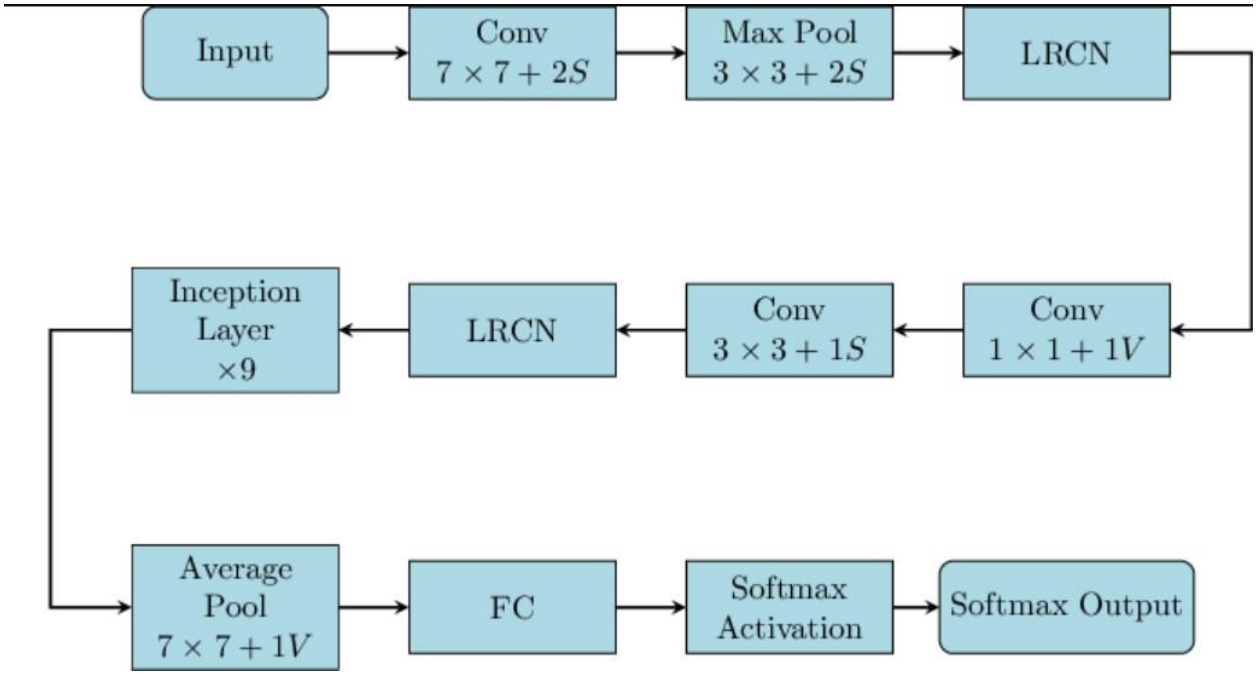
3.5.3 ZF-Net: The network has relatively fewer parameters than Alex-Net, is computationally more efficient than Alex-Net by introducing an approximate inference stage through deconvolutional layers in the middle of CNNs.

3.5.4 VGG-Net: VGG Net is a 16-layer CNN with up to ninety-five million parameters and trained on over one billion images (1000 classes). It can take large input images of 224 x 224-pixel size for which it has 4096 convolutional features. CNNs with such large filters are expensive to train and require a lot of data, which is the main reason CNN architectures like Google-Net (Alex Net architecture) work better than VGG Net for most image classification tasks where input images have a size between 100 x 100-pixel and 350 x 350 pixels. Real-world applications of VGG Net CNN architecture include the ILSVRC 2014 classification task, which was also won by Google Net CNN architecture. The VGG CNN model is computationally efficient and serves as a strong baseline for many applications in computer vision due to its applicability for numerous tasks including object detection. Its deep feature representations are used across multiple neural network architectures like YOLO, SSD.



3.5.5 Google-Net: It has been shown to have a notably reduced error rate in comparison with previous winners Alex-Net and ZF-Net. In terms of error rate, the error is significantly lesser than VGG. It achieves deeper architecture by employing several distinct techniques, including 1×1 convolution and global average pooling. Google-Net CNN architecture is computationally expensive.

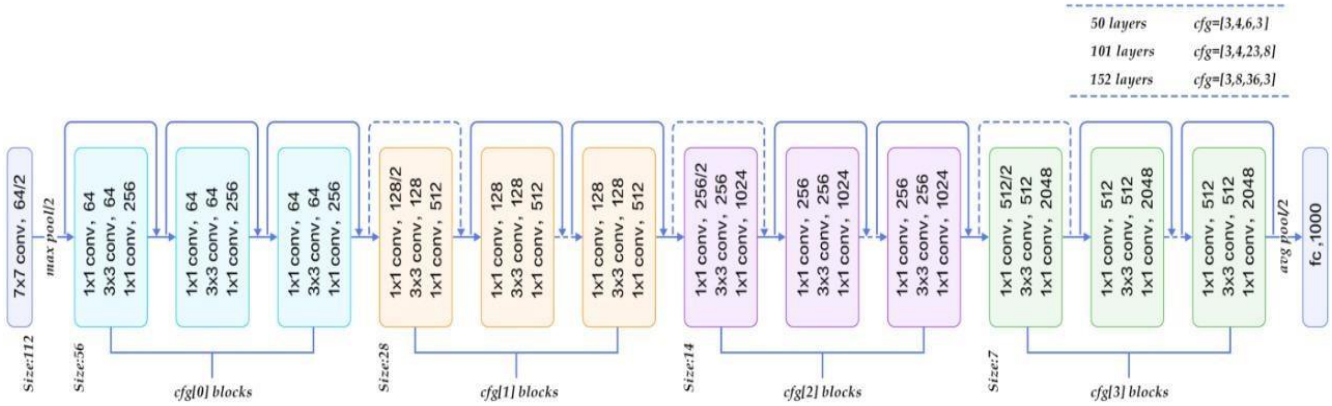
Real-world applications: Street View House Number, digit recognition task, which is often used as a proxy for roadside object detection.



3.5.6 Residual-Net: The network has 152 layers and over one million parameters, which is considered deep even for CNNs because it would have taken more than 40 days on 32 GPUs to train the network on the ILSVRC 2015 dataset. CNNs are mostly used for image classification tasks with 1000 classes, but Res-Net proves that CNNs can also be used successfully to solve natural language processing problems like sentence completion or machine comprehension.

Real-life applications of Res-Net CNN architecture include Microsoft's machine comprehension system, which has used CNNs to generate the answers for more than 100k questions in over 20 categories. The CNN architecture Res-Net is

computationally efficient and can be scaled up or down to match the computational power of GPUs.



3.5.7 Mobile-Nets: Mobile-Nets are CNNs that can be fitted on a mobile device to classify images or detect objects with low latency.

They are usually very small CNN architectures, which makes them easy to run in real-time using embedded devices like smartphones and drones. The architecture is also flexible, so it has been tested on CNNs with 100-300 layers and it still works better than other architectures like VGG-Net. Real-life examples of Mobile-Nets CNN architecture include CNNs that are built into Android phones to run Google's Mobile Vision API, which can automatically identify labels of popular objects in images.

3.6 CNN vs Transfer Learning

For CNN you need to do more preprocessing of the dataset but with transfer learning, you only need to do little processing of the dataset like resize to 227 x 227 or 224 x224 according to selected Pre-trained Models (Alex-Net, Google-Net, Res-Net, VGG Networks) this saves much time of preprocessing data.
Pretrained Models may be given much greater accuracy for completely new datasets such as image datasets.

As we use transfer learning it gives higher accuracy. While CNN gives slightly less accuracy than transfer learning. When we have used a single CPU which itself takes a lot of time in training and one of the models also failed to train the dataset. Thus, parallel computing should be preferred in the transfer learning of training data.

Advantages of Transfer Learning

- Transfer learning is useful for insufficient data and imbalanced class problems. So that we can limit training data and training time. It can give a hundred percent accuracy for less amount of data.
- Transfer learning gives a good combination of features even for complex tasks within a brief period.

Disadvantages of CNN

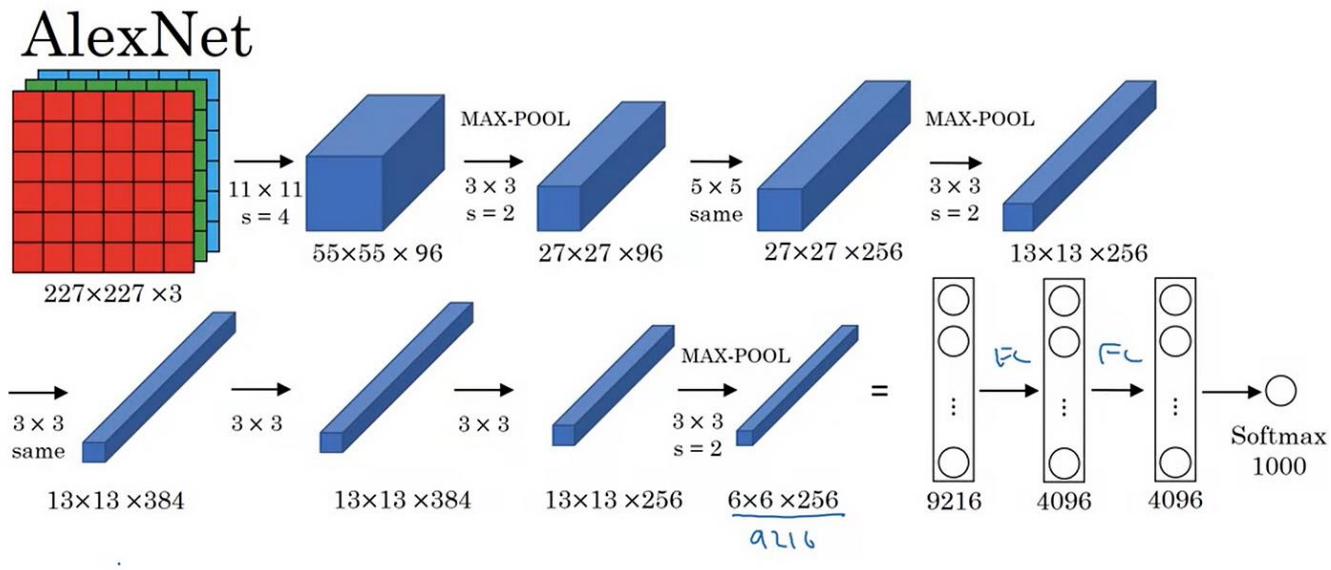
- Training a convolutional neural network takes a long time, especially with huge datasets.
- While CNNs are translation-invariant, they struggle with rotation and scale invariance unless data is augmented explicitly.

