**NEUROFUSION ADVANCING BRAIN TUMOR DIAGNOSIS**

**USING CNN ARCHITECTURES**

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **TITLE** | **PAGE.NO** |
| **01** | **ABSTRACT** |  |
| **02** | **EXISTING SYSTEM**  2.1 DRAWBACKS |  |
| **03** | **INTRODUCTION**  3.1 DATA SCIENCE  3.2 ARTIFICIAL INTELLIGENCE |  |
| **04** | **MACHINE LEARNING** |  |
| **05** | **PREPARING DATASET** |  |
| **06** | **PROPOSED SYSTEM**  6.1 ADVANTAGES |  |
| **07** | **LITERATURE SURVEY** |  |
| **08** | **SYSTEM STUDY**  8.1 OBJECTIVES  8.2 PROJECT GOAL  8.3 SCOPE OF THE PROJECT |  |
| **09** | **FEASIBILITY STUDY** |  |
| **10** | **LIST OF MODULES** |  |
| **11** | **PROJECT REQUIREMENTS**  11.1 FUNCTIONAL REQUIREMENTS  11.2 NON-FUNCTIONAL REQUIREMENTS |  |
| **12** | **ENVIRONMENT REQUIREMENT** |  |
| **13** | **SOFTWARE DESCRIPTION**  13.1 ANACONDA NAVIGATOR  13.2 JUPYTER NOTEBOOK |  |
| **14** | **PYTHON** |  |
| **15** | **SYSTEM ARCHITECTURE** |  |
| **16** | **WORKFLOW DIAGRAM** |  |
| **17** | **USECASE DIAGRAM** |  |
| **18** | **CLASS DIAGRAM** |  |
| **19** | **ACTIVITY DIAGRAM** |  |
| **20** | **SEQUENCE DIAGRAM** |  |
| **21** | **ER – DIAGRAM** |  |
| **22** | **MODULE DESCRIPTION**  22.1 MODULE DIAGRAM  22.2 MODULE GIVEN INPUT EXPECTED OUTPUT |  |
| **23** | **DEPLOYMENT** |  |
| **24** | **HTML** |  |
| **25** | **CSS** |  |
| **26** | **CODING** |  |
| **27** | **CONCLUSION** |  |
| **28** | **FUTURE WORK** |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **TITLE** | **PAGE.NO** |
| **01** | **SYSTEM ARCHITECTURE** |  |
| **02** | **WORKFLOW DIAGRAM** |  |
| **03** | **USECASE DIAGRAM** |  |
| **04** | **CLASS DIAGRAM** |  |
| **05** | **ACTIVITY DIAGRAM** |  |
| **06** | **SEQUENCE DIAGRAM** |  |
| **07** | **ER – DIAGRAM** |  |
| **08** | **MODULE DIAGRAM** |  |

**LIST OF SYMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | nAME  Class B  Class A    Class B  Class A | Associations represents static relationships between classes. Roles representsthe way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | EXTENDS | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processs. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Usecase |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which is a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or acion. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**Abstract:**

Brain tumor diagnosis is a critical area in medical imaging, and the use of Convolutional Neural Networks (CNNs) with TensorFlow has shown significant promise in enhancing the accuracy and efficiency of this process. CNNs, known for their powerful feature extraction and classification capabilities, can automatically identify complex patterns in medical images that are indicative of brain tumors. By leveraging TensorFlow, an open-source machine learning framework, these CNN architectures can be trained on vast datasets of MRI scans, learning to distinguish between different types of brain tumors with high precision. This approach not only reduces the reliance on manual interpretation by radiologists but also accelerates the diagnostic process, leading to quicker and potentially life-saving treatment decisions. The integration of CNNs with TensorFlow in brain tumor diagnosis exemplifies the transformative impact of artificial intelligence in healthcare, offering a robust, scalable, and efficient solution for early detection and improved patient outcomes.

**Existing System:**

Tumor growth models have the potential to model and predict the spatiotemporal evolution of glioma in individual patients. Infiltration of glioma cells is known to be faster along the white matter tracts, and therefore structural magnetic resonance imaging (MRI) and diffusion tensor imaging (DTI) can be used to inform the model. However, applying and evaluating growth models in real patient data is challenging. In this work, we propose to formulate the problem of tumor growth as a ranking problem, as opposed to a segmentation problem, and use the average precision (AP) as a performance metric. This enables an evaluation of the spatial pattern that does not require a volume cut-off value. Using the AP metric, we evaluate diffusion-proliferation models informed by structural MRI and DTI, after tumor resection. We applied the models to a unique longitudinal dataset of 14 patients with low-grade glioma (LGG), who received no treatment after surgical resection, to predict the recurrent tumor shape after tumor resection. The diffusion models informed by structural MRI and DTI showed a small but significant increase in predictive performance with respect to homogeneous isotropic diffusion, and the DTI-informed model reached the best predictive performance. We conclude there is a significant improvement in the prediction of the recurrent tumor shape when using a DTI-informed anisotropic diffusion model with respect to istropic diffusion, and that the AP is a suitable metric to evaluate these models. All code and data used in this publication are made publicly available .

**Drawback:**

* They use only one architecture.
* High time complexity.
* They are not use Django framework deployment.

**INTRODUCTION:**

NeuroFusion is at the forefront of advancing brain tumor diagnosis through the integration of Convolutional Neural Network (CNN) architectures. As brain tumors continue to present complex challenges in early detection and accurate diagnosis, traditional methods often fall short in terms of precision and efficiency. NeuroFusion leverages the powerful capabilities of CNNs, a class of deep learning models renowned for their proficiency in image analysis, to enhance diagnostic accuracy. By employing sophisticated CNN architectures, NeuroFusion aims to transform the diagnostic process, offering a more reliable and nuanced approach to identifying and classifying brain tumors. This innovative methodology promises not only to improve early detection rates but also to facilitate more tailored treatment plans, ultimately contributing to better patient outcomes and advancing the field of neuro-oncology.

**Domain overview:**

**Data Science:**

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammer bacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

**Data Scientist:**

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

**Required Skills for a Data Scientist:**

* **Programming**: Python, SQL, Scala, Java, R, MATLAB.
* **Machine Learning**: Natural Language Processing, Classification, Clustering.
* **Data Visualization**: Tableau, SAS, D3.js, Python, Java, R libraries.
* **Big data platforms**: MongoDB, Oracle, Microsoft Azure, Cloudera.

**ARTIFICIAL INTELLIGENCE**:

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is [intelligence](https://en.wikipedia.org/wiki/Intelligence) demonstrated by [machines](https://en.wikipedia.org/wiki/Machine), as opposed to the natural intelligence [displayed by humans](https://en.wikipedia.org/wiki/Human_intelligence) or [animals](https://en.wikipedia.org/wiki/Animal_cognition). Leading AI textbooks define the field as the study of “[intelligent agents](https://en.wikipedia.org/wiki/Intelligent_agent)” any system that perceives its environment and takes actions that maximize its chance of achieving its goals.

Some popular accounts use the term “artificial intelligence” to describe machines that mimic “cognitive” functions that humans associate with the [human mind](https://en.wikipedia.org/wiki/Human_mind), such as “learning” and “problem solving”, however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require “intelligence” are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an “AI winter”), followed by new approaches, success and renewed funding.

AI research has tried and discarded many different approaches during its lifetime, including simulating the covid, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field’s long-term goals.

To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence “can be so precisely described that a machine can be made to simulate it”. This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence.

These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning.

AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states.

In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

**Learning processes.** This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

**Reasoning processes.** This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

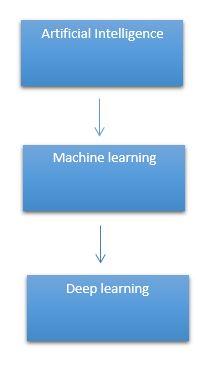
**Self-correction processes.** This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

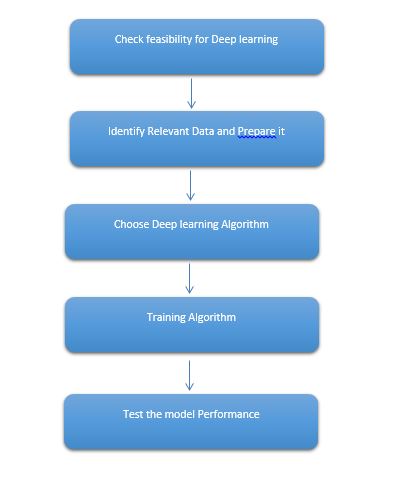
Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

**DEEP LEARNING**

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human disease so deep learning is also a kind of mimic of human disease. It’s on hype nowadays because earlier we did not have that much processing power and a lot of data. A formal definition of deep learning is- neurons Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. In disease approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbors. The question here is how it recreates these neurons in a computer. So, it creates an artificial structure called an artificial neural net where we have nodes or neurons. It has some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.



It need to identify the actual problem in order to get the right solution and it should be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). It needs to identify the relevant data which should correspond to the actual problem and should be prepared accordingly. Choose the Deep Learning Algorithm appropriately. Algorithm should be used while training the dataset. Final testing should be done on the dataset



Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological disease. Specifically, neural networks tend to be static and symbolic, while the biological disease of most living organisms is dynamic (plastic) and analogue.

The adjective "deep" in deep learning refers to the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, but that a network with a non-polynomial activation function with one hidden layer of unbounded width can. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed connectionist models, for the sake of efficiency, trainability and understandability, whence the "structured" part.

Deep learning is a class of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [algorithms](https://en.wikipedia.org/wiki/Algorithm) that uses multiple layers to progressively extract higher-level features from the raw input. For example, in [image processing](https://en.wikipedia.org/wiki/Image_processing), lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

**Interpretations:**

Deep neural networks are generally interpreted in terms of the universal approximation theorem or probabilistic inference.

The classic universal approximation theorem concerns the capacity of feed-forward neural networks with a single hidden layer of finite size to approximate continuous functions. In 1989, the first proof was published by George Cybenko for sigmoid activation functions and was generalised to feed-forward multi-layer architectures in 1991 by Kurt Hornik. Recent work also showed that universal approximation also holds for non-bounded activation functions such as the rectified linear unit.

The universal approximation theorem for deep neural networks concerns the capacity of networks with bounded width but the depth is allowed to grow proved that if the width of a deep neural network with ReLU activation is strictly larger than the input dimension, then the network can approximate any Lebesgue integrable function; If the width is smaller or equal to the input dimension, then deep neural network is not a universal approximator.

The probabilistic interpretation derives from the field of machine learning. It features inference, as well as the optimization concepts of training and testing, related to fitting and generalization, respectively. More specifically, the probabilistic interpretation considers the activation nonlinearity as a cumulative distribution function. The probabilistic interpretation led to the introduction of dropout as regularizer in neural networks. The probabilistic interpretation was introduced by researchers including Hopfield, Widrow and Narendra and popularized in surveys such as the one by Bishop.

**Deep learning revolution:**

In 2012, a team led by George E. Dahl won the "Merck Molecular Activity Challenge" using multi-task deep neural networks to predict the biomolecular target of one drug. In 2014, Hochreiter's group used deep learning to detect off-target and toxic effects of environmental chemicals in nutrients, household products and drugs and won the "Tox21 Data Challenge" of NIH, FDA and NCATS.

Significant additional impacts in image or object recognition were felt from 2011 to 2012. Although CNNs trained by back-propagation had been around for decades, and GPU implementations of NNs for years, including CNNs, fast implementations of CNNs on GPUs were needed to progress on computer vision. In 2011, this approach achieved for the first time superhuman performance in a visual pattern recognition contest. Also in 2011, it won the ICDAR Chinese handwriting contest, and in May 2012, it won the ISBI image segmentation contest. Until 2011, CNNs did not play a major role at computer vision conferences, but in June 2012, a paper by Ciresan et al. at the leading conference CVPR showed how max-pooling CNNs on GPU can dramatically improve many vision benchmark records.

In October 2012, a similar system by Krizhevsky et al. won the large-scale ImageNet competition by a significant margin over shallow machine learning methods. In November 2012, Ciresan et al.'s system also won the ICPR contest on analysis of large medical images for cancer detection, and in the following year also the MICCAI Grand Challenge on the same topic. In 2013 and 2014, the error rate on the ImageNet task using deep learning was further reduced, following a similar trend in large-scale speech recognition.

Image classification was then extended to the more challenging task of generating descriptions (captions) for images, often as a combination of CNNs and LSTMs.

Some researchers state that the October 2012 ImageNet victory anchored the start of a "deep learning revolution" that has transformed the AI industry.

In March 2019, Yoshua Bengio, Geoffrey Hinton and Yann LeCun were awarded the Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.

**PROPOSED SYSTEM:**

The proposed system for brain tumor diagnosis utilizes Convolutional Neural Network (CNN) architectures and TensorFlow to enhance the accuracy and efficiency of detecting brain tumors from MRI scans. This system integrates a Django-based web application to provide an accessible and user-friendly interface. The process begins with the collection of MRI images, which are pre-processed to ensure uniformity and improve the quality of the input data. The pre-processed images are then fed into a carefully designed CNN model, which leverages multiple convolutional layers to extract intricate features and patterns indicative of brain tumors. TensorFlow, with for building and training neural networks, is employed to develop and fine-tune the CNN model, ensuring high precision in diagnosis. The Django application acts as a bridge between the users and the backend machine learning model. It allows users to upload MRI scans, which are then analyzed by the CNN model. The results, including the probability of the presence of a tumor and its classification, are displayed on the web interface, accompanied by visualizations to aid in interpretation. This integration of advanced Deep learning techniques with a dynamic web application aims to provide a reliable, efficient, and scalable solution for brain tumor diagnosis, potentially improving early detection rates and aiding in better patient outcomes.

**ADVANTAGES:**

* We build a framework based application for deployment purposes.
* Our process require less compute power.
* We more than 4 classify
* We compared more than a two architecture to getting better accuracy level.

**LITERATURE SURVEY**

**General**

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period.

Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them. Loan default trends have been long studied from a socio-economic stand point.

Most economics surveys believe in empirical modeling of these complex systems in order to be able to predict the loan default rate for a particular individual. The use of machine learning for such tasks is a trend which it is observing now. Some of the survey’s to understand the past and present perspective of loan approval or not.

**Review of Literature Survey**

**Title** : Brain Tumor Detection Using Deep Learning Approaches.

**Author** : Razia Sultana Misu.

**Year** : 2023

Brain tumors are collections of abnormal cells that can develop into masses or clusters. Because they have the potential to infiltrate other tissues, they pose a risk to the patient. The main imaging technique used, MRI, may be able to identify a brain tumor with accuracy. The fast development of Deep Learning methods for use in computer vision applications has been facilitated by a vast amount of training data and improvements in model construction that offer better approximations in a supervised setting. The need for these approaches has been the main driver of this expansion. Deep learning methods have shown promise in improving the precision of brain tumor detection and classification using magnetic resonance imaging (MRI). The study on the use of deep learning techniques, especially ResNet50, for brain tumor identification is presented in this abstract. As a result, this study investigates the possibility of automating the detection procedure using deep learning techniques. In this study, I utilized five transfer learning models which are VGG16, VGG19, DenseNet121, ResNet50 and YOLO V4 where ResNet50 provide the best or highest accuracy 99.54%. The goal of the study is to guide researchers and medical professionals toward powerful brain tumor detecting systems by employing deep learning approaches by way of this evaluation and analysis.

**Title**: A Comprehensive Survey on Brain Tumor Diagnosis Using Deep Learning and Emerging Hybrid Techniques with Multi-modal MR Image.

**Author**: Saqib Ali, Jianqiang Li, Yan Pei, Rooha Khurram, Khalil ur Rehman, Tariq Mahmood.

**Year** : 2021

The brain tumor is considered the deadly disease of the century. At present, neuroscience and artifcial intelligence conspire in the timely delineation, detection, and classifcation of brain tumors. The process of manually classifying and segmenting many volumes of MRI scans is a challenging and laborious task. Therefore, there is an essential requirement to build computer-aided diagnosis systems to diagnose brain tumors timely. Herein review focuses on the advances of the last decade in brain tumor segmentation, feature extraction, and classifcation through powerful and versatile brain imaging modality Magnetic Resonance Imaging (MRI). However, particular emphasis on deep learning and hybrid techniques. We have summarized the work of researchers published in the last decade (2010–2019) termed as the 10s and the present decade (only including the year 2020) termed as the 20s. The decades in review reveal the bore witness to the critical revolutionary paradigm shift in artifcial intelligence viz. conventional/machine learning methods, emerged deep learning, and emerging hybrid techniques. This review also covers some persistent concerns on using the type of classifer and striking trends in commonly employed MRI modalities for brain tumor diagnosis. Moreover, this study ensures the limitation, solutions, and future trends or opens up the researchers’ advanced challenges to develop an efcient system exhibiting clinically acceptable accuracy that assists the radiologists for the brain tumor prognosis.