But even so we have used CNNs, and we got the best accuracy for brain cancer classification project.

3.7 Proposed Models

3.7.1 The Proposed Transfer Learning Alex-Net Model

As we mentioned in the previous chapter, Alex-Net is a pioneering convolutional neural network (CNN) architecture that won the 2012 ImageNet competition. It consists of eight layers, including five convolutional and three fully connected layers. Alex-Net introduced concepts like ReLU activation, local response normalization, and dropout, demonstrating the power of deep learning in computer vision.

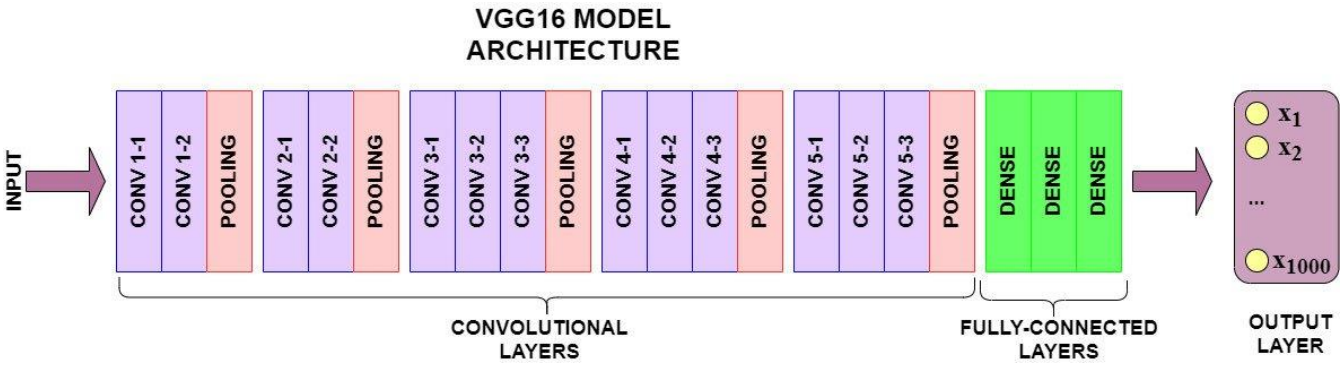


[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Andrew Ng

3.7.2 The Proposed Transfer Learning VGG Model

VGG16, or Visual Geometry Group 16, is a deep convolutional neural network architecture designed for image classification tasks. It was introduced by the Visual Geometry Group at the University of Oxford. VGG16 is characterized by its simplicity and uniform architecture, consisting of 16 layers, including 13 convolutional layers and 3 fully connected layers. The convolutional layers use small 3x3 filters with a stride of 1, and max-pooling layers are applied to downsample the spatial dimensions. VGG16 has been widely used as a baseline model in computer vision and has achieved strong performance on various image recognition benchmarks.

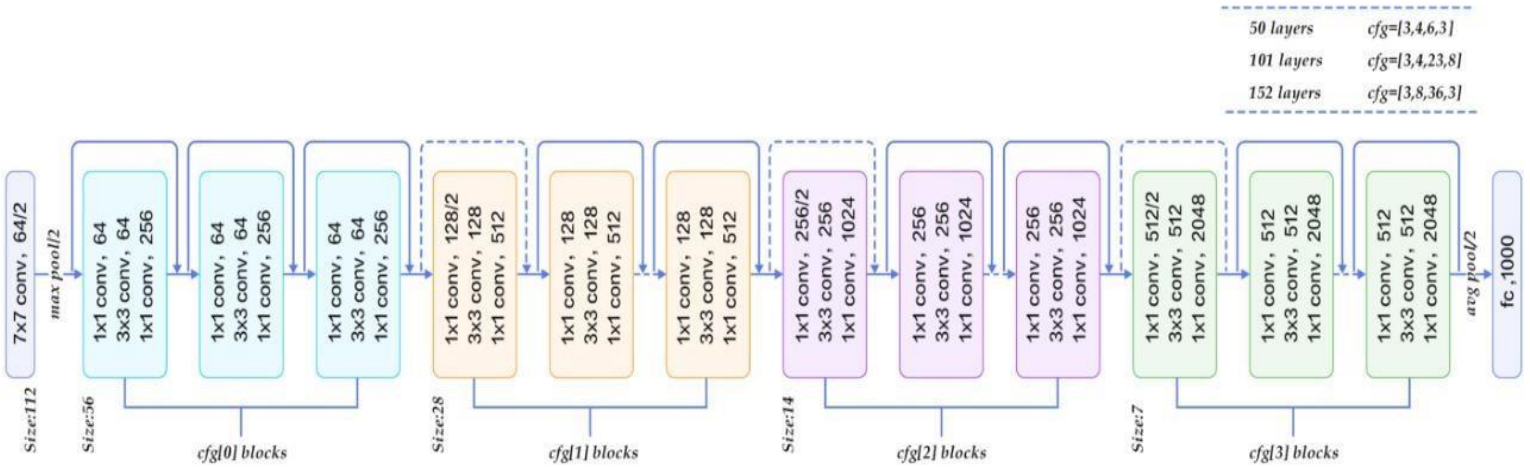


- The only preprocessing that was done was that they subtracted the mean RGB value from each pixel, computed over the whole training set.
- Used kernels of (3 * 3) size with a stride size of 1 pixel, this enabled them to cover the whole notion of the image.
- spatial padding was used to preserve the spatial resolution of the image.
- max pooling was performed over a (2 * 2) pixel window with S = 2.
- this was followed by Rectified linear unit (ReLU) to introduce non-linearity to make the model classify better and to improve computational time as the previous models used tanh or sigmoid functions this proved much better than those.
- implemented three fully connected layers from which the first two were of size 4096 and after that a layer with 1000 channels for 1000-way ILSVRC classification and the final layer is a soft-max function.

3.7.3 The Proposed Transfer Learning Res-Net Model

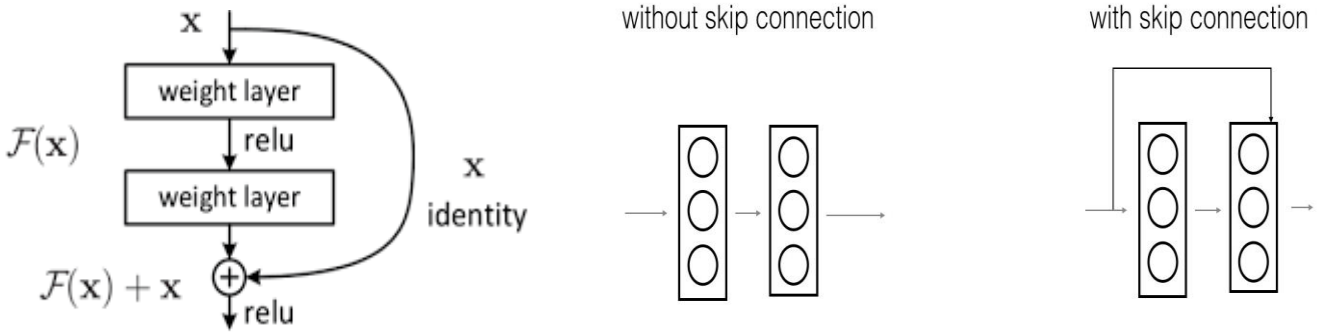
Short for Residual Networks is a classic neural network used as a backbone for many computers vision tasks.

Our work depends on using a pre-trained architecture named Res-Net which consists of 50 layers have 4 blocks each one consists of 3 blocks each block consists of a 3 conv2D layers the architecture is built as shown in the next diagram.



Skip Connection — The Strength of Res-Net

Res-Net first introduced the concept of skip connection. The diagram below illustrates skip connection. The figure on the left is stacking convolution layers together one after the other. On the right we still stack convolution layers as before but we now also add the original input to the output of the convolution block. This is called skip connection.



3.7.4 The Proposed CNN Model

- We have made some modifications to the model of the first paper to get the best results.

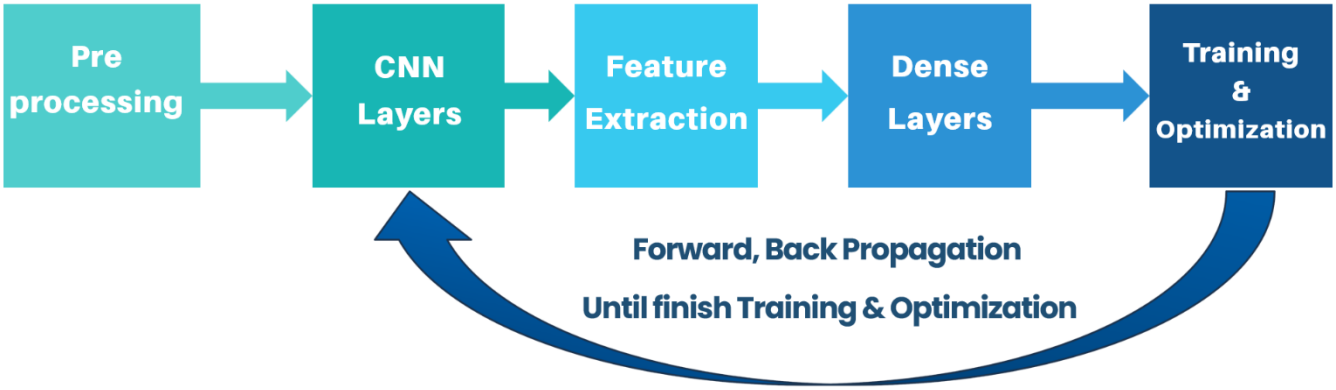
- The CNN architecture is defined and consists of four convolutional layers followed by four layers, and we add batch normalization and dropout (type of regularization) to avoid overfitting.