**Title** : Brain Tumor Detection Using Deep Learning.

**Author**: Dheiver Santos, Ewerton Santos.

**Year** : 2022

A brain tumor is understood by the scientific community as the growth of abnormal cells in the brain, some of which can lead to cancer. The traditional method to detect brain tumors is nuclear magnetic resonance (MRI). Having the MRI images, information about the uncontrolled growth of tissue in the brain is identified. In several research articles, brain tumor detection is done through the application of Machine Learning and Deep Learning algorithms. When these systems are applied to MRI images, brain tumor prediction is done very quickly and greater accuracy helps to deliver treatment to patients. These predictions also help the radiologist to make quick decisions. In the proposed work, a set of Artificial Neural Networks (ANN) are applied in the detection of the presence of brain tumor, and its performance is analyzed through different metrics.

**Title** : Brain Tumor Classification Using Deep Learning Technique - A Comparison between Cropped, Uncropped, and Segmented Lesion Images with Different Sizes.

**Author**: Ali Mohammad Alqudah1 , Hiam Alquraan , Isam Abu Qasmieh , Amin Alqudah , Wafaa Al-Sharu.

**Year** : 2019

Deep Learning is the newest and the current trend of the machine learning field that paid a lot of the researchers' attention in the recent few years. As a proven powerful machine learning tool, deep learning was widely used in several applications for solving various complex problems that require extremely high accuracy and sensitivity, particularly in the medical field. In general, the brain tumor is one of the most common and aggressive malignant tumor diseases which is leading to a very short expected life if it is diagnosed at a higher grade. Based on that, brain tumor grading is a very critical step after detecting the tumor in order to achieve an effective treating plan. In this paper, we used Convolutional Neural Network (CNN) which is one of the most widely used deep learning architectures for classifying a dataset of 3064 T1 weighted contrast-enhanced brain MR images for grading (classifying) the brain tumors into three classes (Glioma, Meningioma, and Pituitary Tumor). The proposed CNN classifier is a powerful tool and its overall performance with an accuracy of 98.93% and sensitivity of 98.18% for the cropped lesions, while the results for the uncropped lesions are 99% accuracy and 98.52% sensitivity and the results for segmented lesion images are 97.62% for accuracy and 97.40% sensitivity.

**Title** : Brain Tumor Detection and Classification from Multi-Channel MRIs using Deep Learning and Transfer Learning.

**Author**: Subhashis Banerjee.

**Year** : 2021

Glioblastoma Multiforme constitutes 80% of malignant primary brain tumors in adults, and is usually classified as High Grade Glioma (HGG) and Low Grade Glioma (LGG). LGG tumors are less aggressive, with slower growth rate as compared to HGG, and are responsive to therapy. Tumor biopsy being challenging for brain tumor patients, noninvasive imaging techniques like Magnetic Resonance Imaging (MRI) have been extensively employed in diagnosing brain tumors. Therefore, development of automated systems for the detection and prediction of the grade of tumors based on MRI data become necessary. In this paper, we investigate Deep Convolutional Neural Networks (ConvNets) for classification of brain tumors using multisequence MR images. We propose three ConvNets, which are trained from scratch, on MRI patches, slices, and multi-planar volumetric slices. The suitability of transfer learning for the task is also studied by applying two existing ConvNets models (VGGNet and ResNet) trained on ImageNet dataset, through fine-tuning of the last few layers. Leave-one-patient-out (LOPO) testing scheme is used to evaluate the performance of the ConvNets. Results demonstrate that ConvNet achieves better accuracy in all cases where the model is trained on the multi-planar volumetric dataset. It obtains a testing accuracy of 97% without any additional effort towards extraction and selection of features, as required in conventional models. We also compare our results with stateof-the-art methods that require manual feature engineering for the task. It shows a maximum improvement of 12% on grading performance of ConvNets. We also study the properties of selflearned kernels/filters in different layers through visualization of the intermediate layers outputs.

**SYSTEM STUDY:**

Project Goals:

Load the data

* Loading the given dataset
* Import required libraries packages

Pre-process the data

* Reshape, data augmentations

Define model

* Sequential or Functional
* Number of layers to be used, Number of nodes to be used in the model, Evaluation metrics

Compile the model

* Define loss function, optimizer, weights and bias

Fit the model

* Train data, Test data, epoch, Batch size.

**Aim:**

The aim of the "Neurofusion Advancing Brain Tumor Diagnosis Using CNN Architectures" project is to enhance the accuracy and efficiency of brain tumor diagnosis through the application of Convolutional Neural Networks (CNNs). By leveraging advanced CNN architectures, the project seeks to develop a robust and reliable diagnostic tool that can analyze medical imaging data, such as MRI scans, with high precision. The goal is to improve early detection, classification, and prognosis of brain tumors, thereby aiding healthcare professionals in making informed decisions and ultimately improving patient outcomes. The integration of cutting-edge deep learning techniques aims to overcome the limitations of traditional diagnostic methods and provide a more effective solution for brain tumor diagnosis.

**Objectives:**

The primary objective of the "Neurofusion Advancing Brain Tumor Diagnosis Using CNN Architectures" project is to enhance the accuracy and efficiency of brain tumor diagnosis through the implementation of advanced Convolutional Neural Network (CNN) architectures. This initiative aims to leverage the power of deep learning to analyze medical imaging data, including MRI and CT scans, to identify and classify brain tumors with high precision. By developing and training sophisticated CNN models, the project seeks to improve early detection rates, reduce diagnostic errors, and facilitate timely intervention for patients. Additionally, the project intends to integrate these CNN models into a user-friendly diagnostic tool that can be utilized by medical professionals, ultimately advancing the field of neuro-oncology and contributing to better patient outcomes.

**Scope:**

The "NeuroFusion Advancing Brain Tumor Diagnosis Using CNN Architectures" project aims to revolutionize the field of neurology by leveraging Convolutional Neural Networks (CNNs) to enhance the accuracy and efficiency of brain tumor diagnosis. This project focuses on developing a sophisticated deep learning model that can analyze medical imaging data, such as MRI scans, with unprecedented precision. By integrating advanced CNN architectures, the project seeks to improve tumor detection rates, classify different tumor types, and predict tumor progression. The scope includes designing and training a CNN model tailored to identify subtle patterns and anomalies in brain images, refining the model's performance through rigorous testing, and implementing it within a user-friendly diagnostic tool. Ultimately, the project aims to support medical professionals in making more informed decisions, reduce diagnostic errors, and contribute to better patient outcomes in the fight against brain tumors.

**OUTLINE OF THE PROJECT**

**Overview of the system:**

* Define a problem
* Gathering image data set
* Evaluating algorithms
* Detecting results

The steps involved in Building the data model is depicted below.

**Data collection** (Splitting Training set & Test) set)

**Pre Processing** (Sequential)

**Building classification Model**

**Detection(**Brain Tumor Diagnosis**)**

Fig: data flow diagram for CNN model

**PROJECT REQUIREMENTS**

**General:**

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements

2. Non-Functional requirements

3. Environment requirements

A. Hardware requirements

B. software requirements

**Functional requirements:**

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like tensorflow, keras, matplotlib.

**Non-Functional Requirements:**

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithm
4. Improving results
5. Prediction the result

**Environment Requirements:**

**Framework :** Keras.

**Software Requirements:**

* Operating System : Windows / Linux
* Simulation Tool : Anaconda with Jupyter Notebook
* Language : Python

**Hardware requirements:**

* Processor : Intel i3
* Hard disk : minimum 400 GB
* RAM : minimum 4 GB

**FEASIBILITY STUDY**

**Splitting the dataset:**

The data use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. It has the test dataset (or subset) in order to test our models and it will do this using Tensor flow library in Python using the Keras method.

**Construction of a Detecting Model:**

## Deep learning needs data gathering have lot of past image data’s. Training and testing this model working and predicting correctly.

Data Gathering

CNN Algorithm

Train model

Prediction

Test model

Fig: Steps of dataflow diagram

**DESIGN ARCHITECTURE**

**General**

Design is meaningful engineering representation of something that is to be built. Software design is a process design is the perfect way to accurately translate requirements in to a finished software product. Design creates a representation or model, provides detail about software data structure, architecture, interfaces and components that are necessary to implement a system.

**Data Flow Diagram:**

Image Details

Test dataset

Preprocessing

Brain Tumor Diagnosis

CNN Algorithm

Training dataset

Fig: Process of dataflow diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model. Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top down approach to Systems Design. Symbols and Notations Used in DFDs Using any convention’s DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

External entity: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

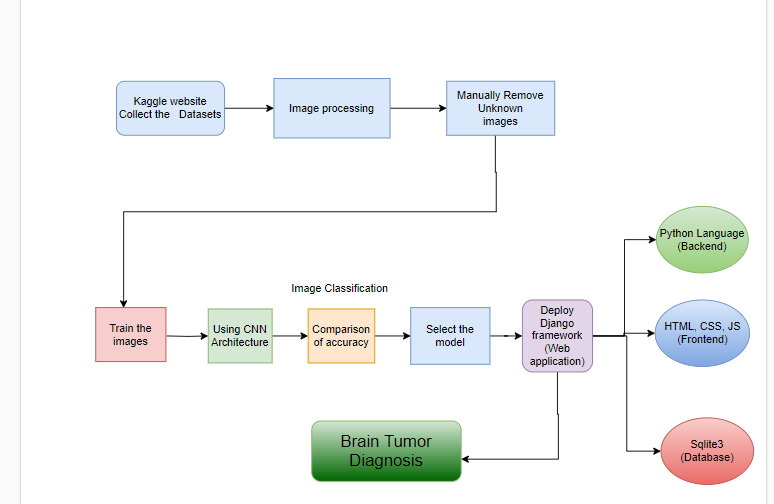
Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.

Data store: files or repositories that hold information for later use, such as a database table or a membership form.

Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

DFD levels and layers A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish. DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analyzed or modeled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

**System Architecture:**



**Work flow diagram:**

Source images

Testing Dataset

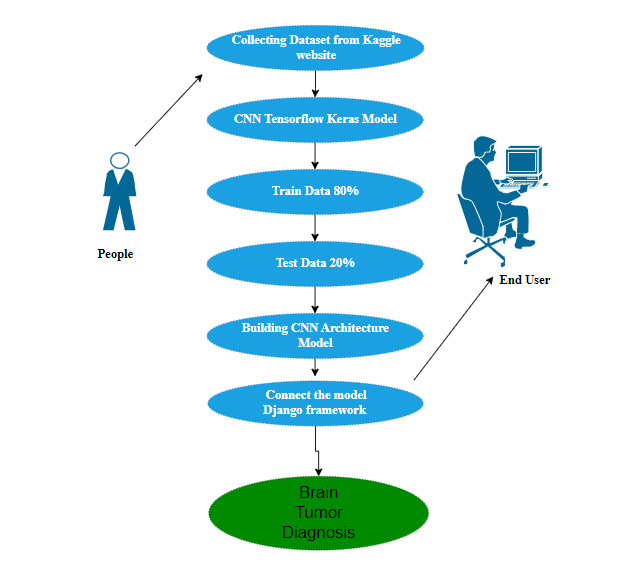
Training Dataset

CNN algorithm

**Detection(**Brain Tumor Diagnosis**)**

Fig: Workflow Diagram

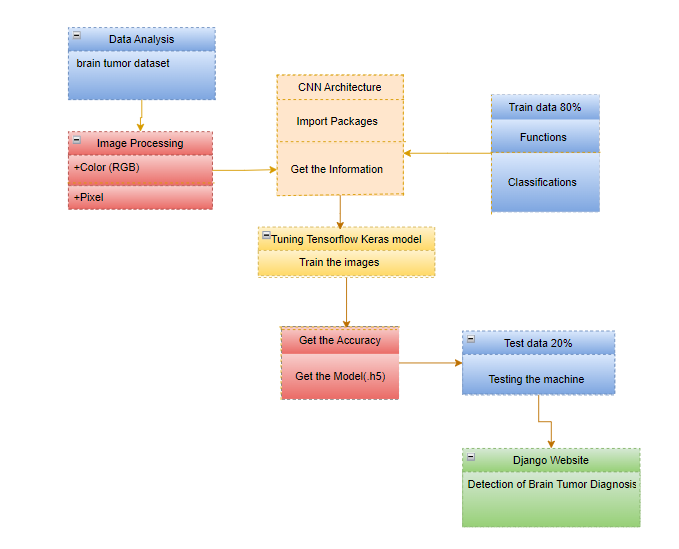
**USECASE DIAGRAM:**

****

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

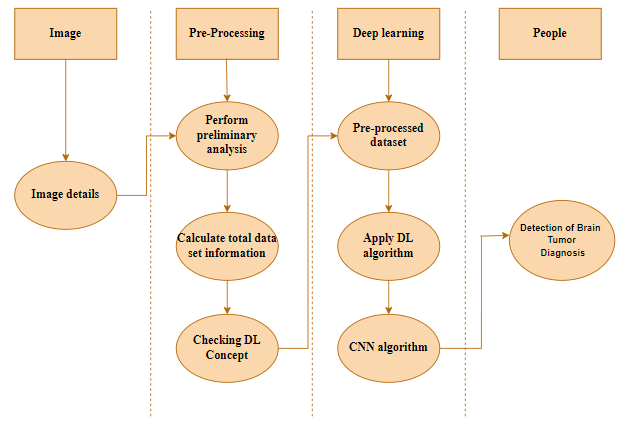
**CLASS DIAGRAM:**

%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3B%22%20edge%3D%221%22%20source%3D%223%22%20target%3D%225%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%223%22%20value%3D%22%22%20style%3D%22verticalLabelPosition%3Dbottom%3Bhtml%3D1%3BverticalAlign%3Dtop%3Balign%3Dcenter%3BstrokeColor%3Dnone%3BfillColor%3D%2300BEF2%3Bshape%3Dmxgraph.azure.azure\_website%3BpointerEvents%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22110%22%20y%3D%22120%22%20width%3D%2280%22%20height%3D%2270%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%224%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3B%22%20edge%3D%221%22%20source%3D%225%22%20target%3D%228%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%225%22%20value%3D%22%22%20style%3D%22sketch%3D0%3Bhtml%3D1%3Baspect%3Dfixed%3BstrokeColor%3Dnone%3Bshadow%3D0%3Balign%3Dcenter%3BverticalAlign%3Dtop%3BfillColor%3D%232D9C5E%3Bshape%3Dmxgraph.gcp2.external\_data\_center%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22370%22%20y%3D%22104%22%20width%3D%2277%22%20height%3D%22100%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%226%22%20value%3D%22%26lt%3Bb%26gt%3BWebsite%26lt%3B%2Fb%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22118%22%20y%3D%22187%22%20width%3D%2264%22%20height%3D%2226%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%227%22%20value%3D%22%26lt%3Bb%26gt%3BDataset%26lt%3B%2Fb%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22378.5%22%20y%3D%22197%22%20width%3D%2261%22%20height%3D%2226%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%228%22%20value%3D%22%22%20style%3D%22verticalLabelPosition%3Dbottom%3Bsketch%3D0%3Baspect%3Dfixed%3Bhtml%3D1%3BverticalAlign%3Dtop%3BstrokeColor%3Dnone%3Balign%3Dcenter%3BoutlineConnect%3D0%3Bshape%3Dmxgraph.citrix.process%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22610%22%20y%3D%22123%22%20width%3D%2252%22%20height%3D%2262%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%229%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3BentryX%3D0.4%3BentryY%3D0%3BentryDx%3D0%3BentryDy%3D0%3BentryPerimeter%3D0%3B%22%20edge%3D%221%22%20target%3D%2211%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%3E%3CmxPoint%20x%3D%22631%22%20y%3D%22195%22%20as%3D%22sourcePoint%22%2F%3E%3C%2FmxGeometry%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2210%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3BentryX%3D1%3BentryY%3D0.5%3BentryDx%3D0%3BentryDy%3D0%3B%22%20edge%3D%221%22%20source%3D%2211%22%20target%3D%2213%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2211%22%20value%3D%22%22%20style%3D%22shape%3Dimage%3Bhtml%3D1%3BverticalAlign%3Dtop%3BverticalLabelPosition%3Dbottom%3BlabelBackgroundColor%3D%23ffffff%3BimageAspect%3D0%3Baspect%3Dfixed%3Bimage%3Dhttps%3A%2F%2Fcdn4.iconfinder.com%2Fdata%2Ficons%2Fessentials-74%2F24%2F017\_-\_Pie\_Chart-128.png%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22606%22%20y%3D%22350%22%20width%3D%2260%22%20height%3D%2260%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2212%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3BentryX%3D1%3BentryY%3D0.5%3BentryDx%3D0%3BentryDy%3D0%3B%22%20edge%3D%221%22%20source%3D%2213%22%20target%3D%2217%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2213%22%20value%3D%22%22%20style%3D%22shape%3Dimage%3Bhtml%3D1%3BverticalAlign%3Dtop%3BverticalLabelPosition%3Dbottom%3BlabelBackgroundColor%3D%23ffffff%3BimageAspect%3D0%3Baspect%3Dfixed%3Bimage%3Dhttps%3A%2F%2Fcdn2.iconfinder.com%2Fdata%2Ficons%2Fseo-web-1-filled-outline%2F64%2Falgorithm-scheme-workflow-128.png%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22460%22%20y%3D%22344%22%20width%3D%2272%22%20height%3D%2272%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2214%22%20value%3D%22%26lt%3Bb%26gt%3Bvisualization%26lt%3B%2Fb%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22591%22%20y%3D%22413%22%20width%3D%2290%22%20height%3D%2226%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2215%22%20value%3D%22%26lt%3Bb%26gt%3BAlgorithm%26lt%3Bbr%26gt%3BImplementation%26lt%3B%2Fb%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22443%22%20y%3D%22416%22%20width%3D%22107%22%20height%3D%2241%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2216%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3B%22%20edge%3D%221%22%20source%3D%2217%22%20target%3D%2220%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2217%22%20value%3D%22%22%20style%3D%22shape%3Dimage%3Bhtml%3D1%3BverticalAlign%3Dtop%3BverticalLabelPosition%3Dbottom%3BlabelBackgroundColor%3D%23ffffff%3BimageAspect%3D0%3Baspect%3Dfixed%3Bimage%3Dhttps%3A%2F%2Fcdn1.iconfinder.com%2Fdata%2Ficons%2Funicons-line-vol-2%2F24%2Fchart-line-128.png%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22270%22%20y%3D%22339.5%22%20width%3D%2281%22%20height%3D%2281%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2218%22%20value%3D%22%26lt%3Bb%26gt%3BAccuracy%26lt%3B%2Fb%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22274.5%22%20y%3D%22322%22%20width%3D%2272%22%20height%3D%2226%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2219%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3B%22%20edge%3D%221%22%20source%3D%2220%22%20target%3D%2223%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%3E%3CArray%20as%3D%22points%22%3E%3CmxPoint%20x%3D%22139%22%20y%3D%22470%22%2F%3E%3CmxPoint%20x%3D%22385%22%20y%3D%22470%22%2F%3E%3C%2FArray%3E%3C%2FmxGeometry%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2220%22%20value%3D%22%22%20style%3D%22sketch%3D0%3BoutlineConnect%3D0%3BfontColor%3D%23232F3E%3BgradientColor%3Dnone%3BfillColor%3D%23067F68%3BstrokeColor%3Dnone%3Bdashed%3D0%3BverticalLabelPosition%3Dbottom%3BverticalAlign%3Dtop%3Balign%3Dcenter%3Bhtml%3D1%3BfontSize%3D12%3BfontStyle%3D0%3Baspect%3Dfixed%3BpointerEvents%3D1%3Bshape%3Dmxgraph.aws4.sagemaker\_model%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22100%22%20y%3D%22341%22%20width%3D%2278%22%20height%3D%2278%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2221%22%20value%3D%22%26lt%3Bb%26gt%3BModel%26lt%3B%2Fb%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22114%22%20y%3D%22307%22%20width%3D%2253%22%20height%3D%2226%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2222%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3BentryX%3D0%3BentryY%3D0.5%3BentryDx%3D0%3BentryDy%3D0%3BentryPerimeter%3D0%3B%22%20edge%3D%221%22%20source%3D%2223%22%20target%3D%2227%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2223%22%20value%3D%22%22%20style%3D%22outlineConnect%3D0%3Bdashed%3D0%3BverticalLabelPosition%3Dbottom%3BverticalAlign%3Dtop%3Balign%3Dcenter%3Bhtml%3D1%3Bshape%3Dmxgraph.aws3.codedeploy%3BfillColor%3D%23759C3E%3BgradientColor%3Dnone%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22351%22%20y%3D%22550%22%20width%3D%2267.5%22%20height%3D%2281%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2224%22%20value%3D%22%26lt%3Bb%26gt%3BDeploy%26lt%3B%2Fb%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22355.75%22%20y%3D%22633%22%20width%3D%2258%22%20height%3D%2226%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2225%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3BentryX%3D0%3BentryY%3D0.5%3BentryDx%3D0%3BentryDy%3D0%3BentryPerimeter%3D0%3B%22%20edge%3D%221%22%20source%3D%2226%22%20target%3D%2223%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2226%22%20value%3D%22%22%20style%3D%22shape%3Dmxgraph.bpmn.data%3BlabelPosition%3Dcenter%3BverticalLabelPosition%3Dbottom%3Balign%3Dcenter%3BverticalAlign%3Dtop%3BwhiteSpace%3Dwrap%3Bsize%3D15%3Bhtml%3D1%3BbpmnTransferType%3Dinput%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22190%22%20y%3D%22560.5%22%20width%3D%2240%22%20height%3D%2260%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2227%22%20value%3D%22%22%20style%3D%22shape%3Dmxgraph.bpmn.data%3BlabelPosition%3Dcenter%3BverticalLabelPosition%3Dbottom%3Balign%3Dcenter%3BverticalAlign%3Dtop%3BwhiteSpace%3Dwrap%3Bsize%3D15%3Bhtml%3D1%3BbpmnTransferType%3Doutput%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22551%22%20y%3D%22560.5%22%20width%3D%2240%22%20height%3D%2260%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2228%22%20value%3D%22%26lt%3Bb%26gt%3BInput%26lt%3B%2Fb%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22187%22%20y%3D%22622%22%20width%3D%2247%22%20height%3D%2226%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2229%22%20value%3D%22%26lt%3Bb%26gt%3Boutput%26lt%3B%2Fb%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3Bresizable%3D0%3Bpoints%3D%5B%5D%3Bautosize%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22544%22%20y%3D%22622%22%20width%3D%2255%22%20height%3D%2226%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2230%22%20value%3D%22%26lt%3Bb%26gt%3BPreprocessing%26lt%3B%2Fb%26gt%3B%22%20style%

****

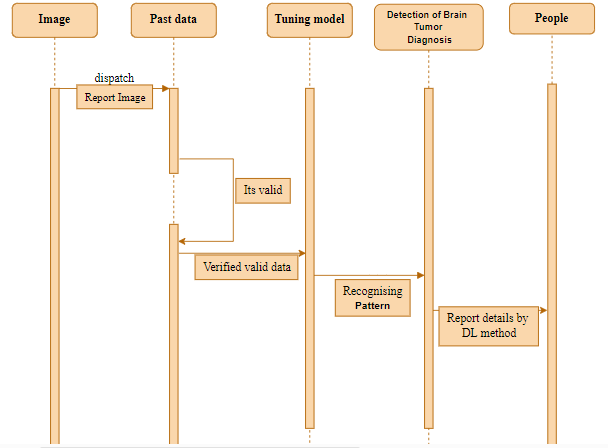
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

**ACTIVITY DIAGRAM:**

****

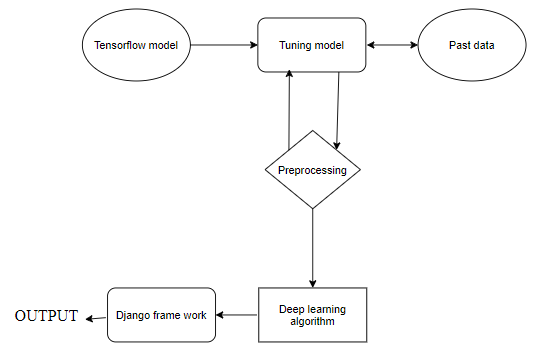
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

**SEQUENCE DIAGRAM:**



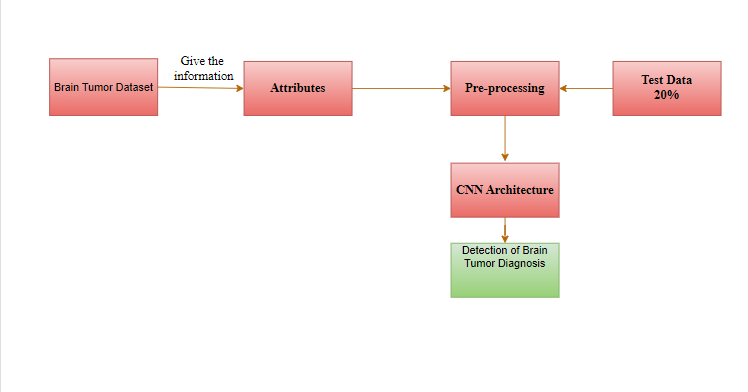
Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include [activity diagramming](http://agilemodeling.com/artifacts/activityDiagram.htm), [communication diagramming](http://agilemodeling.com/artifacts/communicationDiagram.htm), [timing diagramming](http://agilemodeling.com/artifacts/timingDiagram.htm), and [interaction overview diagramming](http://agilemodeling.com/artifacts/interactionOverviewDiagram.htm). Sequence diagrams, along with [class diagrams](http://agilemodeling.com/artifacts/classDiagram.htm) and [physical data models](http://agiledata.org/essays/dataModeling101.html) are in my opinion the most important design-level models for modern business application development.

**ER DIAGRAM:**



An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

**COLLABORATION DIAGRAM:**



A collaboration diagram show the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram.it can be the decomposition of a use case, use case diagram, or part of a use case diagram.

The collaboration diagram shows messages being sent between classes and object (instances). A diagram is created for each system operation that relates to the current development cycle (iteration).

**SOFTWARE DESCRIPTION**

NeuroFusion is a cutting-edge software solution designed to revolutionize the diagnosis of brain tumors through advanced Convolutional Neural Network (CNN) architectures. This innovative tool leverages state-of-the-art deep learning techniques to analyze and interpret complex medical imaging data, offering unparalleled accuracy in identifying and classifying brain tumors. By integrating CNNs, NeuroFusion can automatically detect subtle patterns and anomalies in MRI and CT scans that may elude traditional diagnostic methods. The software is equipped with a user-friendly interface that allows clinicians to input imaging data and receive real-time diagnostic insights, facilitating faster and more precise decision-making. With its ability to continuously learn and adapt from new data, NeuroFusion is poised to significantly enhance diagnostic capabilities, reduce diagnostic errors, and ultimately improve patient outcomes in the field of neurology.

**ANACONDA NAVIGATOR:**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

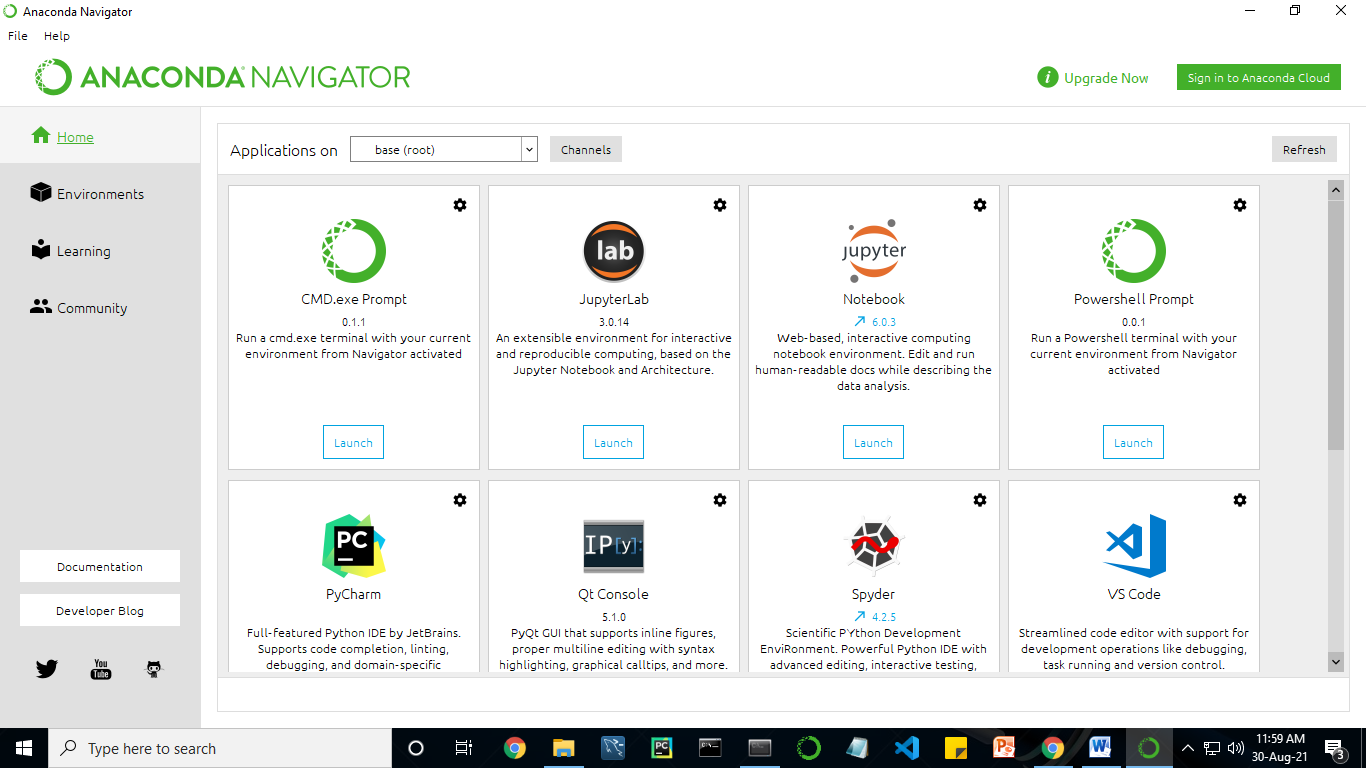
In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

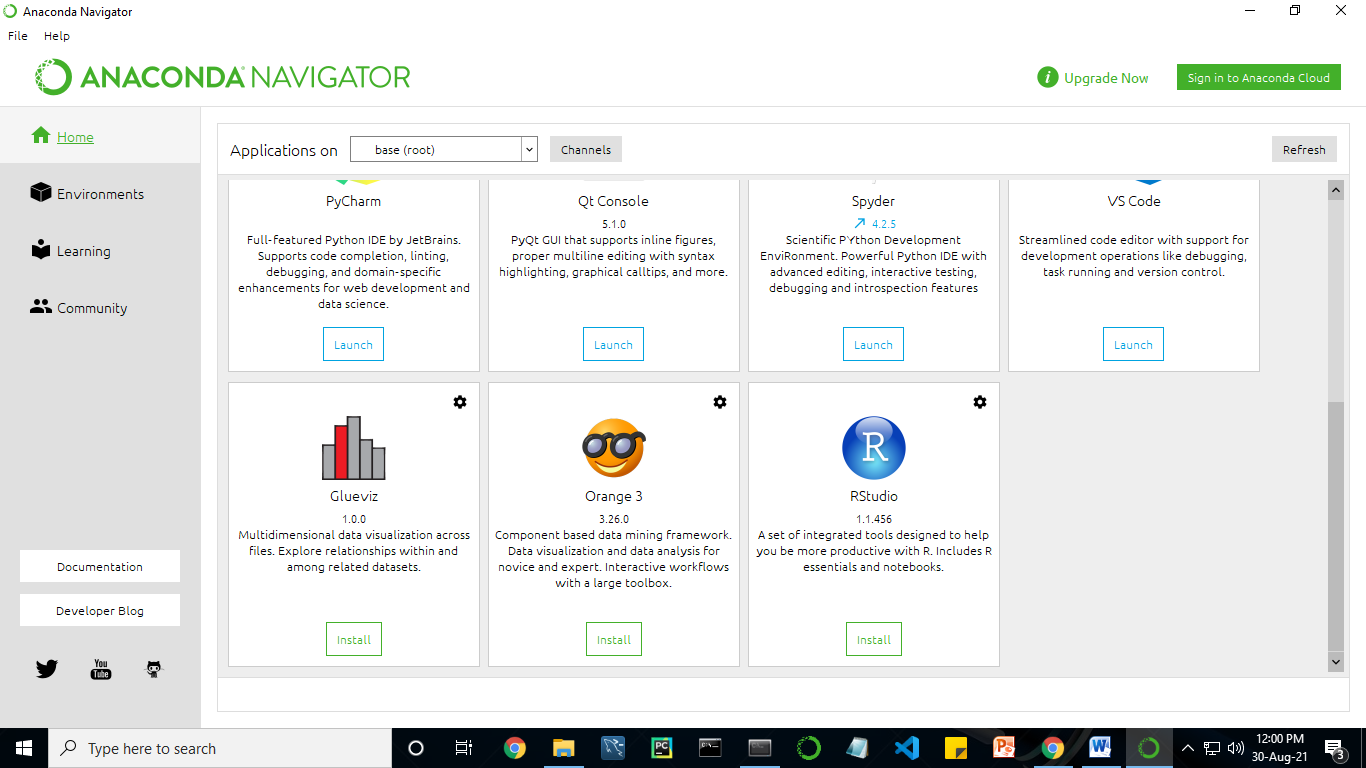
The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