- Our work depends on using Convolutional Neural Networks to make a multi classification of different Brain Cancer with different optimizers like Max Pooling and Batch Normalization with flatten and dense layer with ReLU and SoftMax techniques.

- Consists of 20 layers the output of each layer is input to the next layer.

1. Conv2D
2. Batch Normalization
3. Max Pooling
4. Dropout
5. Conv2D
6. Max Pooling
7. Dropout
8. Conv2D
9. Max Pooling
10. Dropout
11. Conv2D
12. Max Pooling
13. Dropout
14. Flatten
15. Dense (ReLU)
16. Dropout
17. Dense (ReLU)
18. Dropout
19. Dense (ReLU)
20. Dense (SoftMax)

Model Training:



Model:

```
model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (3, 3), activation = 'relu', input_shape = (IMGSIZE, IMGSIZE, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 128, kernel_size = (3, 3), activation = 'relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 256, kernel_size = (3, 3), activation = 'relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation = 'relu'))
model.add(Dropout(0.25))

model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.25))

model.add(Dense(64, activation = 'relu'))
model.add(Dense(4, activation = 'softmax'))

model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
```

```
early_stopping = EarlyStopping(monitor = 'val_loss', patience = 5, restore_best_weights = True)
results = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs = 25, batch_size = 32, callbacks = [early_stopping])
# val_loss, val_acc = model.evaluate(X_test, y_test)

y_pred = model.predict(X_test)

model.summary()
```

Hyper-Parameters:

```
numOfCNNLayers = 4 & Activation_functhion = 'relu'
kernal_size = (3,3)& MaxPooling = (2,2)
numOfDenseLayers = 4 & Activation_functhion = 'relu', 'softmax' for output
optimizer = 'adam'
loss = 'sparse_categorical_crossentropy'
epochs = 25
batch_size = 32 (Defualt)
image_size = 128 * 128
channels = 3 (RGB)
```

We have changed the hyperparameters in the model many times to get the best results**.**

Chapter 4: Models Results and Graphs

4.1 Transfer Learning Results

4.1.1 Alex-Net Results

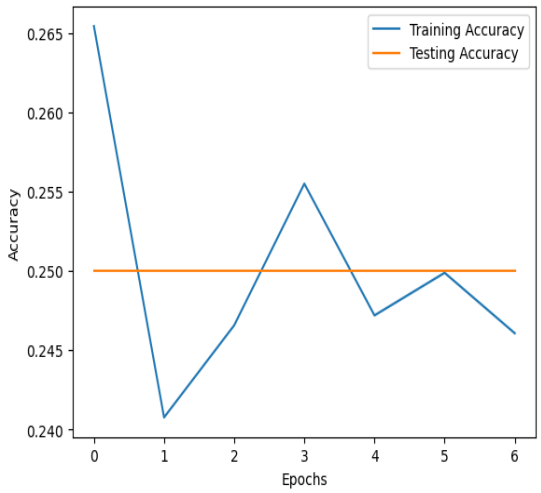
Summary of Model:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 55, 55, 96)	34944
batch_normalization (Batch Normalization)	(None, 55, 55, 96)	384
max_pooling2d (MaxPooling2D)	(None, 27, 27, 96)	0
conv2d_1 (Conv2D)	(None, 27, 27, 256)	614656
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 256)	0
conv2d_2 (Conv2D)	(None, 13, 13, 384)	885120
conv2d_3 (Conv2D)	(None, 13, 13, 384)	1327488
conv2d_4 (Conv2D)	(None, 13, 13, 256)	884992
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 256)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 4096)	37752832
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 4)	16388

Total params: 58298116 (222.39 MB)
Trainable params: 58297924 (222.39 MB)
Non-trainable params: 192 (768.00 Byte)

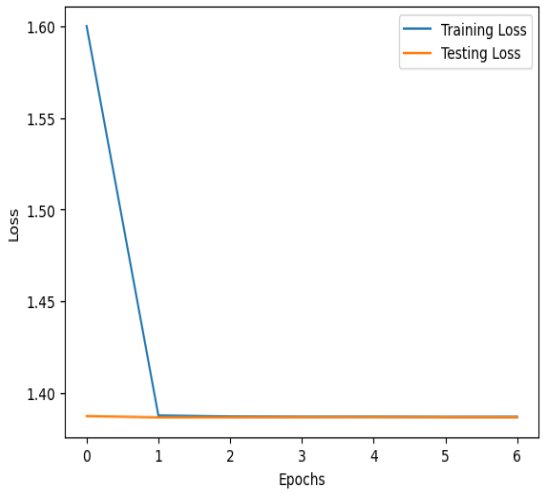
flatten (Flatten)	(None, 9216)	0

Accuracy Graph



Training Accuracy = 24.61 %
Validation Accuracy = 25.0 %

Loss Graph



Training Loss = 100.0 %
Validation Loss = 100.0 %

4.1.2 VGG16-Net Results

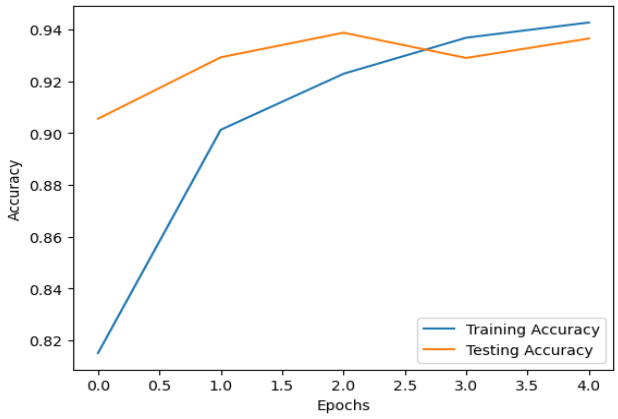
Summary of Model:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 4, 4, 512)         14714688

 max_pooling2d_3 (MaxPooling  (None, 2, 2, 512)        0
 2D)

 flatten_2 (Flatten)         (None, 2048)              0

 dense_9 (Dense)             (None, 128)               262272

 dropout_3 (Dropout)         (None, 128)               0

 dense_10 (Dense)            (None, 64)                8256

 dense_11 (Dense)            (None, 4)                 260

=================================================================
Total params: 14,985,476
Trainable params: 270,788
Non-trainable params: 14,714,688
_________________________________________________________________
```

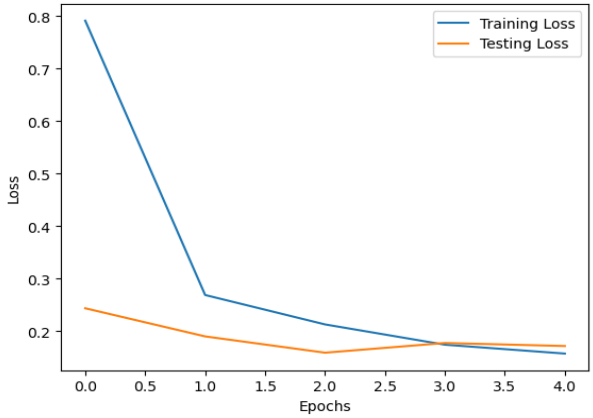
Performance:

	precision	recall	f1-score	support
brain menin	0.91	0.97	0.94	1000
brain glioma	0.93	0.83	0.88	1000
brain pituitary	0.99	1.00	0.99	1000
no tumor	0.91	0.95	0.93	1000
Accuracy			0.94	4000
macro avg	0.94	0.94	0.94	4000
weighted avg	0.94	0.94	0.94	4000

Accuracy Graph



Loss Graph



Training Accuracy = 94.27 %
Validation Accuracy = 93.65 %

Training Loss = 15.76 %
Validation Loss = 17.20 %

4.1.3 Residual-Net Results

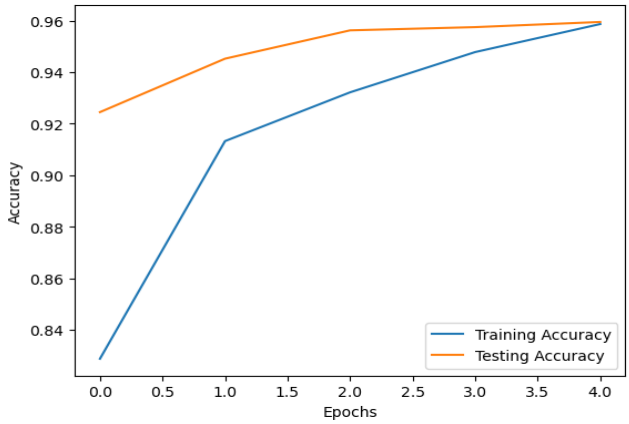
Summary of Model:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 4, 4, 2048)        23587712

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 256)               8388864

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 4)                 260

=================================================================
Total params: 32,017,988
Trainable params: 8,430,276
Non-trainable params: 23,587,712
_________________________________________________________________
```

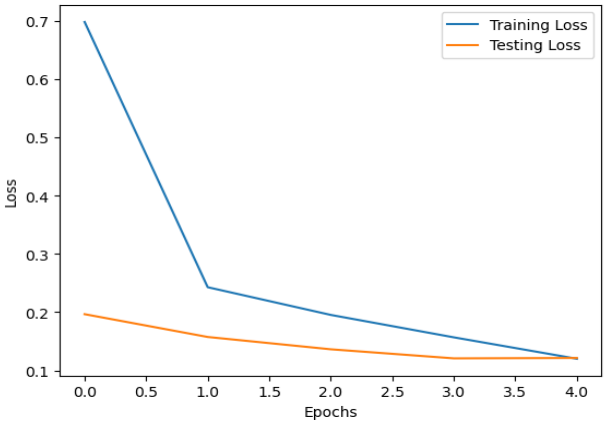
Performance:

	precision	recall	f1-score	support
brain menin	0.93	0.98	0.96	1000
brain glioma	1.00	1.00	1.00	1000
brain pituitary	0.94	0.91	0.92	1000
no tumor	0.97	0.96	0.96	1000
Accuracy			0.96	4000
macro avg	0.96	0.96	0.96	4000
weighted avg	0.96	0.96	0.96	4000

Accuracy Graph



Loss Graph



Training Accuracy = 95.87 %
Validation Accuracy = 95.55 %

Training Loss = 11.99 %
Validation Loss = 12.13 %

Discussion & Notes about TL Models:

- There are many problems with Alex-Net Model because it is overfitting, and the model can't work with the dataset. So, we can't train this model and the solution is with the next models, especially Custom CNN.

- VGG16 Model and Res-Net Model have shown great results in a very small number of epochs during the training, and we rely completely on our project on our Custom CNN.

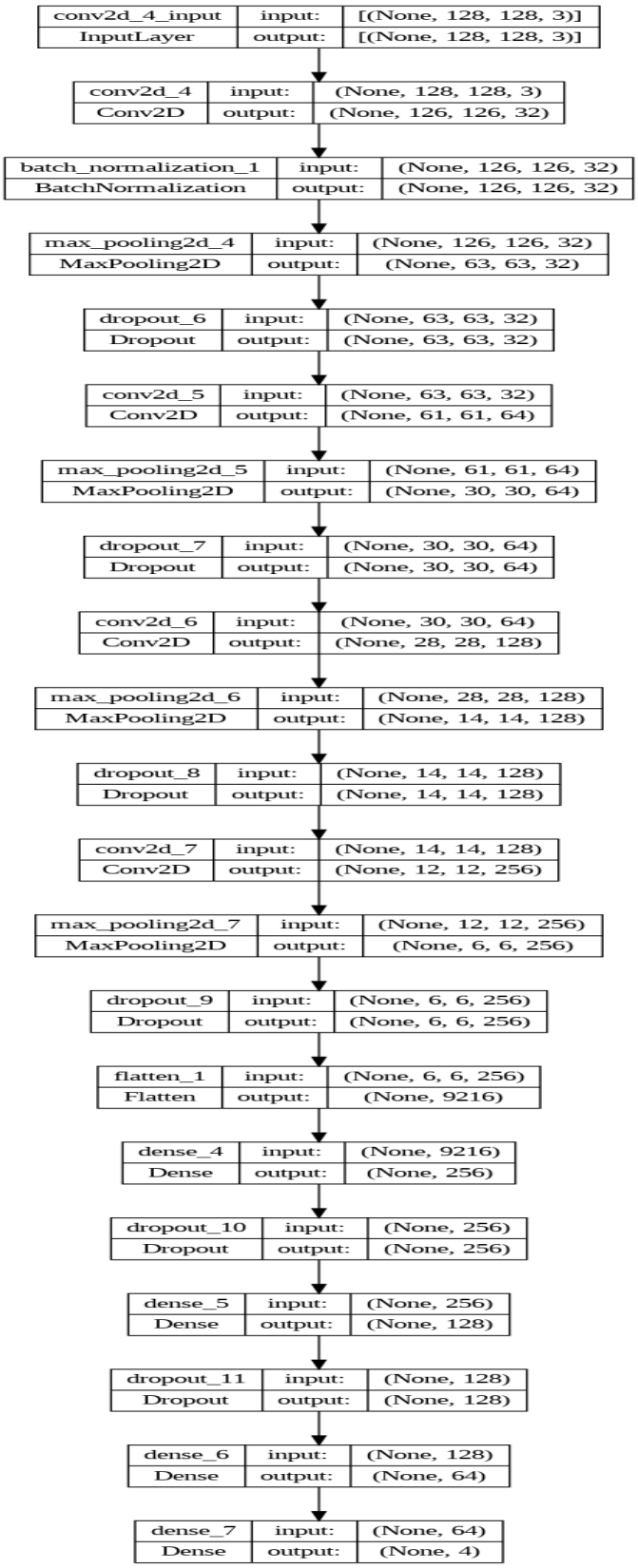
- Therefore, we did not care much about training both models (VGG16 and Res-Net) any further, and we were satisfied with these results.

4.2 Custom CNN Results

Summary of Model:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 126, 126, 32)      896

batch_normalization_2 (Batc  (None, 126, 126, 32)      128
hNormalization)

max_pooling2d_8 (MaxPooling  (None, 63, 63, 32)        0
2D)

dropout_12 (Dropout)         (None, 63, 63, 32)        0

conv2d_9 (Conv2D)            (None, 61, 61, 64)        18496

max_pooling2d_9 (MaxPooling  (None, 30, 30, 64)        0
2D)

dropout_13 (Dropout)         (None, 30, 30, 64)        0

conv2d_10 (Conv2D)           (None, 28, 28, 128)       73856

max_pooling2d_10 (MaxPoolin  (None, 14, 14, 128)       0
g2D)

dropout_14 (Dropout)         (None, 14, 14, 128)       0

conv2d_11 (Conv2D)           (None, 12, 12, 256)       295168

max_pooling2d_11 (MaxPoolin  (None, 6, 6, 256)         0
g2D)

dropout_15 (Dropout)         (None, 6, 6, 256)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 9216)              0