* [JupyterLab](https://jupyterlab.readthedocs.io/en/stable/)
* [Jupyter Notebook](https://jupyter.readthedocs.io/en/latest/)
* [Spyder](https://www.spyder-ide.org/)
* [PyCharm](https://www.jetbrains.com/pycharm/documentation/)
* [VSCode](https://code.visualstudio.com/docs)
* [Glueviz](http://glueviz.org/en/stable/)
* [Orange 3 App](http://orange.biolab.si/docs/)
* [RStudio](http://docs.rstudio.com/)
* Anaconda Prompt (Windows only)
* Anaconda PowerShell (Windows only)





Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with conda list on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

**JUPYTER NOTEBOOK:**

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app), which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

## Installation: The easiest way to install the Jupyter Notebook App is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda**

# Running the Jupyter Notebook

## Launching Jupyter Notebook App: The [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) can be launched by clicking on the Jupyter Notebook icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (cmd on Windows): “jupyter notebook”

## This will launch a new browser window (or a new tab) showing the [Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#dashboard), a sort of control panel that allows (among other things) to select which notebook to open.

## When started, the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) start-up folder which will contain all the notebooks.

## Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy…) and save the modifications on the copy.

## Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

* Launch the jupyter notebook app
* In the [Notebook Dashboard](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#dashboard) navigate to find the notebook: clicking on its name will open it in a new browser tab.
* Click on the menu Help -> User Interface Tour for an overview of the [Jupyter Notebook App](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#notebook-app) user interface.
* You can run the notebook document step-by-step (one cell a time) by pressing shift + enter.
* You can run the whole notebook in a single step by clicking on the menu Cell -> Run All.
* To restart the [kernel](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#kernel) (i.e. the computational engine), click on the menu Kernel -> Restart. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc…).

**PYCHARM:**

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive [Python](https://www.jetbrains.com/help/pycharm/python.html), [web](https://www.jetbrains.com/help/pycharm/web-frameworks.html), and [data science](https://www.jetbrains.com/help/pycharm/scientific-tools.html) development. Code faster and with more easily in a smart and configurable editor with code completion, snippets, code folding and split windows support.

**PYTHON**

**Introduction:**

**Python** is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Its design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) as well as its [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects.

Python is [dynamically-typed](https://en.wikipedia.org/wiki/Type_system#DYNAMIC) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigms), including [structured](https://en.wikipedia.org/wiki/Structured_programming) (particularly, [procedural](https://en.wikipedia.org/wiki/Procedural_programming)), object-oriented and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). It is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

[Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) began working on Python in the late 1980s, as a successor to the [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)), and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension) and a garbage collection system using [reference counting](https://en.wikipedia.org/wiki/Reference_counting). Python 3.0 was released in 2008 and was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility). Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages

**History:**

Python was conceived in the late 1980s by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) at [Centrum Wiskunde & Informatica](https://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica) (CWI) in the [Netherlands](https://en.wikipedia.org/wiki/Netherlands) as a successor to [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)), which was inspired by [SETL](https://en.wikipedia.org/wiki/SETL),  capable of [exception handling](https://en.wikipedia.org/wiki/Exception_handling) and interfacing with the [Amoeba](https://en.wikipedia.org/wiki/Amoeba_(operating_system)) operating system. Its implementation began in December 1989.  Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's [Benevolent Dictator For Life](https://en.wikipedia.org/wiki/Benevolent_Dictator_For_Life), a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project.  As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a [cycle-detecting](https://en.wikipedia.org/wiki/Cycle_detection) [garbage collector](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) and support for [Unicode](https://en.wikipedia.org/wiki/Unicode).

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility). Many of its major features were [backported](https://en.wikipedia.org/wiki/Backporting) to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's [end-of-life](https://en.wikipedia.org/wiki/End-of-life_(product)) date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's [end-of-life](https://en.wikipedia.org/wiki/End-of-life_(product)), only Python 3.6.x  and later are supported.

Python 3.9.2 and 3.8.8 were expeditedas all versions of Python (including 2.7) had security issues, leading to possible [remote code execution](https://en.wikipedia.org/wiki/Remote_code_execution) and [web cache poisoning](https://en.wikipedia.org/wiki/Cache_poisoning).

**METHODOLOGY**

The methodology for "NeuroFusion Advancing Brain Tumor Diagnosis Using CNN Architectures" involves a multi-step process aimed at improving the accuracy and efficiency of brain tumor diagnosis. Initially, high-resolution MRI images of the brain are preprocessed to enhance image quality and normalize intensity levels. These images are then fed into Convolutional Neural Networks (CNNs) designed to extract and learn hierarchical features from the data. The CNN architectures are fine-tuned using a combination of transfer learning and custom-designed layers to optimize performance. The model's output is evaluated using various metrics to ensure high sensitivity and specificity in detecting and classifying different types of brain tumors. Finally, the CNN-based system is integrated into a clinical workflow, allowing for real-time analysis and support in diagnostic decision-making.

CNN Weights

**Detection of Brain**

**Tumor  
 Diagnosis**

Raw image

Build a sequential model

CNN train

The train dataset is used to train the model (CNN) so that it can identify the test image and the disease it has CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D. After the model is trained successfully, the software can identify the Brain Tumor Classification image contained in the dataset. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the Brain Tumor.

**CNN Model steps:**

**Conv2d:**

The 2D convolution is a fairly simple operation at heart: you start with a kernel, which is simply a small matrix of weights. This kernel “slides” over the 2D input data, performing an elementwise multiplication with the part of the input it is currently on, and then summing up the results into a single output pixel.

The kernel repeats this process for every location it slides over, converting a 2D matrix of features into yet another 2D matrix of features. The output features are essentially, the weighted sums (with the weights being the values of the kernel itself) of the input features located roughly in the same location of the output pixel on the input layer.

Whether or not an input feature falls within this “roughly same location”, gets determined directly by whether it’s in the area of the kernel that produced the output or not. This means the size of the kernel directly determines how many (or few) input features get combined in the production of a new output feature.

This is all in pretty stark contrast to a fully connected layer. In the above example, we have 5×5=25 input features, and 3×3=9 output features. If this were a standard fully connected layer, you’d have a weight matrix of 25×9 = 225 parameters, with every output feature being the weighted sum of every single input feature. Convolutions allow us to do this transformation with only 9 parameters, with each output feature, instead of “looking at” every input feature, only getting to “look” at input features coming from roughly the same location. Do take note of this, as it’ll be critical to our later discussion.

**MaxPooling2D layer**

Down samples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool\_size) for each channel of the input. The window is shifted by strides along each dimension.

The resulting output, when using the "valid" padding option, has a spatial shape (number of rows or columns) of: output\_shape = math.floor((input\_shape - pool\_size) / strides) + 1 (when input\_shape >= pool\_size)

The resulting output shape when using the "same" padding option is: output\_shape = math.floor((input\_shape - 1) / strides) + 1

**Arguments**

• pool\_size: integer or tuple of 2 integers, window size over which to take the maximum. (2, 2) will take the max value over a 2x2 pooling window. If only one integer is specified, the same window length will be used for both dimensions.

• strides: Integer, tuple of 2 integers, or None. Strides values. Specifies how far the pooling window moves for each pooling step. If None, it will default to pool\_size.

• padding: One of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.

• data\_format: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape (batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

**Input shape**

• If data\_format='channels\_last': 4D tensor with shape (batch\_size, rows, cols, channels).

• If data\_format='channels\_first': 4D tensor with shape (batch\_size, channels, rows, cols).

**Output shape**

• If data\_format='channels\_last': 4D tensor with shape (batch\_size, pooled\_rows, pooled\_cols, channels).

• If data\_format='channels\_first': 4D tensor with shape (batch\_size, channels, pooled\_rows, pooled\_cols).

**Flatten layer**

It is used to flatten the dimensions of the image obtained after convolving it. Dense: It is used to make this a fully connected model and is the hidden layer. Dropout: It is used to avoid over fitting on the dataset and dense is the output layer contains only one neuron which decide to which category image belongs.

Flatten is used to flatten the input. For example, if flatten is applied to layer having input shape as (batch\_size, 2,2), then the output shape of the layer will be (batch\_size, 4)

Flatten has one argument as follows

keras.layers.Flatten(data\_format = None)

data\_format is an optional argument and it is used to preserve weight ordering when switching from one data format to another data format. It accepts either channels\_last or channels\_first as value. channels\_last is the default one and it identifies the input shape as (batch\_size, ..., channels) whereas channels\_first identifies the input shape as (batch\_size, channels, ...)

**Dense layer**

Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use\_bias is True). These are all attributes of Dense.

Note: If the input to the layer has a rank greater than 2, then Dense computes the dot product between the inputs and the kernel along the last axis of the inputs and axis 0 of the kernel (using tf.tensordot). For example, if input has dimensions (batch\_size, d0, d1), then we create a kernel with shape (d1, units), and the kernel operates along axis 2 of the input, on every sub-tensor of shape (1, 1, d1) (there are batch\_size \* d0 such sub-tensors). The output in this case will have shape (batch\_size, d0, units).

Besides, layer attributes cannot be modified after the layer has been called once (except the trainable attribute). When a popular kwarg input\_shape is passed, then keras will create an input layer to insert before the current layer. This can be treated equivalent to explicitly defining an InputLayer.

**Arguments**

• units: Positive integer, dimensionality of the output space.

• activation: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).

• use\_bias: Boolean, whether the layer uses a bias vector.

• kernel\_initializer: Initializer for the kernel weights matrix.

• bias\_initializer: Initializer for the bias vector.

• kernel\_regularizer: Regularizer function applied to the kernel weights matrix.

• bias\_regularizer: Regularizer function applied to the bias vector.

• activity\_regularizer: Regularizer function applied to the output of the layer (its "activation").

• kernel\_constraint: Constraint function applied to the kernel weights matrix.

• bias\_constraint: Constraint function applied to the bias vector.

**Input shape**

N-D tensor with shape: (batch\_size, ..., input\_dim). The most common situation would be a 2D input with shape (batch\_size, input\_dim).

**Output shape**

N-D tensor with shape: (batch\_size, ..., units). For instance, for a 2D input with shape (batch\_size, input\_dim), the output would have shape (batch\_size, units).

# **Dropout layer**

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

Note that the Dropout layer only applies when training is set to True such that no values are dropped during inference. When using model.fit, training will be appropriately set to True automatically, and in other contexts, you can set the kwarg explicitly to True when calling the layer.

(This is in contrast to setting trainable=False for a Dropout layer. trainable does not affect the layer's behavior, as Dropout does not have any variables/weights that can be frozen during training.)

**Arguments**

* **rate**: Float between 0 and 1. Fraction of the input units to drop.
* **noise\_shape**: 1D integer tensor representing the shape of the binary dropout mask that will be multiplied with the input. For instance, if your inputs have shape (batch\_size, timesteps, features) and you want the dropout mask to be the same for all timesteps, you can use noise\_shape=(batch\_size, 1, features).

**seed**: A Python integer to use as random seed.

**Image Data Generator:**

It is that rescales the image, applies shear in some range, zooms the image and does horizontal flipping with the image. This Image Data Generator includes all possible orientation of the image.

**Training Process:**

train\_datagen.flow\_from\_directory is the function that is used to prepare data from the train\_dataset directory Target\_size specifies the target size of the image. Test\_datagen.flow\_from\_directory is used to prepare test data for the model and all is similar as above. fit\_generator is used to fit the data into the model made above, other factors used are steps\_per\_epochs tells us about the number of times the model will execute for the training data.

**Epochs:**

It tells us the number of times model will be trained in forward and backward pass.

**Validation process:**

Validation\_data is used to feed the validation/test data into the model. Validation\_steps denotes the number of validation/test samples.

**ARCHITECTURE OF CNN**

**CONVOLUTIONAL NEURAL NETWORK:**

A Convolutional neural network (CNN) is one type of Artificial Neural Network. A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data.

**Models API:**

**There are three ways to create Keras models:**

• The Sequential model, which is very straightforward (a simple list of layers), but is limited to single-input, single-output stacks of layers (as the name gives away).

• The Functional API, which is an easy-to-use, fully-featured API that supports arbitrary model architectures. For most people and most use cases, this is what you should be using. This is the Keras "industry strength" model.

• Model subclassing, where you implement everything from scratch on your own. Use this if you have complex, out-of-the-box research use cases.

**Types of Keras Models**

**Models in keras are available in two types:**

• Keras Sequential Model

• Keras Functional API

**1. Sequential Model in Keras**

It allows us to create models layer by layer in sequential order. But it does not allow us to create models that have multiple inputs or outputs.

It is best for simple stack of layers which have 1 input tensor and 1 output tensor.

This model is not suited when any of the layer in the stack has multiple inputs or outputs. Even if we want non-linear topology, it is not suited.

**2. Functional API in Keras**

It provides more flexibility to define a model and add layers in keras. Functional API allows us to create models that have multiple input or output.

It also allows us to share these layers. In other words. we can make graphs of layers using Keras functional API.

As functional API is a data structure, it is easy to save it as a single file that helps in recreating the exact model without having the original code. Also its easy to model the graph here and access its nodes as well.

**TYPES OF CNN:**

* **VGG**
* **LENET NET**

**LIST OF MODULES**

1. Manual Net

2. VGG

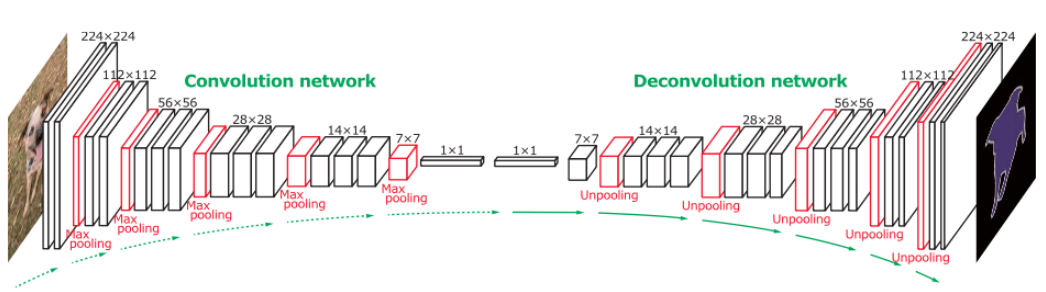
3. LENET

4. Deploy

**MANUAL ARCHITECTURE:**

Manual Architecture is a novel convolutional neural network designed to address specific challenges in machine learning and computer vision tasks. Leveraging a carefully crafted structure, ManuLAL aims to optimize both model performance and computational efficiency.

**Architecture of Manual:**

****

**Modular Feature Extraction Blocks:**

ManuLAL incorporates modular feature extraction blocks, each consisting of a combination of depthwise separable convolutions and dilated convolutions. This design choice allows the network to capture multi-scale features efficiently, enabling robust representation learning.