dense_8 (Dense)              (None, 256)               2359552

dropout_16 (Dropout)         (None, 256)               0

dense_9 (Dense)              (None, 128)               32896

dropout_17 (Dropout)         (None, 128)               0

dense_10 (Dense)             (None, 64)                8256

dense_11 (Dense)             (None, 4)                 260

=================================================================
Total params: 2,789,508
Trainable params: 2,789,444
Non-trainable params: 64
_________________________________________________________________
```

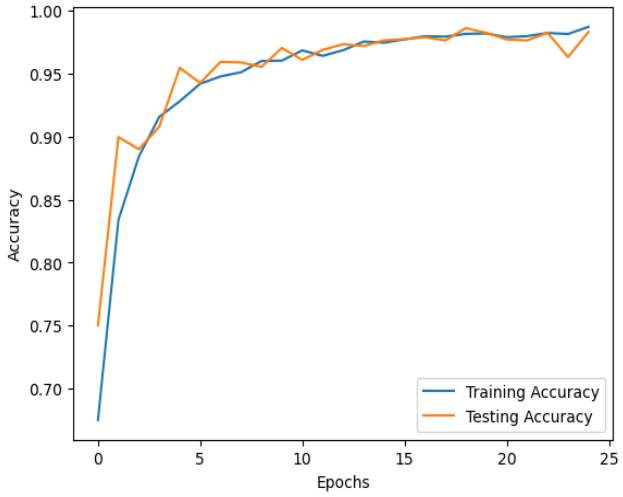
Block Diagram:



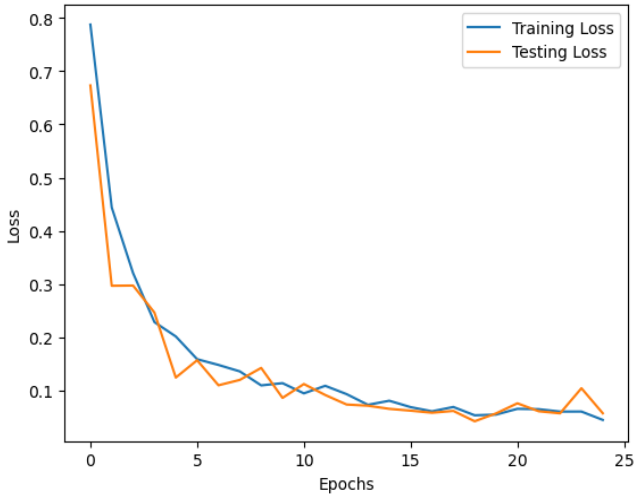
Performance:

	precision	recall	f1-score	support
brain menin	0.95	0.99	0.97	1000
brain glioma	1.00	0.98	0.99	1000
brain pituitary	0.99	0.97	0.98	1000
no tumor	1.00	1.00	1.00	1000
accuracy			0.98	4000
macro avg	0.98	0.98	0.98	4000
weighted avg	0.98	0.98	0.98	4000

Accuracy Graph



Loss Graph



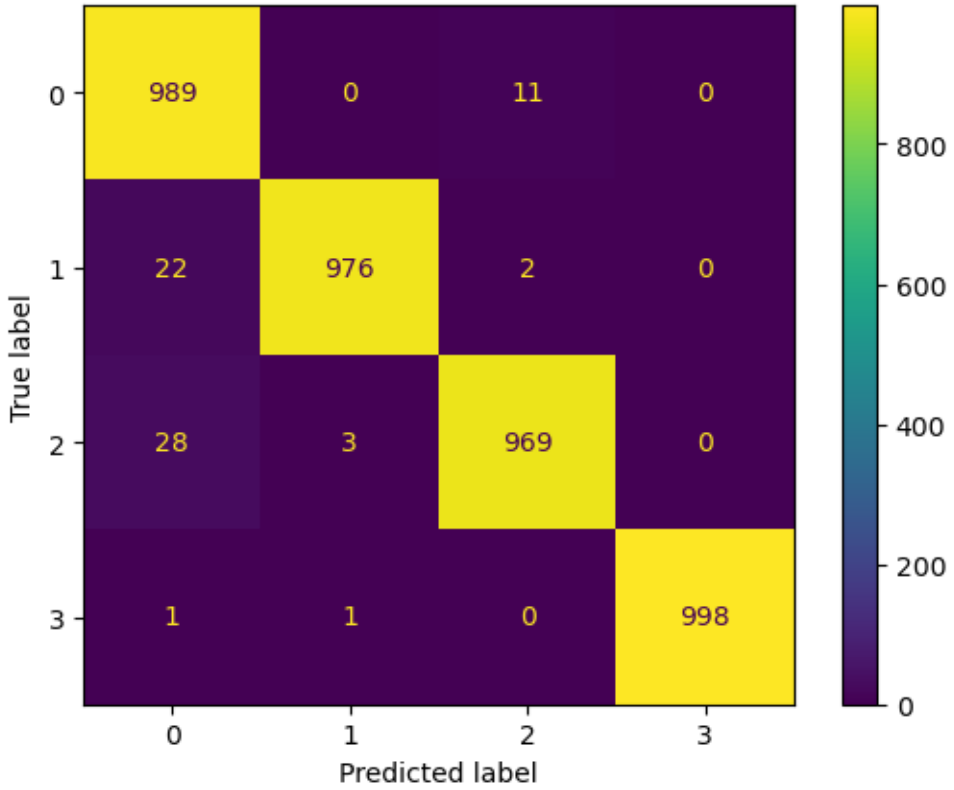
Training Accuracy = 98.70 %

Validation Accuracy = 98.30 %

Training Loss = 5.00 %

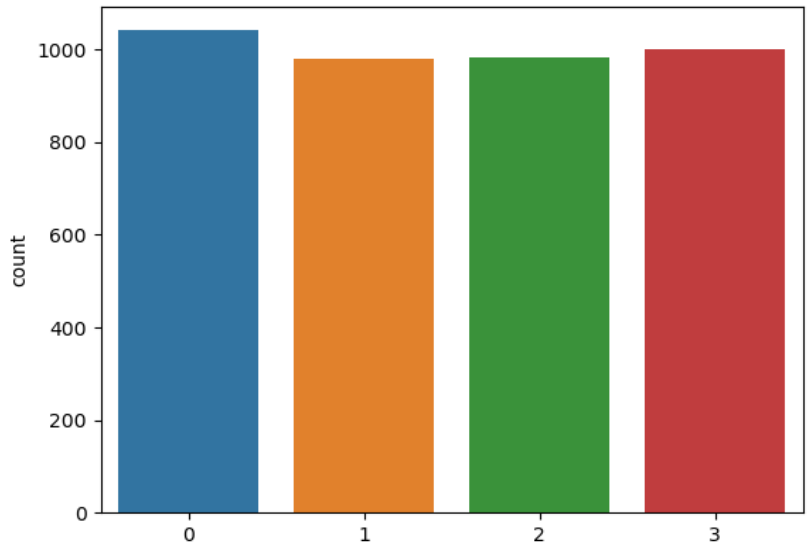
Validation Loss = 5.30 %

Confusion Matrix (Evaluation):



Zero One Loss → Only 68 from 4000 images
There are 3932 from 4000 images that
our model predicted correctly.

Distribution of Predicting Data:



Chapter 5: Mobile Application, Project Planning and Analysis

5.1 Project Planning

Why do we use Flutter?

1. **Cross-Platform Development:** Write dart code once, deploy on iOS, Android, web, and desktop. Single codebase saves time and effort.

2. **Performance:** High-performance Flutter compiles to native ARM code, no intermediate layer. Faster execution, smoother user experience.

3. **Strong Community and Ecosystem:** Rapidly growing community provides resources, tutorials, and packages. Ensures framework stays updated from Google.

4. **Rich Set of Widgets:** Comprehensive, customizable widgets and high flexibility in design.

5. **Hot-Reload:** Shows instant code changes, speeding up development for experimentation.

5.1.1 Feasibility Study

Brain tumors represent one of the most critical challenges facing the global population, impacting millions of lives with profound medical, emotional, and societal consequences.

With an increasing incidence worldwide, brain tumors affect individuals of all ages, from children to the elderly. Statistics reveal a rising trend, highlighting the urgent need for heightened awareness, research, and support.

So, our system supports people and the patients with brain tumor or not and how to get the right treatment.

The system also provides people with all information like (Display medical analysis values and their range (patient status, brain cancer type, accuracy of MRI brain cancer) uploading

MRI scans of a specific patient, medications required to treat and some advice to keep the patient healthy by chat with doctor) and he can do it by entering his MRI Brain.

Cost: The application has no cost because it is completely free, making the people know he is infected or not and the right treatments. Without paying any cost.

Accessibility: Users will be able to access the application anytime and there will be no time constraints.

Communication: The application provides an easy way of communication via chat between the user and the doctor.

Time: Save users time instead of Spending a lot of time at the Doctor's clinic. The user can use the app from home or which place without waste in time.

Awareness: Spread awareness of information and correct treatment that may be unknown to users and clear the fear among users of using medical programs that help patients.

Emergency: Able to serve many users in emergency situations for finding a suitable treatment or visiting the doctor in serious cases

5.1.2 Estimated Cost

Hardware Materials:
Personal laptop or personal computer or phone connected to the internet.