**Attention Mechanism:**

An attention mechanism is integrated into ManuLAL to enhance the focus on salient regions within the input data. This attention mechanism dynamically adjusts the weights assigned to different parts of the feature maps, promoting more effective feature learning.

**Residual Connections:**

To facilitate gradient flow and encourage the learning of residual features, Manual employs residual connections. These connections enable the network to better capture intricate patterns and ease the training process, contributing to overall model stability.

**Adaptive Pooling Layers:**

Manual features adaptive pooling layers that dynamically adjust their pooling size based on the input dimensions. This adaptability ensures that spatial information is preserved, particularly crucial for tasks where fine-grained details are essential.

**Cross-Modal Fusion:**

In addressing multi-modal learning challenges, Manual introduces cross-modal fusion layers. These layers enable the network to seamlessly integrate information from different modalities, promoting a holistic understanding of complex input data.

**Sparse Connections:**

To optimize computational resources, Manual incorporates sparse connections within the network architecture. This strategic sparsity enhances training efficiency without compromising the model's ability to capture intricate features.

**Flexible Activation Functions:**

Manual provides flexibility in activation functions, allowing users to choose from a variety of non-linearities based on the task requirements. This adaptability empowers researchers and practitioners to tailor the network to specific use cases.

**Practical Applications:**

The Manual Architecture is designed with versatility in mind, making it suitable for a wide range of applications, including image classification, object detection, and semantic segmentation. Its modular design and attention to efficiency make it particularly well-suited for deployment in resource-constrained environments.

**Community Integration and Contributions:**

As an open-source initiative, Manual encourages community involvement and contributions. Researchers and developers can extend the architecture, experiment with different modules, and collaborate to further enhance its capabilities.

Manual Architecture represents a forward-thinking approach to convolutional neural networks, emphasizing modularity, efficiency, and adaptability. Its unique features make it a promising candidate for various machine learning tasks, contributing to the evolving landscape of deep learning architectures.

**VGG:**

The VGG (Visual Geometry Group) architecture is a deep convolutional neural network (CNN) architecture designed for image classification. It was developed by researchers at the University of Oxford and was a runner-up in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

Here is a description of the key characteristics and components of the VGG architecture:

Deep Convolutional Layers: VGG is known for its deep architecture, which consists of a stack of convolutional layers. Unlike earlier architectures like AlexNet, which had a few large convolutional layers, VGG uses many smaller 3x3 convolutional filters with a stride of 1, which allows it to capture more fine-grained features.

Stacked Convolutional Blocks: The VGG architecture is organized into blocks, with each block containing multiple convolutional layers followed by a max-pooling layer. There are different versions of VGG, such as VGG16 and VGG19, which have varying numbers of these blocks.

Pooling Layers: After each set of convolutional layers in a block, a max-pooling layer with a 2x2 filter and a stride of 2 is applied. This helps reduce the spatial dimensions of the feature maps, making the network computationally more efficient and allowing it to focus on more abstract features.

Fully Connected Layers: After the convolutional and pooling layers, VGG has a series of fully connected layers. These layers are typically used for classification tasks and are followed by a softmax activation function to produce class probabilities.

Rectified Linear Units (ReLU): VGG uses the rectified linear unit (ReLU) activation function after each convolutional and fully connected layer. ReLU helps introduce non-linearity into the network and has become a standard choice for deep neural networks.

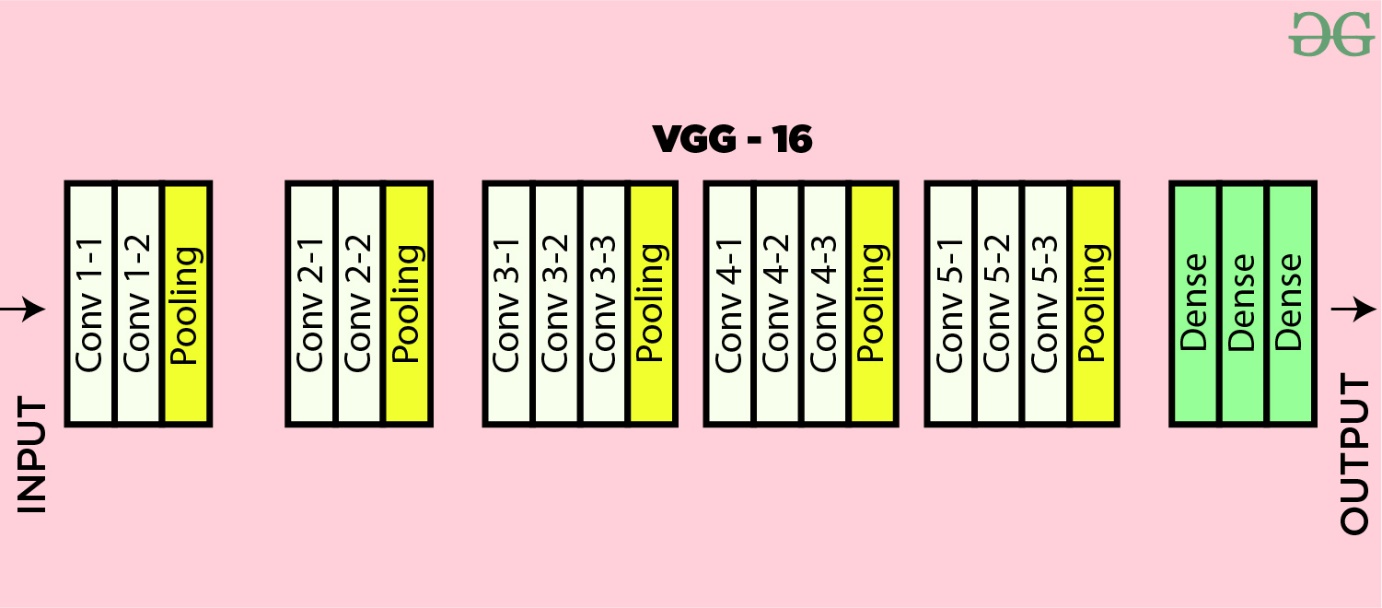
Dropout: Dropout layers are often added to prevent overfitting. During training, dropout randomly sets a fraction of the neurons to zero, reducing the reliance on any one specific neuron and helping to generalize better.

Number of Parameters: VGG architectures have a large number of parameters, which can make them computationally expensive and memory-intensive. For example, VGG16 has approximately 138 million parameters.

Image Input Size: VGG models typically expect images of a fixed input size, such as 224x224 pixels, which is a common size for many image classification datasets.

Pre-Trained Models: Pre-trained versions of VGG on large datasets like ImageNet are often used as a starting point for various computer vision tasks. Transfer learning using pre-trained VGG models has been proven effective for tasks like object detection and image segmentation.

**ARCHITECHTURE OF VGG**:



**Convolutional Layers:**

In ShuffleNet, convolutional layers play a crucial role where filters are applied to the original image or other feature maps within the deep CNN. Key parameters include the number of kernels and the size of the kernels, providing flexibility and customization in the network.

**Pooling Layers:**

Pooling layers in ShuffleNet, akin to convolutional layers, serve specific functions such as max pooling or average pooling. Max pooling extracts the maximum value in a defined filter region, while average pooling calculates the average value. These layers are instrumental in reducing the dimensionality of the network.

**Dense or Fully Connected Layers:**

Before reaching the classification output of ShuffleNet, fully connected layers are employed to flatten the results, a step similar to the output layer of a Multilayer Perceptron (MLP). This ensures that the network's features are effectively transformed for classification purposes.

**Innovative Shuffle Operation:**

ShuffleNet introduces a distinctive Shuffle operation, a pivotal element in its architecture. This operation enhances the network's capacity for feature learning and enables efficient information exchange between channels, contributing to improved model performance.

**Compact Design and Computational Efficiency**:

ShuffleNet is renowned for its compact design and computational efficiency. By incorporating group convolution and pointwise convolution, it achieves impressive accuracy with reduced computational cost, making it particularly suitable for resource-constrained environments.

**Channel Shuffle:**

The Channel Shuffle operation, a unique aspect of ShuffleNet, enables inter-group information flow by rearranging feature maps. This innovation contributes to enhanced representation learning and allows the network to capture diverse and complex patterns.

**Practical Applications:**

ShuffleNet finds applications in various domains, including image classification, object detection, and semantic segmentation. Its efficient architecture makes it a compelling choice for scenarios where computational resources are limited.

**LENET:**

The LeNet architecture is a pioneering convolutional neural network (CNN) architecture developed by Yann LeCun and his colleagues in the early 1990s. It played a crucial role in the advancement of deep learning and was specifically designed for handwritten digit recognition tasks, such as recognizing digits in checks and postal addresses. LeNet laid the foundation for modern CNNs and their applications in image recognition and computer vision tasks.

The LeNet architecture consists of the following layers:

Input Layer: This layer accepts the input image, which is typically a grayscale image of a handwritten digit. The input images are usually of size 32x32 pixels.

Convolutional Layers: LeNet uses two convolutional layers to extract features from the input images. Each convolutional layer applies convolutional filters (also called kernels) to the input image, capturing different patterns and features. These filters slide over the image to create feature maps.

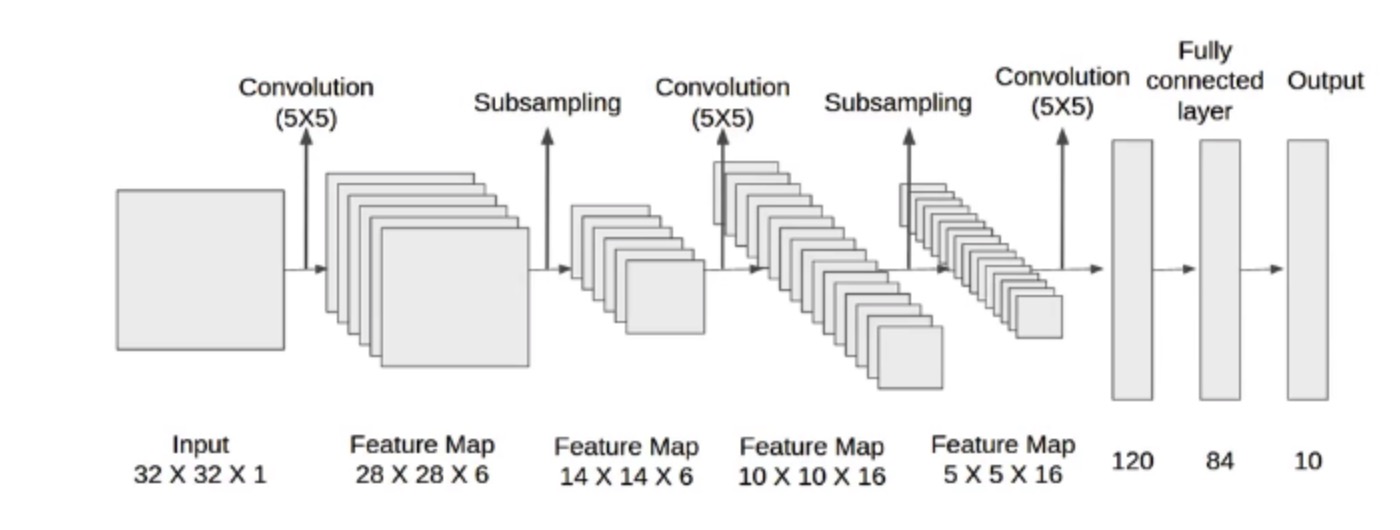
Subsampling (Pooling) Layers: After each convolutional layer, a subsampling layer (also known as a pooling layer) is applied to reduce the spatial dimensions of the feature maps and help in retaining important information while reducing computation.

Fully Connected Layers: The subsampled feature maps are then flattened and passed through fully connected layers, which are traditional neural network layers. These layers are responsible for making classification decisions based on the extracted features.

Output Layer: The final fully connected layer produces the output of the network, which represents the predicted class probabilities. For handwritten digit recognition, this would typically involve 10 output nodes, each corresponding to a digit class (0 to 9).

LeNet was designed before the deep learning era as we know it today, so it has fewer layers compared to modern architectures like ResNet, VGG, or Inception. However, its fundamental design principles, including the use of convolutional and pooling layers, inspired the development of more complex CNN architectures. LeNet demonstrated the potential of neural networks in handling image recognition tasks and played a pivotal role in the resurgence of interest in neural networks during the early 2000s.

**ARCHITECHTURE OF LENET:**



**Fire Modules:**

SqueezeNet introduces the concept of Fire Modules, which are the building blocks of its architecture. These modules efficiently combine both 1x1 (squeeze) and 3x3 (expand) convolutions, aiming to reduce the number of parameters in the network while maintaining expressive feature learning capabilities.

**Bypass Connections:**

Bypass connections, also known as skip connections, are integrated into SqueezeNet to facilitate gradient flow during training. These connections enhance the network's ability to learn both low-level and high-level features, contributing to improved model performance.

**Pooling Layers:**

Similar to other convolutional neural networks, SqueezeNet utilizes pooling layers to downsample feature maps and reduce the spatial dimensions of the data. This helps manage computational complexity and enables the network to focus on essential features.

**Fully Connected Layers:**

Before reaching the final classification output, SqueezeNet employs fully connected layers to flatten the feature maps. This step is crucial for transforming the hierarchical representations learned by the network into a format suitable for classification.

**Efficient Model Size:**

SqueezeNet is specifically designed to be computationally efficient with a small model size. By using 1x1 convolutions judiciously, the network achieves impressive accuracy with significantly fewer parameters compared to traditional architectures.

**Compression Techniques:**

SqueezeNet incorporates compression techniques to further reduce the model size. This includes reducing the number of input channels to 1x1 convolutions, ensuring a balance between computational efficiency and model accuracy.

**Practical Applications:**

SqueezeNet finds applications in scenarios where computational resources are limited, such as edge devices and mobile applications. Its efficient design makes it suitable for real-time image classification and object detection in resource-constrained environments.

**Community Adoption:**

Due to its lightweight and efficient architecture, SqueezeNet has gained popularity in both research and industry. Its adoption extends to various computer vision tasks, demonstrating its versatility and effectiveness in diverse applications.

**MODULE DESCRIPTION:**

**IMPORT THE GIVEN IMAGE FROM DATASET:**

## We have to import our data set using keras preprocessing image data generator function also we create size, rescale, range, zoom range, horizontal flip. Then we import our image dataset from folder through the data generator function. Here we set train, test, and validation also we set target size, batch size and class-mode from this function we have to train using our own created network by adding layers of CNN.

**WORKING PROCESS OF LAYERS IN CNN MODEL:**

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human disease and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units.

Input Layer:

Input layer in CNN contain image data. Image data is represented by three dimensional matrixes. It needs to reshape it into a single column. Suppose you have image of dimension 28 x 28 =784, it need to convert it into 784 x 1 before feeding into input.

## **Convo Layer:**

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then the filter over the next receptive field of the same input image by a Stride and do the same operation again. It will repeat the same process again and again until it goes through the whole image. The output will be the input for the next layer.

**Pooling Layer:**

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If it applies FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive. So, the max pooling is only way to reduce the spatial volume of input image. It has applied max pooling in single depth slice with Stride of 2. It can observe the 4 x 4 dimension input is reducing to 2 x 2 dimensions.

## **Fully Connected Layer (FC):**

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

## **Softmax / Logistic Layer:**

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

## **Output Layer:**

Output layer contains the label which is in the form of one-hot encoded. Now you have a good understanding of CNN.

**ORAL AND KIDNEY CANCER** CLASSIFICATION **IDENTIFICATION:**

We give input image using keras preprocessing package. That input Image converted into array value using pillow and image to array function package. We have already classified diagnosis of oral and Brain Tumor in our dataset. It classifies what are the feelings. Then we have to predict our automobile parts using predict function.

Given dataset

**Detection of Brain**

**Tumor  
 Diagnosis**

Feature Extractions

CNN Model

Input image

**DEPLOY:**

**Deploying the model in Django Framework and predicting output**

In this module the trained deep learning model is converted into hierarchical data format file (.h5 file) which is then deployed in our django framework for providing better user interface and predicting the output.

**Django (Web FrameWork) :**

Django is a micro web framework written in Python.

It is classified as a micro-framework because it does not require particular tools or libraries.

It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Django supports extensions that can add application features as if they were implemented in Django itself.

Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Django was created by [Armin Ronacher](https://en.wikipedia.org/wiki/Armin_Ronacher) of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an [April Fool’s](https://en.wikipedia.org/wiki/April_Fool%27s) joke that was popular enough to make into a serious application. The name is a play on the earlier [Bottle](https://en.wikipedia.org/wiki/Bottle_(web_framework)) framework.

When Ronacher and Georg Brand created a bulletin board system written in Python, the Pocoo projects Werkzeug and [Jinja](https://en.wikipedia.org/wiki/Jinja_(template_engine)) were developed.

In April 2016, the Pocoo team was disbanded and development of Django and related libraries passed to the newly formed Pallets project.

Django has become popular among Python enthusiasts. As of October 2020, it has second most stars on [GitHub](https://en.wikipedia.org/wiki/GitHub) among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.

The micro-framework Django is part of the Pallets Projects, and based on several others of them.

Django **is** based on Werkzeug, [Jinja2](http://quintagroup.com/cms/python/jinja2) and inspired by Sinatra Ruby framework, available under BSD licence. It was developed at pocoo by Armin Ronacher. Although Django is rather young compared to most [Python](https://quintagroup.com/services/python) frameworks, it holds a great promise and has already gained popularity among Python web developers. Let’s take a closer look into Django, so-called “micro” framework for Python.

**FEATURES:**

Django was designed to be **easy to use and extend**.  The idea behind Django is to build a solid foundation for web applications of different complexity. From then on you are free to**plug in any extensions** you think you need. Also you are free to build your own modules. Django is great for all kinds of projects.  It's especially good for prototyping. Django depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit.

Still the question remains why use Django as your web application framework if we have immensely powerful [Django](https://quintagroup.com/services/python/django), [Pyramid,](https://quintagroup.com/cms/python/pyramid) and don’t forget web mega-framework [Turbo-gears](https://quintagroup.com/cms/python/turbogears)? Those are supreme[Python web frameworks](https://quintagroup.com/services/python/python-web-development.png) BUT out-of-the-box Django is pretty impressive too with it’s:

* Built-In Development server and Fast debugger
* integrated support for unit testing
* RESTful request dispatching
* Uses [Jinja2](https://quintagroup.com/cms/python/jinja2) Templating
* support for secure cookies
* Unicode based
* Extensive Documentation
* Google App Engine Compatibility
* Extensions available to enhance features desired

Plus Django gives you so much more **CONTROL** on the development stage of **your project**. It follows the principles of minimalism and let you decide how you will build your application.

* Django has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
* ORM-agnostic: you can plug in your favourite ORM e.g. [SQLAlchemy](https://quintagroup.com/cms/python/sqlalchemy).
* Basic foundation API is nicely shaped and coherent.
* Django documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Django.
* It is super easy to deploy Django in production (Django is 100%WSGI 1.0 compliant”)
* HTTP request handling functionality
* High Flexibility

The configuration is even more flexible than that of Django, giving you plenty of solution for every production need.

To sum up, Django is one of the most polished and feature-rich micro frameworks, available. Still young, Django has a thriving community, first-class extensions, and an **elegant API**.  Django comes with all the benefits of fast templates, strong WSGI features, **thorough unit testability** at the web application and library level, **extensive documentation**. So next time you are starting a new project where you need some good features and a vast number of extensions, definitely check out Django.

Django is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Django's framework is more explicit than Django framework and is also easier to learn because it has less base code to implement a simple web-Application

Django is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Overview of Python Django Framework Web apps are developed to generate content based on retrieved data that changes based on a user’s interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications.

Django is an excellent web development framework for REST API creation. It is built on top of Python which makes it powerful to use all the python features.

Django is used for the backend, but it makes use of a templating language called Jinja2 which is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Django is considered to be more popular because it provides many out of box features and reduces time to build complex applications. Django is a good start if you are getting into web development. Django is a simple, un-opinionated framework; it doesn't decide what your application should look like developers do.

Django is a web framework. This means Django provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, and a wiki or go as big as a web-based calendar application or a commercial website.

**Advantages of Django:**

* Higher compatibility with latest technologies.
* Technical experimentation.
* Easier to use for simple cases.
* Codebase size is relatively smaller.
* High scalability for simple applications.
* Easy to build a quick prototype.
* Routing URL is easy.
* Easy to develop and maintain applications.

Framework Django is a web framework from Python language. Django provides a library and a collection of codes that can be used to build websites, without the need to do everything from scratch. But Framework Django still doesn't use the Model View Controller (MVC) method.

Django-RESTful is an extension for Django that provides additional support for building REST APIs. You will never be disappointed with the time it takes to develop an API. Django-Restful is a lightweight abstraction that works with the existing ORM/libraries. Django-RESTful encourages best practices with minimal setup.

Django Restful is an extension for Django that adds support for building REST APIs in Python using Django as the back-end. It encourages best practices and is very easy to set up. Django restful is very easy to pick up if you're already familiar with Django.

Django is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates and also we can add to this application to create our API.

**Start Using an API**

1. Most APIs require an API key. ...
2. The easiest way to start using an API is by finding an HTTP client online, like REST-Client, Postman, or Paw.
3. The next best way to pull data from an API is by building a URL from existing API documentation.

The Django object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an \_\_init\_\_.py file inside) or a standard module (just a .py file).

For more information about resource loading, see [open resource()](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Flask.open_resource).

Usually you create a [Django](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Flask) instance in your main module or in the \_\_init\_\_.py file of your package.

**Parameters**

* **rule** ([str](https://docs.python.org/3/library/stdtypes.html#str)) – The URL rule string.
* **endpoint** (Optional[[str](https://docs.python.org/3/library/stdtypes.html#str)]) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to view\_func.\_\_name\_\_.
* **view\_func** (Optional[Callable]) – The view function to associate with the endpoint name.
* **provide\_automatic\_options** (Optional[bool]) – Add the OPTIONS method and respond to OPTIONS requests automatically.
* **options** (Any) – Extra options passed to the [Rule](https://werkzeug.palletsprojects.com/en/2.0.x/routing/#werkzeug.routing.Rule) object.

Return type -- [None](https://docs.python.org/3/library/constants.html#None)

After\_Request(f)

Register a function to run after each request to this object.

The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.

If a function raises an exception, any remaining after request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use [teardown\_request()](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Flask.teardown_request) for that.

**Parameters:**

**f** (Callable[[[Response](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Response)], [Response](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Response)])

Return type

Callable[[[Response](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Response)], [Response](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Response)]

after\_request\_funcs: t.Dict[AppOrBlueprintKey,

t.List[AfterRequestCallable]]

A data structure of functions to call at the end of each request, in the format {scope: [functions]}. The scope  key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the [after\_request()](https://flask.palletsprojects.com/en/2.0.x/api/#flask.Flask.after_request) decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

app\_context()

Create an [AppContext](https://flask.palletsprojects.com/en/2.0.x/api/#flask.ctx.AppContext). Use as a with block to push the context, which will make [current\_app](https://flask.palletsprojects.com/en/2.0.x/api/#flask.current_app) point at this application.

An application context is automatically pushed by [RequestContext.push()](https://flask.palletsprojects.com/en/2.0.x/api/#flask.ctx.RequestContext.push) when handling a request, and when running a CLI command. Use this to manually create a context outside of these situations.

With app.app\_context():

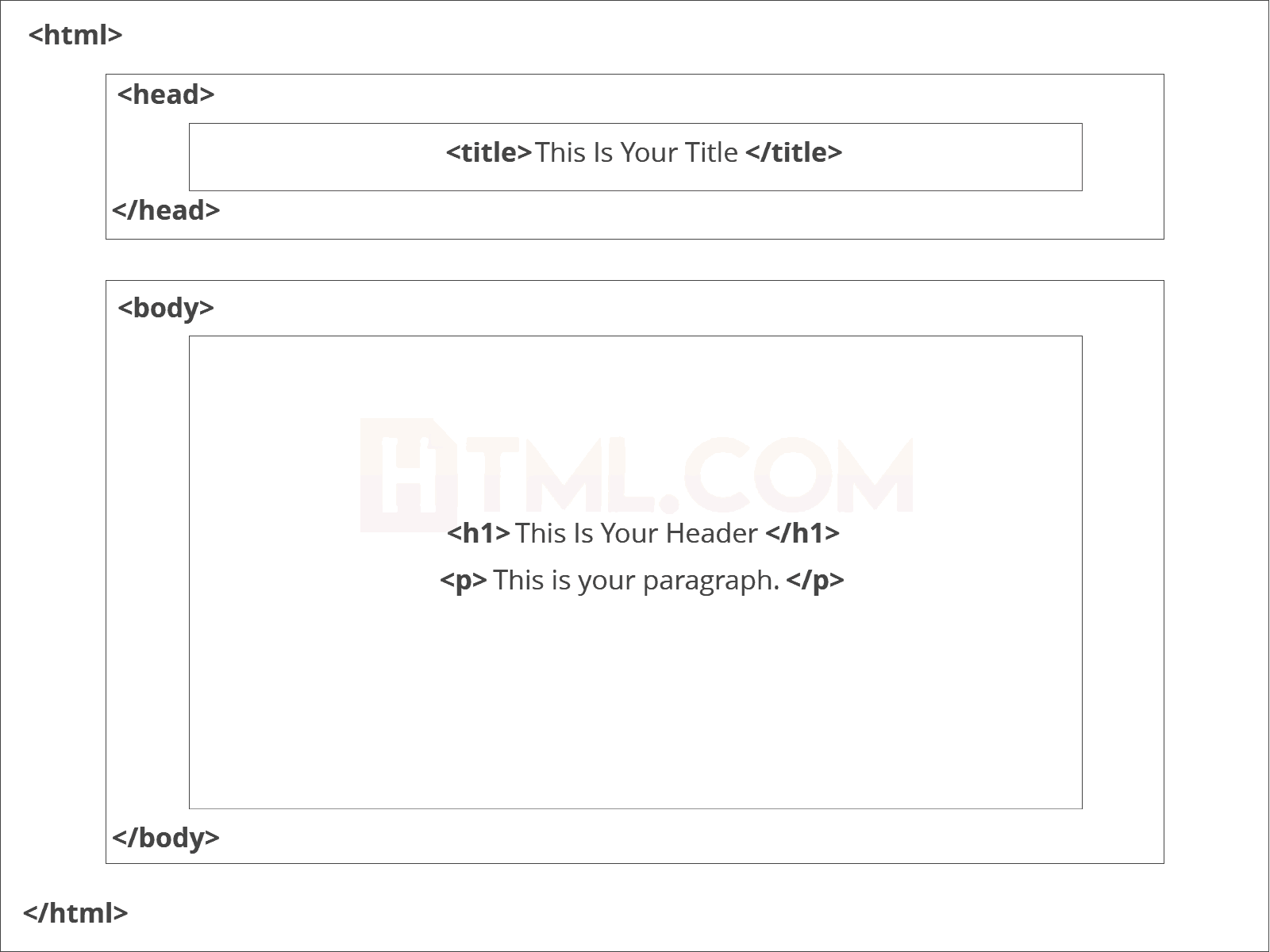
Init\_db()

**HTML**

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

#### Basic Construction of an HTML Page

These tags should be placed underneath each other **at the top of every HTML page** that you create.



<!DOCTYPE html> — This tag**specifies the language** you will write on the page. In this case, the language is HTML 5.

<html> — This tag signals that from here on we are going to write in HTML code.

<head> — This is where all the **metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

<body> — This is where the**content of the page** goes.

#### Further Tags

Inside the <head> tag, there is one tag that is always included: <title>, but there are others that are just as important:

<title>

This is where we**insert the page name** as it will appear at the top of the browser window or tab.

<meta>

This is where information about the document is stored: character encoding, name (page context), description.

**Head Tag**  
<head>

<title>My First Webpage</title>

<meta charset="UTF-8">

<meta name="description" content="This field contains information about your page. It is usually around two sentences long.">.

<meta name="author" content="Conor Sheils">

</header>

### Adding Content

Next, we will make<body> tag.

The HTML <body> is where we add the content which is designed for viewing by human eyes.

This includes **text, images, tables, forms**and everything else that we see on the internet each day.

#### Add HTML Headings To Web Page

In HTML, [headings](https://html.com/tags/heading/) are written in the following elements:

* <h1>
* <h2>
* <h3>
* <h4>
* <h5>
* <h6>

As you might have guessed <h1> and <h2> should be used for the most important titles, while the remaining tags should be used for sub-headings and less important text.

**Search engine bots use this order**when deciphering which information is most important on a page.

##### Creating Your Heading

Let’s try it out. On a new line in the HTML editor, type:

<h1> Welcome To My Page </h1>

And hit save. We will save this file as “index.html” in a new folder called “my webpage.”

**Add Text In HTML**

Adding text to our HTML page is simple using an element opened with the tag <p> which **creates a new paragraph**. We place all of our regular text inside the element <p>.

When we write text in HTML, we also have a number of other elements we can use **to control the text or make it appear in a certain way.**

#### Add Links In HTML

As you may have noticed, the internet is made up of lots of [links](https://html.com/anchors-links/).

Almost everything you click on while surfing the web is a link **takes you to another page** within the website you are visiting or to an external site.

Links are included in an attribute opened by the [**<a>**](https://html.com/tags/a/) tag. This element is the first that we’ve met which uses an attribute and so it**looks different to previously mentioned tags.**

<a href=<http://www.google.com>>Google</a>

**Image Tag**

In today’s modern digital world, [images](https://html.com/blog/100-legal-sources-free-stock-images/) are everything. The [**<**img**>**](https://html.com/tags/img/) tag has everything you need to display images on your site. Much like the <a> anchor element, <img> also contains an attribute.

The attribute features information for your computer regarding the source, height, width and alt text of the image

<img src=”yourimage.jpg” alt=”Describe the image” height=“X” width=“X”>

**CSS**

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users.CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let’s try to break the acronym:

* Cascading: Falling of Styles
* Style: Adding designs/Styling our HTML tags
* Sheets: Writing our style in different documents

## **CSS Syntax**

Selector {

Property 1 : value;

Property 2 : value;

Property 3 : value;

}

For example:

h1

{

Color: red;

Text-align: center;

}

#unique

{

color: green;

}

* Selector: selects the element you want to target
* Always remains the same whether we apply internal or external styling
* There are few basic selectors like tags, id’s, and classes
* All forms this key-value pair
* Keys: properties(attributes) like color, font-size, background, width, height,etc
* Value: values associated with these properties

## **CSS Comment**

* Comments don’t render on the browser
* Helps to understand our code better and makes it readable.
* Helps to debug our code
* Two ways to  comment:
  + Single line

## **CSS How-To**

* There are 3 ways to write CSS in our HTML file.
  + Inline CSS
  + Internal CSS
  + External CSS
* Priority order
  + Inline > Internal > External

**Inline CSS**

* Before CSS this was the only way to apply styles
* Not an efficient way to write as it has a lot of redundancy
* Self-contained
* Uniquely applied on each element
* The idea of separation of concerns was lost
* Example:

<h3 style = “color:red”> Have a great day </h3>

<p style = “color:green”> I did this, I did that </p>

**Internal CSS**

* With the help of style tag, we can apply styles within the HTML file
* Redundancy is removed
* But the idea of separation of concerns still lost
* Uniquely applied on a single document
* Example:

<style>

H1{

Color:red;

}

</style>

<h3> Have a great day </h3>

**External CSS**

* With the help of <link> tag in the head tag, we can apply styles
* Reference is added
* File saved with .css extension
* Redundancy is removed
* The idea of separation of concerns is maintained
* Uniquely applied to each document
* Example:

<head>

<link rel= “stylesheet” type= “text/css” href= “name of the CSS file”>

</head>

h1{

color:red; //.css file

}

## **CSS Selectors**

* The selector is used to target elements and apply CSS
* Three simple selectors
  + Element Selector
  + Id Selector
  + Class Selector
* Priority of Selectors

## **CSS Colors**

* There are different colouring schemes in CSS
* **RGB**-This starts with RGB and takes 3 parameter
* **HEX**-Hex code starts with # and comprises of 6 numbers which are further divided into 3 sets
* **RGBA**-This starts with RGB and takes 4 parameter

**CSS Background**

* There are different ways by which CSS can have an effect on HTML elements
* Few of them are as follows:
  + Color – used to set the color of the background
  + Repeat – used to determine if the image has to repeat or not and if it is repeating then how it should do that
  + Image – used to set an image as the background
  + Position – used to determine the position of the image
  + Attachment – It basically helps in controlling the mechanism of scrolling.

## **CSS BoxModel**

* Every element in CSS can be represented using the BOX model
* It allows us to add a border and define space between the content
* It helps the developer to develop and manipulate the elements
* It consists of 4 edges
  + Content edge – It comprises of the actual content
  + Padding edge – It lies in between content and border edge
  + Border edge – Padding is followed by the border edge
  + Margin edge – It is an outside border and controls the margin of the element

**CODE:**

**MODULE 1:**

**# MANUAL NET ARCHITECTURE**

import warnings

warnings.filterwarnings('ignore')

import os

import glob

import numpy as np

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from PIL import Image

from tensorflow.keras.layers import Convolution2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Activation

from keras.callbacks import ModelCheckpoint

import matplotlib.pyplot as plt

glioma\_tumor = 'DATASET/TRAIN/glioma\_tumor'

meningioma\_tumor = 'DATASET/TRAIN/meningioma\_tumor'

Neurocitoma\_tumor = 'DATASET/TRAIN/Neurocitoma\_tumor'

pituitary\_tumor = 'DATASET/TRAIN/pituitary\_tumor'

def plot\_images(item\_dir, n=6):

all\_item\_dir = os.listdir(item\_dir)

item\_files = [os.path.join(item\_dir, file) for file in all\_item\_dir][:n]

plt.figure(figsize=(80, 40))

for idx, img\_path in enumerate(item\_files):

plt.subplot(3, n, idx+1)

img = plt.imread(img\_path)

plt.imshow(img, cmap='gray')

plt.axis('off')

plt.tight\_layout()

def image\_details\_print(data,path):

print('======== Images in: ', path)

for key,values in data.items():

print(key,':\t', values)

def images\_details(path):

files=[f for f in glob.glob(path + "\*\*/\*.\*", recursive=True)]

data={}

data['Images\_count']=len(files)

data['Min\_width']=10\*\*100

data['Max\_width']=0

data['Min\_height']=10\*\*100

data['Max\_height']=0

for f in files:

img=Image.open(f)

width,height=img.size

data['Min\_width']=min(width,data['Min\_width'])

data['Max\_width']=max(width, data['Max\_width'])

data['Min\_height']=min(height, data['Min\_height'])

data['Max\_height']=max(height, data['Max\_height'])

image\_details\_print(data,path)

print("")

print("TRAINING DATA FOR glioma\_tumor:")

print("")

images\_details(glioma\_tumor)

print("")

plot\_images(glioma\_tumor, 10)

print("")

print("TRAINING DATA FOR meningioma\_tumor:")

print("")

images\_details(meningioma\_tumor)

print("")

plot\_images(meningioma\_tumor, 10)

print("")

print("TRAINING DATA FOR Neurocitoma\_tumor:")

print("")

images\_details(Neurocitoma\_tumor)

print("")

plot\_images(Neurocitoma\_tumor, 10)

print("")

print("TRAINING DATA FOR pituitary\_tumor:")

print("")

images\_details(pituitary\_tumor)

print("")

plot\_images(pituitary\_tumor, 10)

train\_datagen=ImageDataGenerator(rescale=1./255,shear\_range=0.2,zoom\_range=0.2,horizontal\_flip=True)

training\_set=train\_datagen.flow\_from\_directory('DATASET/TRAIN',target\_size=(224,224),batch\_size=32,class\_mode='categorical')

test\_datagen=ImageDataGenerator(rescale=1./255)

test\_set=test\_datagen.flow\_from\_directory('DATASET/TEST',target\_size=(224,224),batch\_size=32,class\_mode='categorical')

Classifier=Sequential()

Classifier.add(Convolution2D(32,(3,3),input\_shape=(224,224,3),activation='relu'))

Classifier.add(MaxPooling2D(pool\_size=(2,2)))

Classifier.add(Flatten())

Classifier.add(Dense(38, activation='relu'))

Classifier.add(Dense(6, activation='softmax'))

Classifier.compile(optimizer='rmsprop',loss='categorical\_crossentropy',metrics=['accuracy'])

model\_path = "MANUAL.h5"

callbacks = [

ModelCheckpoint(model\_path, monitor='accuracy', verbose=1, save\_best\_only=True)

]

epochs = 10

batch\_size = 512

#### Fitting the model

history = Classifier.fit(

training\_set, steps\_per\_epoch=training\_set.samples // batch\_size,

epochs=epochs,

validation\_data=test\_set,validation\_steps=test\_set.samples // batch\_size,

callbacks=callbacks)

import matplotlib.pyplot as plt

def graph():

#Plot training & validation accuracy values

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

graph()

import matplotlib.pyplot as plt

def graph():

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

graph()

**MODULE 2:**

**## VGG ARCHITECTURE**

import warnings

warnings.filterwarnings('ignore')

import tensorflow

import tensorflow as tf

print(tf.\_\_version\_\_)

import keras

import keras.backend as K

from keras.models import Model

from keras.layers import Input, Dense, Conv2D, Conv3D, DepthwiseConv2D, SeparableConv2D, Conv3DTranspose

from keras.layers import Flatten, MaxPool2D, AvgPool2D, GlobalAvgPool2D, UpSampling2D, BatchNormalization

from keras.layers import Concatenate, Add, Dropout, ReLU, Lambda, Activation, LeakyReLU, PReLU

from time import time

import numpy as np

from keras.callbacks import ModelCheckpoint

from tensorflow.keras.callbacks import EarlyStopping

import warnings

warnings.filterwarnings('ignore')

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train=ImageDataGenerator(rescale=1./255,shear\_range=0.2,zoom\_range=0.2,horizontal\_flip=True,validation\_split = 0.2)

train\_data=train.flow\_from\_directory(directory = 'DATASET/TRAIN',target\_size=(224,224),

batch\_size=32,class\_mode='categorical')

test=ImageDataGenerator(rescale=1./255)

test\_data=test.flow\_from\_directory(directory = 'DATASET/TEST',target\_size=(224,224),

batch\_size=32,class\_mode='categorical')

def vgg(input\_shape, n\_classes):

input = Input(input\_shape)

x = Conv2D(64, 3, padding='same', activation='relu')(input)

x = Conv2D(64, 3, padding='same', activation='relu')(x)

x = MaxPool2D(2, strides=2, padding='same')(x)

x = Conv2D(128, 3, padding='same', activation='relu')(x)

x = Conv2D(128, 3, padding='same', activation='relu')(x)

x = MaxPool2D(2, strides=2, padding='same')(x)

x = Conv2D(256, 3, padding='same', activation='relu')(x)

x = Conv2D(256, 3, padding='same', activation='relu')(x)

x = Conv2D(256, 3, padding='same', activation='relu')(x)

x = MaxPool2D(2, strides=2, padding='same')(x)

x = Conv2D(512, 3, padding='same', activation='relu')(x)

x = Conv2D(512, 3, padding='same', activation='relu')(x)

x = Conv2D(512, 3, padding='same', activation='relu')(x)

x = MaxPool2D(2, strides=2, padding='same')(x)

x = Conv2D(512, 3, padding='same', activation='relu')(x)

x = Conv2D(512, 3, padding='same', activation='relu')(x)

x = Conv2D(512, 3, padding='same', activation='relu')(x)

x = MaxPool2D(2, strides=2, padding='same')(x)

x = Flatten()(x)

x = Dense(4096, activation='relu')(x)

x = Dense(4096, activation='relu')(x)

output = Dense(n\_classes, activation='softmax')(x)

model = Model(input, output)

model.compile(optimizer='Adam',loss='categorical\_crossentropy',metrics=['accuracy',tensorflow.keras.metrics.Precision()])

return model

input\_shape = 224, 224, 3

n\_classes = 6

K.clear\_session()

model = vgg(input\_shape, n\_classes)

model.summary()

model\_path = "ALEXNET.h5"

from keras.callbacks import ModelCheckpoint

M = ModelCheckpoint(model\_path, monitor='accuracy', verbose=1, save\_best\_only=True)

epochs = 50

batch\_size = 512

#### Fitting the model

history = model.fit(

train\_data, steps\_per\_epoch=train\_data.samples // batch\_size,

epochs=epochs,

validation\_data=test\_data,validation\_steps=test\_data.samples // batch\_size,

callbacks=[M])

history.history.keys()

import matplotlib.pyplot as plt

import numpy as np

plt.figure(figsize=(20, 8))

plt.plot(history.history['accuracy'])

for i in range(epochs):

if i%5 == 0:

plt.annotate(np.round(history.history['accuracy'][i]\*100,2),xy=(i,history.history['accuracy'][i]))

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.show()

plt.figure(figsize=(20, 8))

plt.plot(history.history['loss'])

for i in range(epochs):

if i%5 == 0:

plt.annotate(np.round(history.history['loss'][i]\*100,2),xy=(i,history.history['loss'][i]))

plt.title('Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.show()

**MODULE 3:**

**# LENET ARCHITECTURE**

import warnings

warnings.filterwarnings('ignore')

# import os

# os.environ['KMP\_DUPLICATE\_LIB\_OK']="TRUE"

import tensorflow

import tensorflow as tf

print(tf.\_\_version\_\_)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D

from tensorflow.keras.layers import Convolution2D

from tensorflow.keras.layers import MaxPool2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Dropout

from tensorflow.keras.layers import Activation

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train=ImageDataGenerator(rescale=1./255,shear\_range=0.2,zoom\_range=0.2,horizontal\_flip=True,validation\_split = 0.2)

train\_data=train.flow\_from\_directory(directory = 'DATASET/TRAIN',target\_size=(224,224),

batch\_size=32,class\_mode='categorical')

test=ImageDataGenerator(rescale=1./255)

test\_data=test.flow\_from\_directory(directory = 'DATASET/TEST',target\_size=(224,224),

batch\_size=32,class\_mode='categorical')

MODEL=Sequential()

MODEL.add(Convolution2D(filters=32, kernel\_size=(3,3), strides=(3,3), input\_shape=(224,224,3), padding=('valid'), activation='relu'))

MODEL.add(MaxPooling2D(pool\_size=(2,2), strides=(2,2), padding='valid'))

MODEL.add(Convolution2D(filters=128, kernel\_size=(3,3), strides=(3,3), padding=('valid'), activation='relu'))

MODEL.add(MaxPooling2D(pool\_size=(2,2), strides=(2,2), padding='valid'))

MODEL.add(Flatten())

MODEL.add(Dense(256, activation='relu'))

MODEL.add(Dense(6, activation='softmax'))

OPT = tensorflow.keras.optimizers.Adam(0.001)

MODEL.compile(optimizer=OPT,loss='categorical\_crossentropy',metrics=["accuracy", tensorflow.keras.metrics.Precision(), tensorflow.keras.metrics.Recall()])

MODEL.summary()

model\_path = "LENET.h5"

from tensorflow.keras.callbacks import ModelCheckpoint

M = ModelCheckpoint(model\_path, monitor='accuracy', verbose=1, save\_best\_only=True, mode='max')

epochs = 100

batch\_size = 256

WORKING = MODEL.fit\_generator(

train\_data, steps\_per\_epoch=train\_data.samples // batch\_size,

epochs=epochs,

validation\_data=test\_data,validation\_steps=test\_data.samples // batch\_size,

callbacks=[M])

WORKING.history.keys()

import matplotlib.pyplot as plt

import numpy as np

plt.figure(figsize=(20, 8))

plt.plot(WORKING.history['accuracy'])

for i in range(epochs):

if i%5 == 0:

plt.annotate(np.round(WORKING.history['accuracy'][i]\*100,2),xy=(i,WORKING.history['accuracy'][i]))

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.show()

import matplotlib.pyplot as plt

plt.figure(figsize=(20, 8))

plt.plot(WORKING.history['loss'])

for i in range(epochs):

if i%5 == 0:

plt.annotate(np.round(WORKING.history['loss'][i]\*100,2),xy=(i,WORKING.history['loss'][i]))

plt.title('Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.show()

**DEPLOY:**

from django.shortcuts import render, redirect

from django.urls import reverse\_lazy

from .forms import UserImageForm

from django.contrib.auth.views import LoginView, PasswordResetView, PasswordChangeView

from django.contrib import messages

from django.contrib.messages.views import SuccessMessageMixin

from django.views import View

from django.contrib.auth.decorators import login\_required

from django.contrib.auth import logout as auth\_logout

import numpy as np

import joblib

from .forms import RegisterForm, LoginForm, UpdateUserForm, UpdateProfileForm

from django.contrib.auth import authenticate,login,logout

from .models import UserImageModel

import numpy as np

from tensorflow import keras

from PIL import Image,ImageOps

import pyttsx3

import time

def home(request):

    return render(request, 'users/home.html')

@login\_required(login\_url='users-register')

def index(request):

    return render(request, 'app/index.html')

class RegisterView(View):

    form\_class = RegisterForm

    initial = {'key': 'value'}

    template\_name = 'users/login.html'

    def dispatch(self, request, \*args, \*\*kwargs):

        # will redirect to the home page if a user tries to access the register page while logged in

        if request.user.is\_authenticated:

            return redirect(to='/')

        # else process dispatch as it otherwise normally would

        return super(RegisterView, self).dispatch(request, \*args, \*\*kwargs)

    def get(self, request, \*args, \*\*kwargs):

        form = self.form\_class(initial=self.initial)

        return render(request, self.template\_name, {'form': form})

    def post(self, request, \*args, \*\*kwargs):

        form = self.form\_class(request.POST)

        if form.is\_valid():

            form.save()

            username = form.cleaned\_data.get('username')

            messages.success(request, f'Account created for {username}')

            return redirect(to='login')

        return render(request, self.template\_name, {'form': form})

# Class based view that extends from the built in login view to add a remember me functionality

class CustomLoginView(LoginView):

    form\_class = LoginForm

    def form\_valid(self, form):

        remember\_me = form.cleaned\_data.get('remember\_me')

        if not remember\_me:

            # set session expiry to 0 seconds. So it will automatically close the session after the browser is closed.

            self.request.session.set\_expiry(0)

            # Set session as modified to force data updates/cookie to be saved.

            self.request.session.modified = True

        # else browser session will be as long as the session cookie time "SESSION\_COOKIE\_AGE" defined in settings.py

        return super(CustomLoginView, self).form\_valid(form)

class ResetPasswordView(SuccessMessageMixin, PasswordResetView):

    template\_name = 'users/password\_reset.html'

    email\_template\_name = 'users/password\_reset\_email.html'

    subject\_template\_name = 'users/password\_reset\_subject'

    success\_message = "We've emailed you instructions for setting your password, " \

                      "if an account exists with the email you entered. You should receive them shortly." \

                      " If you don't receive an email, " \

                      "please make sure you've entered the address you registered with, and check your spam folder."

    success\_url = reverse\_lazy('users-home')

class ChangePasswordView(SuccessMessageMixin, PasswordChangeView):

    template\_name = 'users/change\_password.html'

    success\_message = "Successfully Changed Your Password"

    success\_url = reverse\_lazy('users-home')

from .models import Profile

def profile(request):

    user = request.user

    # Ensure the user has a profile

    if not hasattr(user, 'profile'):