Software Materials (Technologies and Tools):

- Python Language
- Jupyter Notebook
- Colab Notebook
- Flask Framework
- Dart Language
- Flutter Framework
- Firebase & Database
- Android Studio & VS Code

5.2 Analysis and Limitation of Existing System

Time: If any user wants to use the app to check about his MRI, is he has Brain Tumor or not it takes few second but with limited number of users.

Limitation of Data: In the future we will make the scaling of application larger to handle many different users.

5.3 Need for the New System

Many techniques have been used to estimate and identify the presence of Brain Tumor Unfortunately is not accurate and some of them are costly and take much time in process can also lead to false negatives so the medical scantest used.

MRI is more accurate, but doctors can’t observe the small change in the MRI images by eyes, so we need machine learning and deep learning.

Deep learning algorithms such as the convolutional neural network (CNN) can automatically process large amounts of medical images and identify complex associations in high-dimensional data for disease diagnosis and treatment planning.

5.4 Analysis of the New System

5.4.1 User Requirements

Users wants to know he is infected by Brain Tumor or not, and that is by uploading some analysis and MRI scan the patient's own as a basic requirement, and then he gets the right treatment for this patient and some advice and information to keep him healthy.

Understanding the needs and expectations of the system's users is vital for designing a user-centric solution. In this section, we identify and analyze the specific requirements and preferences of different user groups, such as patients, doctor, and admin.

Through user interviews, surveys, and usability tests, we gather valuable insights into their needs, expectations, and desired functionalities, which serve as the basis for designing a system that effectively caters to these requirements.

5.4.2 System Requirements

What does your computer need in software and hardware?

Of course, the operating system, and since our application code will be written in many languages, frameworks, and tools. As we referred to them previously.

5.4.3 Domain Requirements

The application must verify all values (in Real-Time) before making the change in the database. application must have updated capabilities for future models and maintenance. database should be backed up every once to avoid case the original becoming corrupt or attacked.

Design & Implementation Constraints:

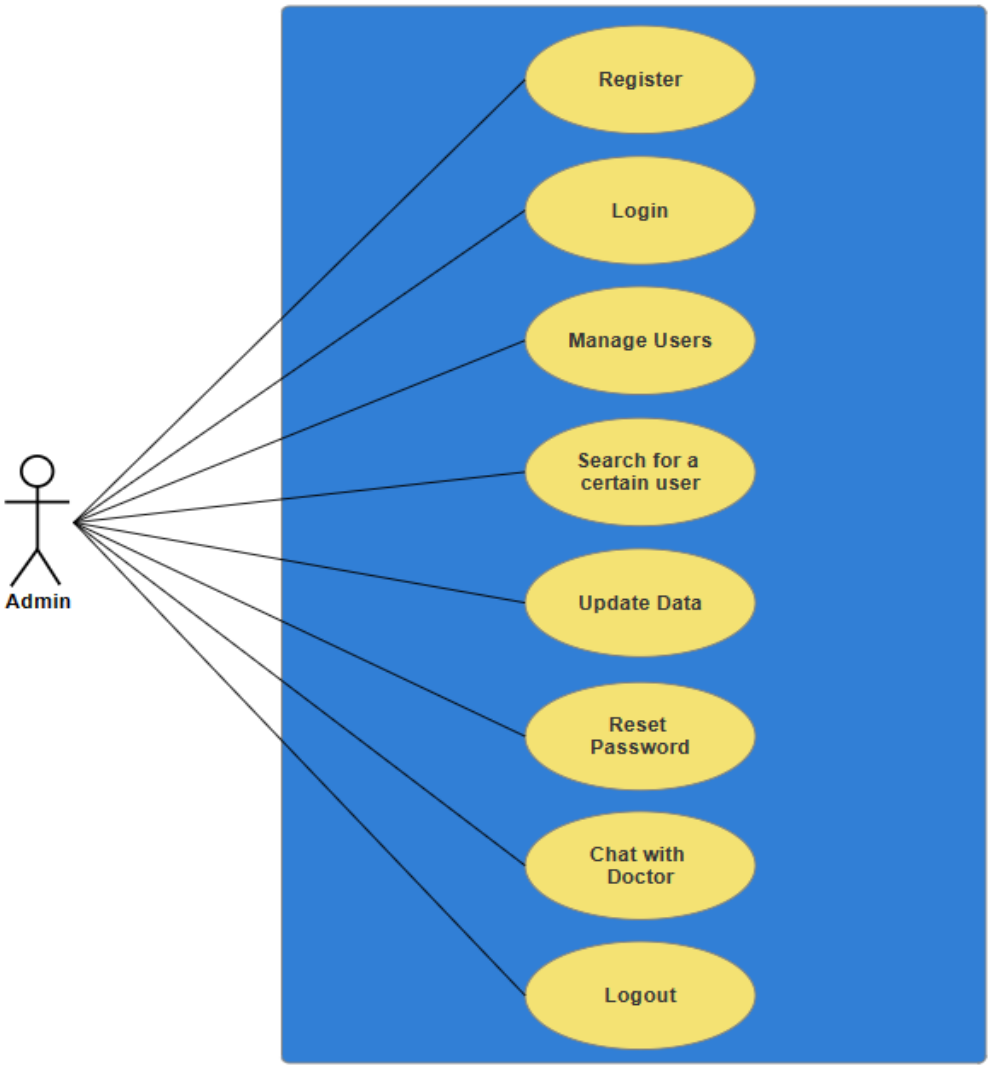
- Username: String Password: String Email: String
- Medical Analysis: double & String Scans: Images

5.4.4 Functional Requirements

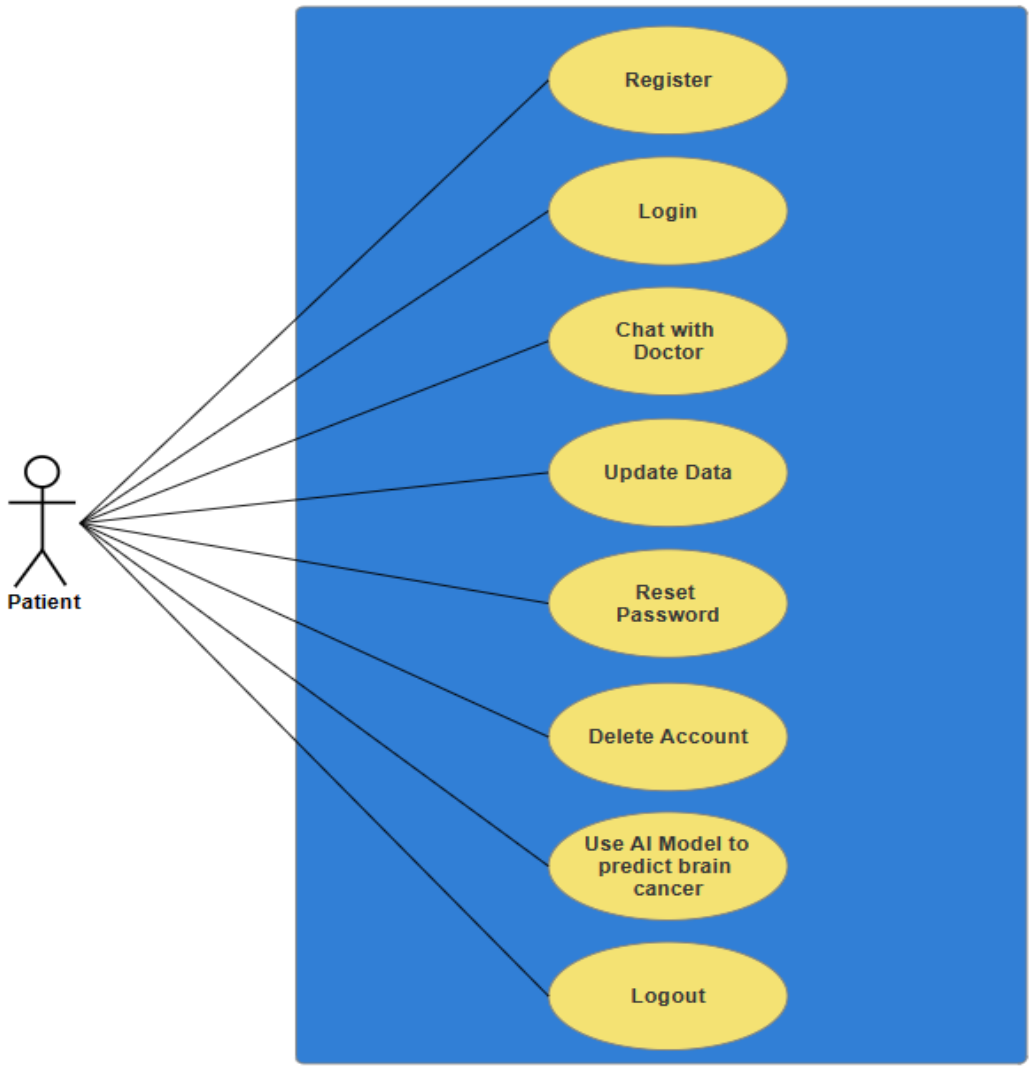
Functional requirements specify the specific features and functionalities that the system should possess to meet the needs of its users such as (Registration, Login, Verifications, Reset-Password, Update Data, Delete Account, and Predict Brain Cancer MRI By AI). This includes the ability to process images input, return the results from model, save the user data in databases, system should have a backend system to handle Firebase, Flutter UI, API, and deep learning model together, and API controller that facilitates seamless communication among different components of the system, ensuring efficient data flow and interaction.

Use Cases Diagrams

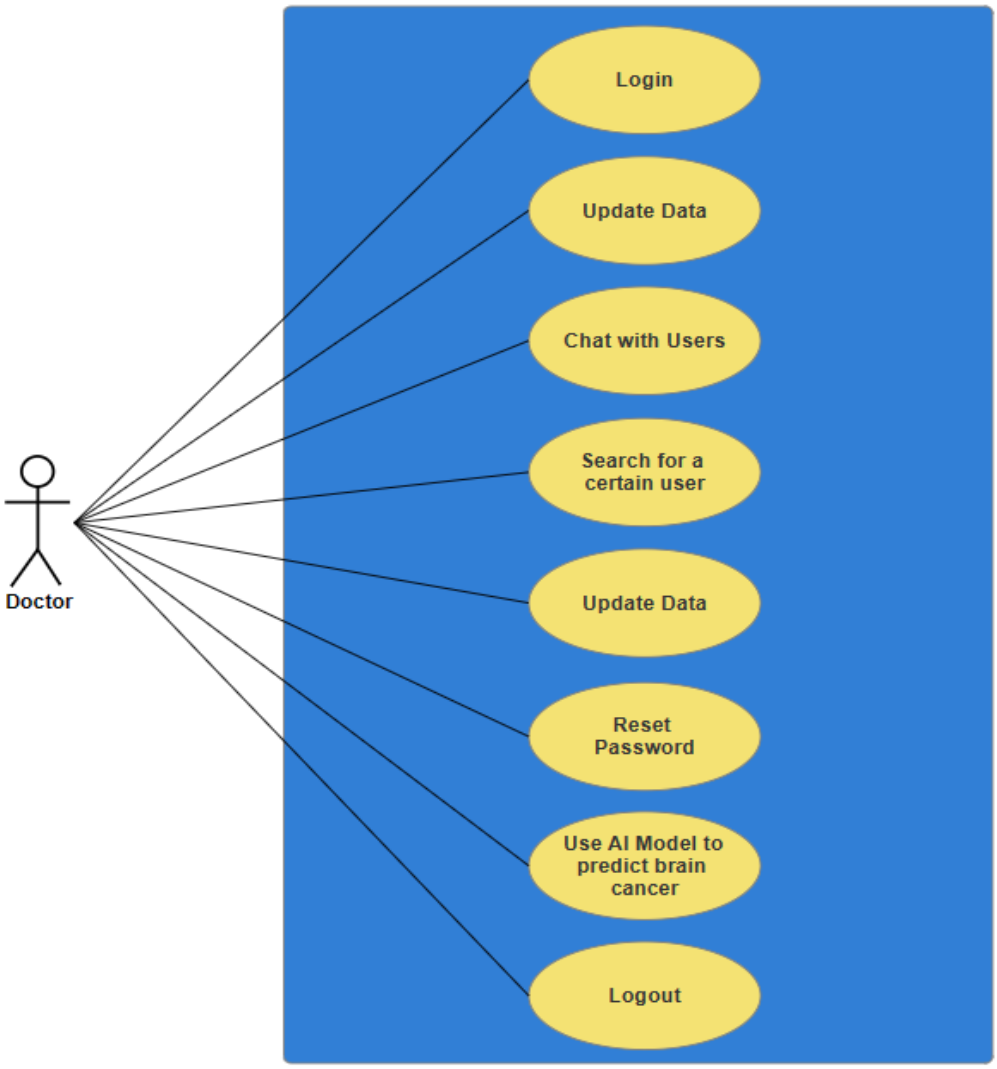
- **Admin Use Case Diagram:**



- **Patient Use Case Diagram:**



- **Doctor Use Case Diagram:**



5.4.5 Non-Functional Requirement

- **Usability:**
  - Application will be easy to use & Application will have a user-friendly interface.

- **Performance:**
  - Defines how fast a system can respond to a particular user's action under a certain workload & Application will be fast and robust.

- **Availability:**
  - Our application can be used in any time and it always “uptime” is the amount of time that it is operational and available for use**.**

- **Reliability:**
  - Is the extent to which the app consistently performs the specified functions without failure?
  - Application will not produce incorrect output.

- **Portability:**
  - Is the ease with which the app can be moved from its current environment to another environment.
  - Application will work in different platforms (Android - IOS).

- **Flexibility:**
  - Is the ease in which the app can be modified to adapt to different environments, configurations, or user expectations?

- **Maintainability:**
  - Application support modifying.
  - Is the ease with which faults in the app can be found and repaired?

- **Ethical:**
  - Application will be license free.

- **Privacy:**
  - Personal info of the registered user will only be accessed by the administrator.

5.5 Advantages of the New System

Some of Features:

- Registration & Login & Logout
- Google Authentication
- Facebook Authentication
- Email Verification & Data Validation
- Chatting with doctor
- Searching for certain user
- Reset Password
- Change Profile Data
- Manage Users (Add & Delete)
- Use AI to predict brain cancer MRI
- Contact with developer

5.6 Risk and Risk Managements

Identifying and managing potential risks are essential for project success. We have conducted a comprehensive risk assessment to identify potential risks and challenges that may arise during the project's development and implementation. We have categorized the risks based on their impact and likelihood and developed a risk management plan to mitigate these risks effectively. By proactively addressing the potential challenges, we ensure smoother and more successful project execution.

Nothing beats personalized advice and treatment from your doctor while you can help identify possible medical conditions by researching your symptoms online, your doctors (and agencies such as the Social Security Administration) will never base your treatment plan or eligibility for disability assistance on a self-diagnosis. When you treat with a medical professional, he or she bases your diagnoses and treatment plans on: Your reported symptoms, objective clinical findings (such as information clinical examinations), available medical research and guidelines, and knowledge about your medical history and lifestyle.

Warning: If the internet connection is weak or unstable the MRI pictures will not load on the app and the server could fall off.

5.7 Design of Database (Firebase)

We chose Firebase because of its power, its many features, and its compatibility with Flutter and it is strongly supported by Google such as Flutter.

Some of Features: (Authentication, Cloud Fire-store, Storage Images, Verification, Reset Password)

The project requires Three Collections of (Firebase) database to store relevant data: Users Database, First Admin Registered, and Chats.

1. **Users Collection:**

   Purpose: The Users Database is intended to store user-related information such as usernames, email, passwords, User-Role, and UID.

   Fields:
   • Email: Stores the email of the user.
   • Username: Stores the username of the user.
   • Password: The system makes sure that the password is strong enough.
   • User-Role: Indicates the user type (Patient, Doctor, or Admin).
   • UID: The Unique number that is generated when the user registers in the system.

   Functionality: The Users Database maintains user credentials and type information, facilitating secure user authentication and distinguishing between patients, doctor, and admin.

2. **First Admin Registered Collection:**

   Purpose: The First Admin Registered Database is intended to store admin-related information such as email.

   Fields:
   • Email: Stores the email of the admin.

   Functionality: We have customized the system to approve only one administrator via e-mail and verify this e-mail if it is an administrator or a regular

user. Only admins are allowed to enter the application designated for it and vice versa.

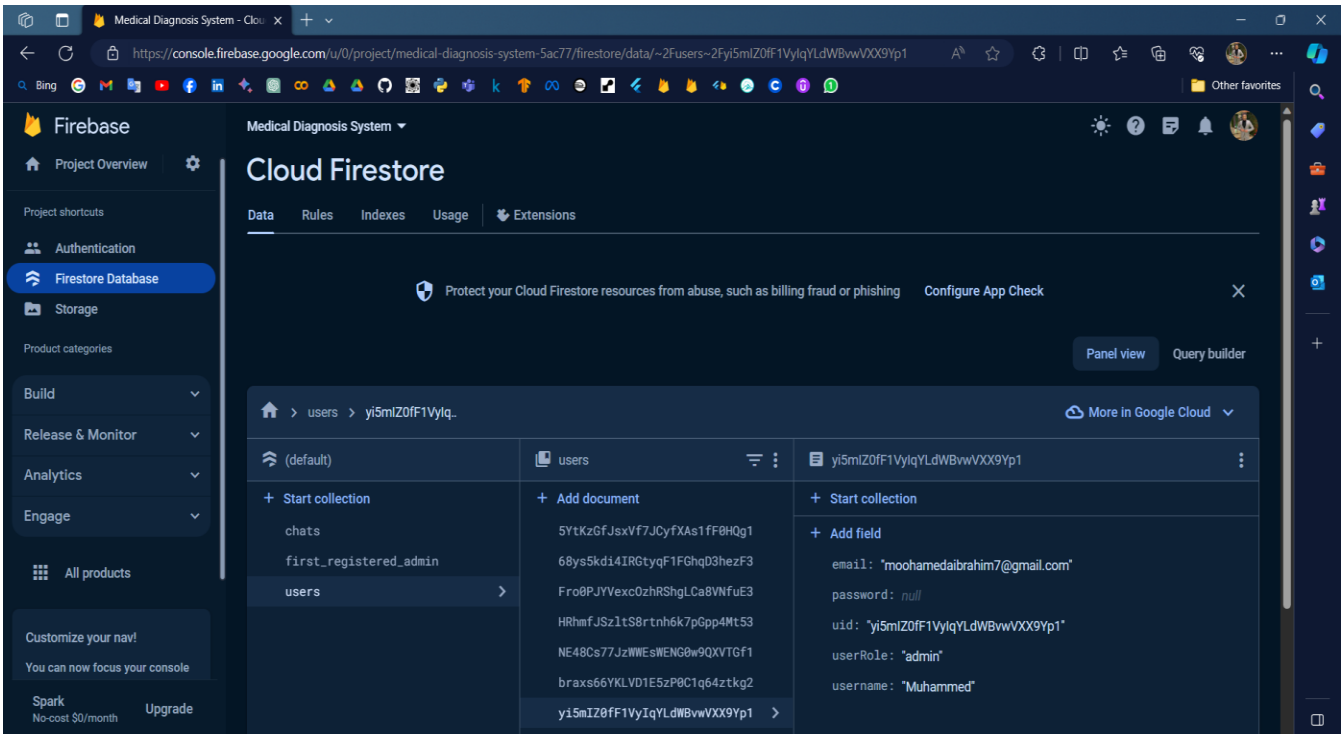
3. Chats Collection:

Purpose: The Chats Database is intended to store chat-related information such as chat-id, created-at, and text.

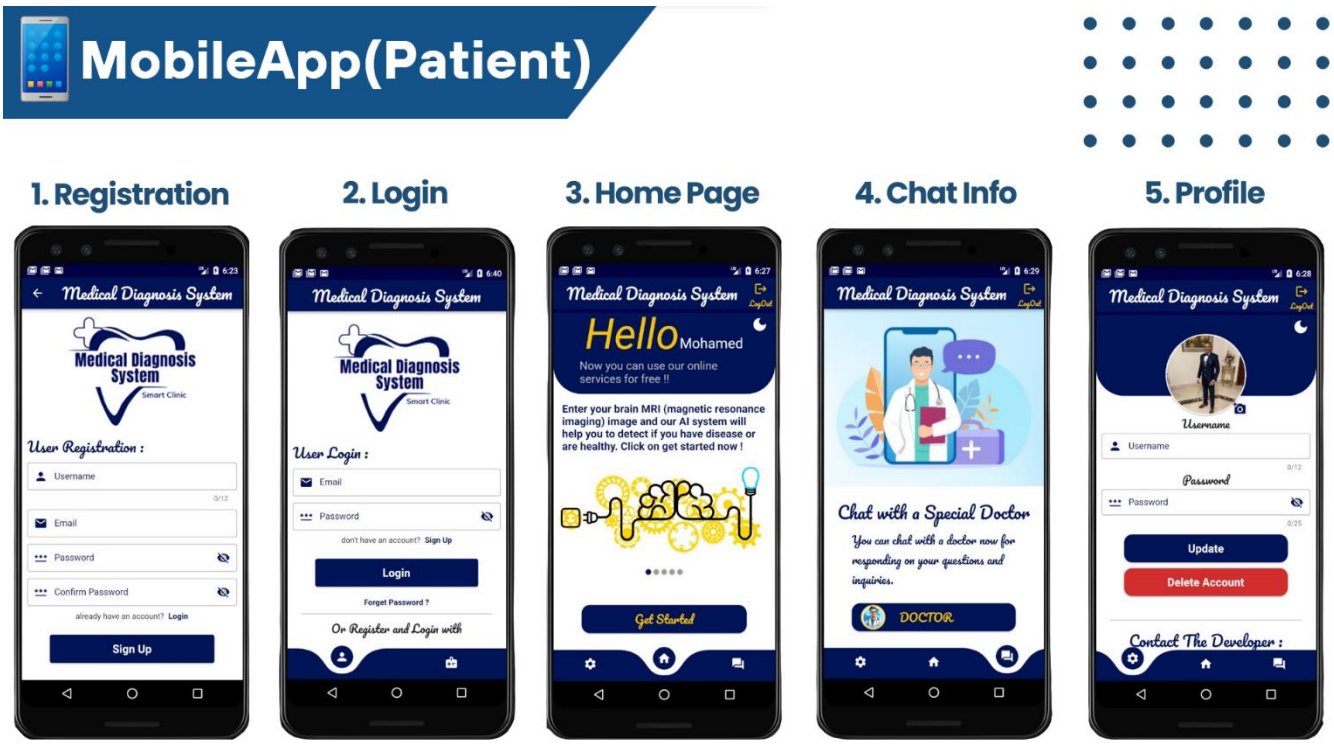
Fields:

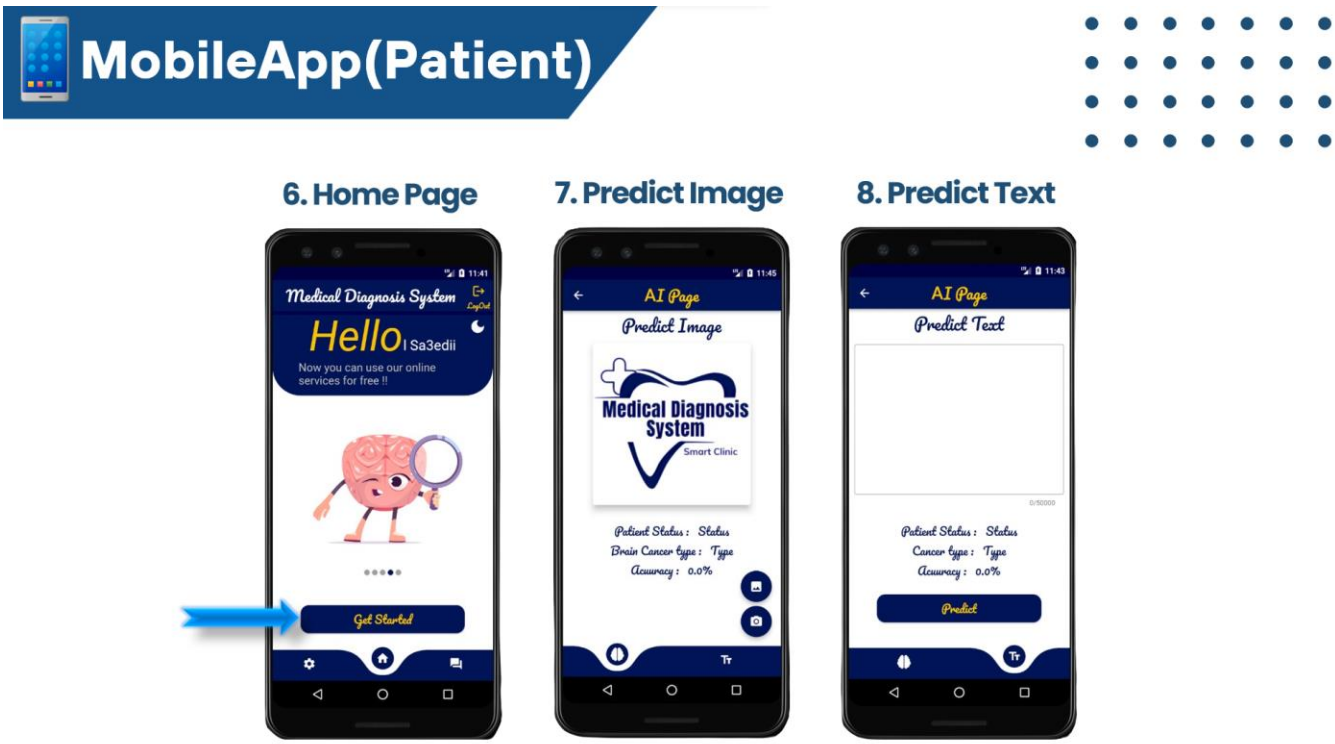
- Chat-Id: Stores the chat-id (email of the user).
- Created-At: The time the message was created.
- Text: The content of message.

Functionality: To handle multi chatting between doctor and users we create a collection contain some of document each of these documents contain other collections and so on (Nested Collections & Documents).



5.8 User Interface





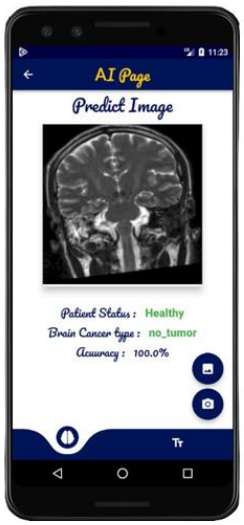
MobileApp(Doctor)

1. Login

2. Chat Page

3. Profile

4. AI Page

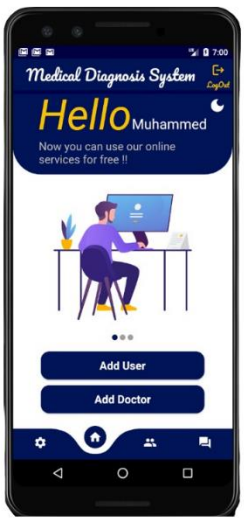


MobileApp(Admin)

1. Registration

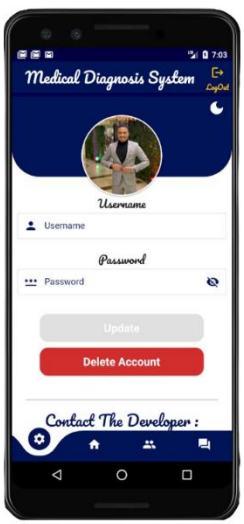
2. Login

3. Home Page



4. Manage Users

5. Profile



MobileApp(Admin)

6. Add User

7. Add Doctor

8. Users Info

MobileApp(Chats)

1. User & Doctor

2. Doctor & User

3. Admin & Doctor

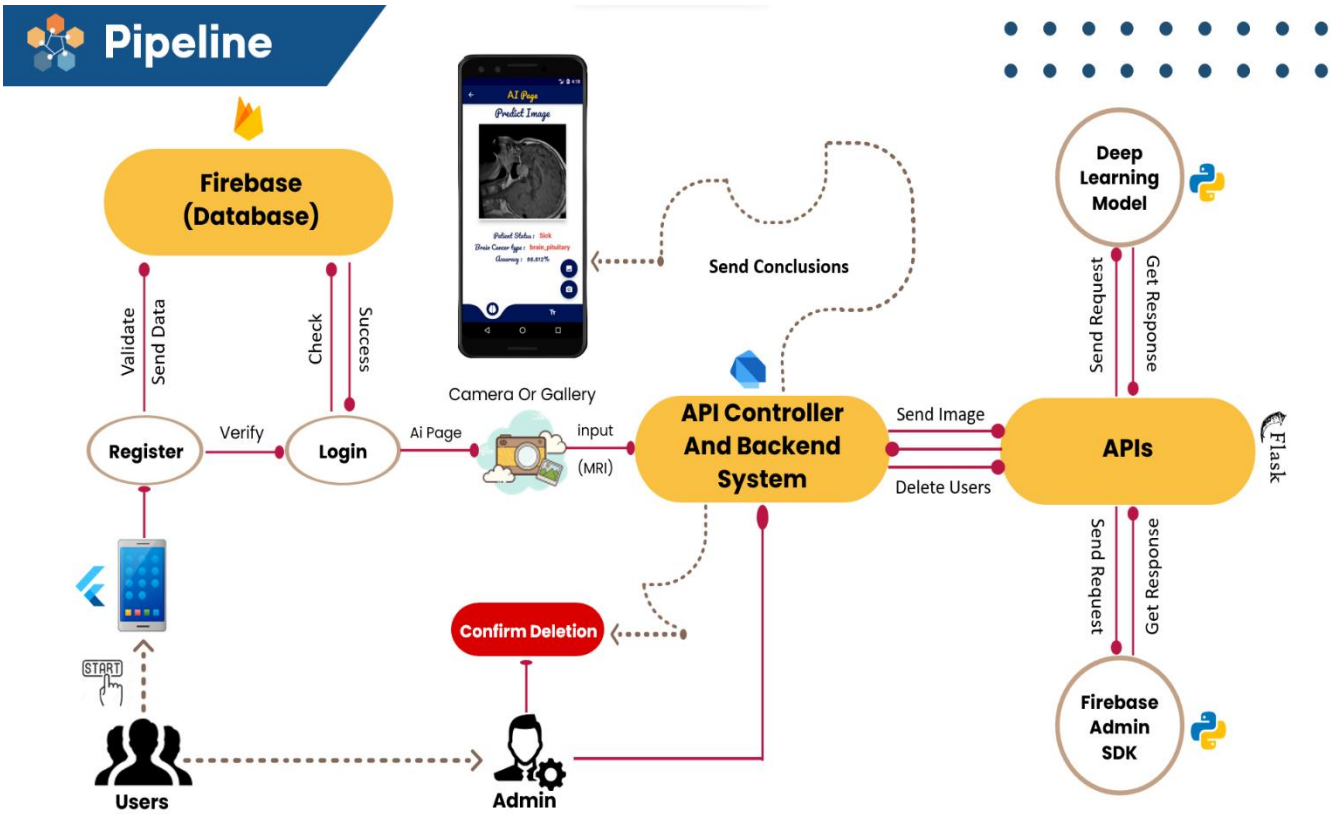
5.9 Workflow

Workflow is a series of tasks or activities that are performed to accomplish a specific goal. It defines the sequence of steps and the people or systems responsible for each step.

Pipeline is a set of processes or steps that are executed in a specific order to achieve a desired outcome. pipeline often involves a series of workflows, where each workflow represents a set of tasks or activities.

Dataflow refers to the movement of data from one point to another within a system or process. It illustrates how data is processed, transformed, and transferred through different stages. data flows through these workflows or pipelines, undergoing transformations or manipulations at each step.

In summary, a pipeline is the overarching process, workflows are the individual sets of tasks, and data flow is the movement of data through these tasks to achieve a specific goal.



5.10 Future Work

- Make the scaling of application is larger to handle many different of users.
- Make an application version for the web to make it easier for users to use our web app.
- Add Video call for users and doctors.
- Support different languages to gain more audience from all over the world.
- Activate Facebook authentication.
- Add dark mode theme to our application.
- Train another model on object detection techniques to detect the Infected cells.
- Train another model on NLP techniques for predicting brain prescription texts.

5.11 References (Bibliography)

i. Andrew NG: https://www.deeplearning.ai/

ii. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10372320/

iii. https://braintumor.org/brain-tumors/about-brain-tumors/brain-tumor-facts/

iv. CS230 Deep Learning (stanford.edu)

v. Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network | upGrad blog

vi. Alex-net explanation and implementation in TensorFlow and Keras | by Syed Sajjad Askari | Medium

vii. VGGNet-16 Architecture: A Complete Guide | Kaggle

viii. Residual neural network - Wikipedia

ix. Batch Normalization in Convolutional Neural Networks | Baeldung on Computer Science

x. Artificial Intelligence in Brain Tumor Imaging: A Step toward Personalized Medicine - PMC (nih.gov)

xi. What is the purpose of image preprocessing in deep learning? (isahit.com)

xii. Paper 1: Brain-Tumor-Classification-Using-Convolutional-Neural-Network.pdf (researchgate.net)

xiii. Paper 2: International Journal of Imaging Systems and Technology | IMA | Wiley Online Library

xiv. Datasets:

- https://www.kaggle.com/datasets/obulisainaren/multi-cancer
- https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset
- https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection
- https://www.kaggle.com/datasets/pradeep2665/brain-mri

xv. Flutter documentation | Flutter

xvi. Firebase Documentation (google.com)

Thank You!