        Profile.objects.create(user=user)

    if request.method == 'POST':

        user\_form = UpdateUserForm(request.POST, instance=request.user)

        profile\_form = UpdateProfileForm(request.POST, request.FILES, instance=request.user.profile)

        if user\_form.is\_valid() and profile\_form.is\_valid():

            user\_form.save()

            profile\_form.save()

            messages.success(request, 'Your profile is updated successfully')

            return redirect(to='users-profile')

    else:

        user\_form = UpdateUserForm(instance=request.user)

        profile\_form = UpdateProfileForm(instance=request.user.profile)

    return render(request, 'users/profile.html', {'user\_form': user\_form, 'profile\_form': profile\_form})

from . models import UserImageModel

from . import forms

from .forms import UserImageForm

def Deploy\_8(request):

    print("HI")

    if request.method == "POST":

        form = forms.UserImageForm(files=request.FILES)

        if form.is\_valid():

            print('HIFORM')

            form.save()

        obj = form.instance

        result1 = UserImageModel.objects.latest('id')

        models = keras.models.load\_model('E:/IYYAPPAN/ITPDL23-FINAL/ITPDL23-FINAL CODING/Deploy/Project/App/Model.h5')

        data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

        image = Image.open("E:/IYYAPPAN/ITPDL23-FINAL/ITPDL23-FINAL CODING/Deploy/Project/" + str(result1)).convert("RGB")

        size = (224, 224)

        image = ImageOps.fit(image, size, Image.ANTIALIAS)

        image\_array = np.asarray(image)

        normalized\_image\_array = (image\_array.astype(np.float32) / 127.0) - 1

        data[0] = normalized\_image\_array

        classes = ["glioma\_tumor","meningioma\_tumor","Neurocitoma\_tumor","Normal","pituitary\_tumor","Schwannoma\_tumor"]

        prediction = models.predict(data)

        idd = np.argmax(prediction)

        a = (classes[idd])

        if a == "glioma\_tumor":

             b ='This Image Detected Glioma\_tumor'

        elif a == "meningioma\_tumor":

            b ='This Image Detected Meningioma\_tumor'

        elif a == "Neurocitoma\_tumor":

            b ='This Image Detected Neurocitoma\_tumor'

        elif a == "Normal":

            b ='This Image Detected Normal'

        elif a == "pituitary\_tumor":

            b ='This Image Detected Pituitary\_tumor'

        elif a == "Schwannoma\_tumor":

            b ='This Image Detected Schwannoma\_tumor'

        else:

            b = 'WRONG INPUT'

        data = UserImageModel.objects.latest('id')

        data.label = a

        data.save()

        # engine = pyttsx3.init()

        # rate = engine.getProperty('rate')

        # engine.setProperty('rate', rate - 10)  # Decrease rate by 50 (default rate is typically around 200)

        # engine.say(a)

        # engine.runAndWait()

        # text\_to\_speech(a, delay=7)

        return render(request, 'App/output.html',{'form':form,'obj':obj,'predict':b})

    else:

        form = forms.UserImageForm()

    return render(request, 'App/model.html',{'form':form})

def Database(request):

    models = UserImageModel.objects.all()

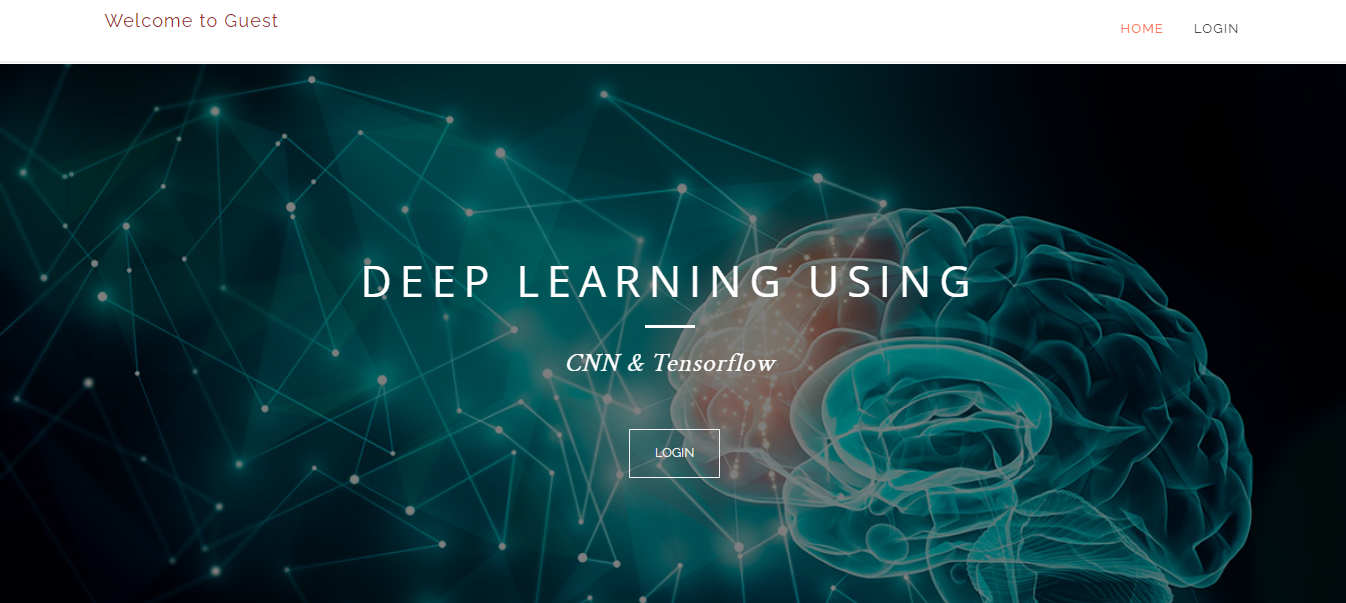
    return render(request, 'App/Database.html', {'models': models})

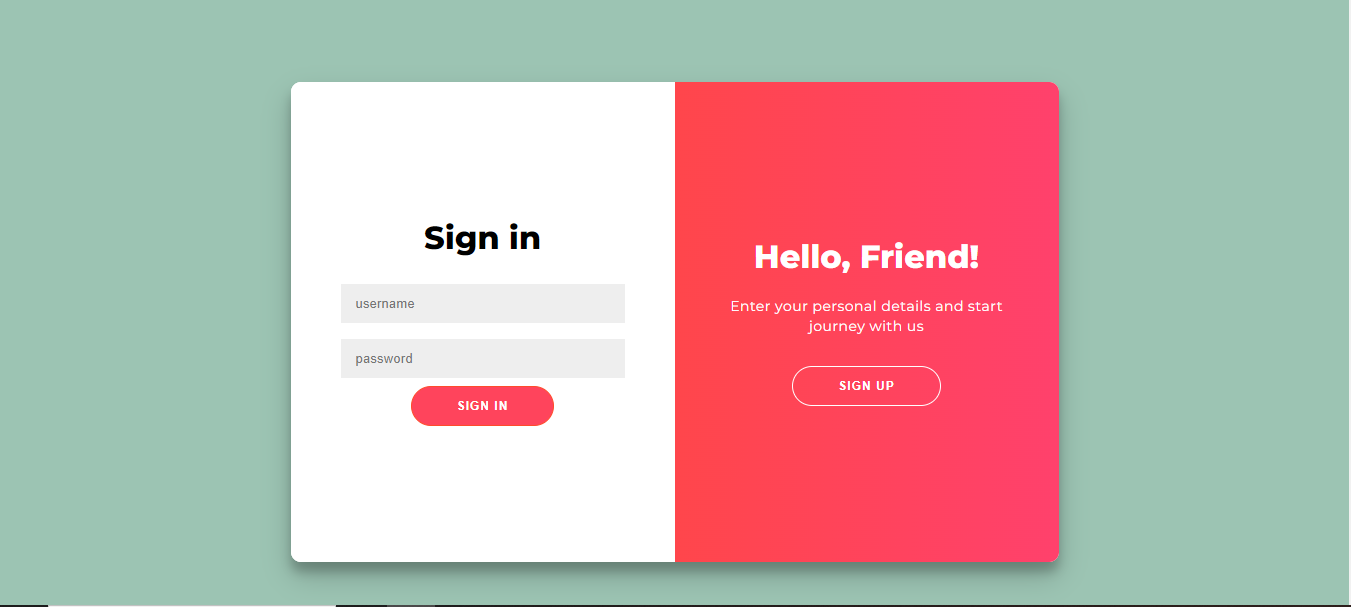
def logout\_view(request):

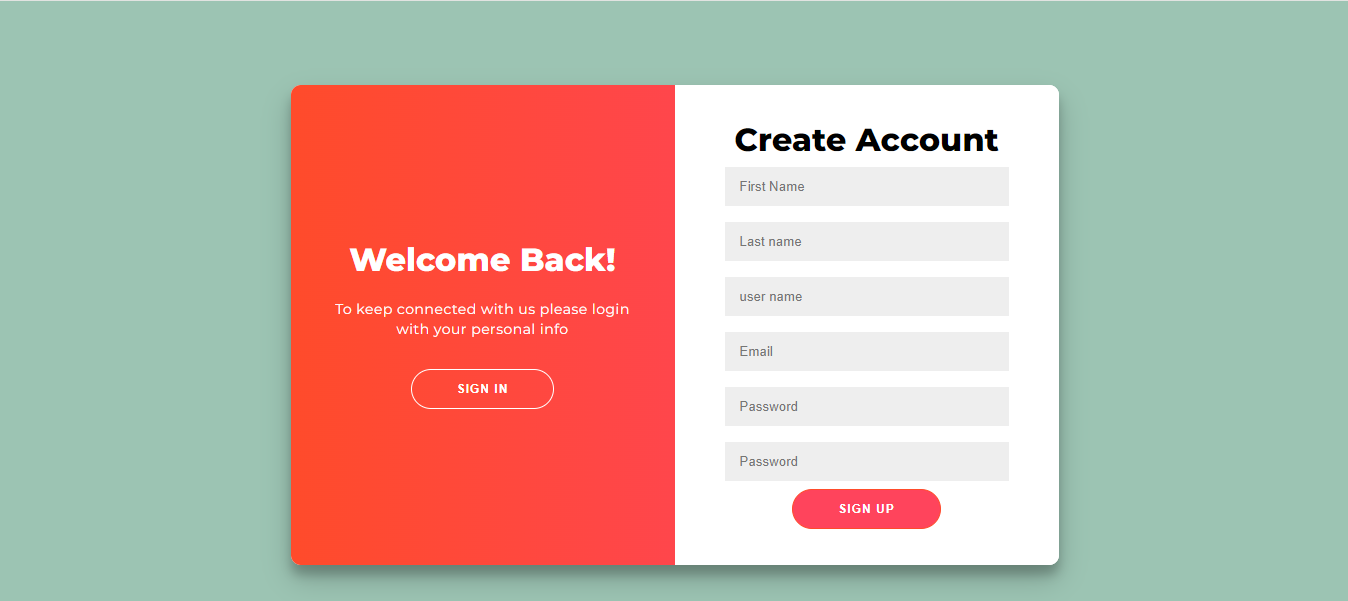
    auth\_logout(request)

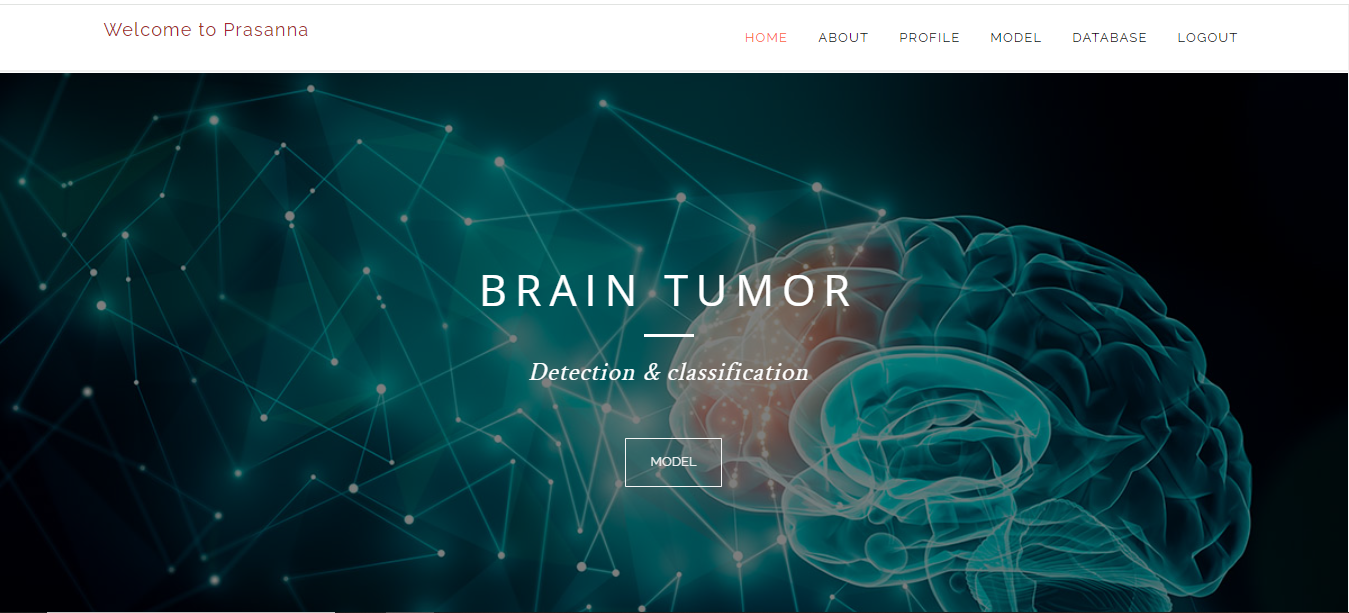
    return redirect('/')

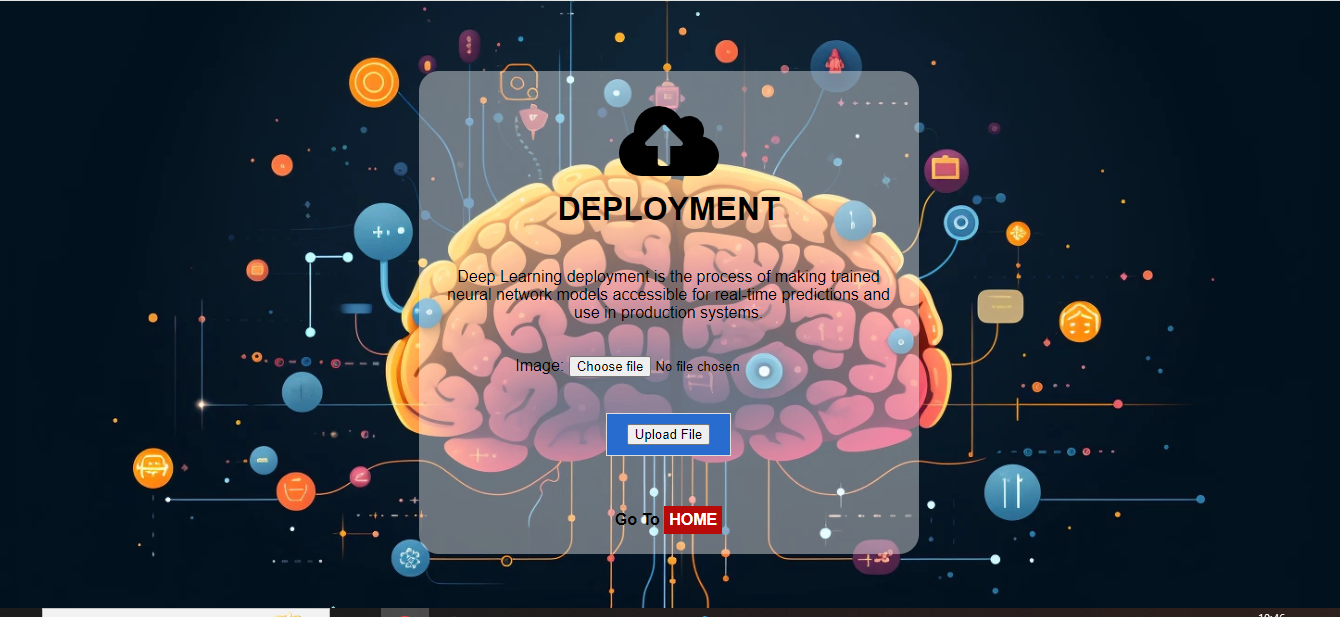
**OUTPUT SCREEN:**

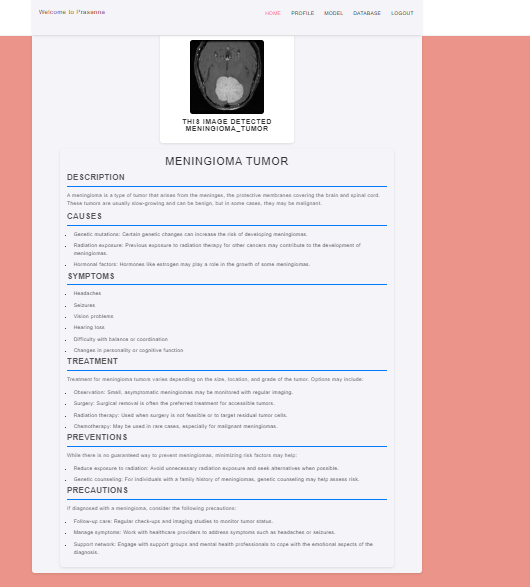
****

****

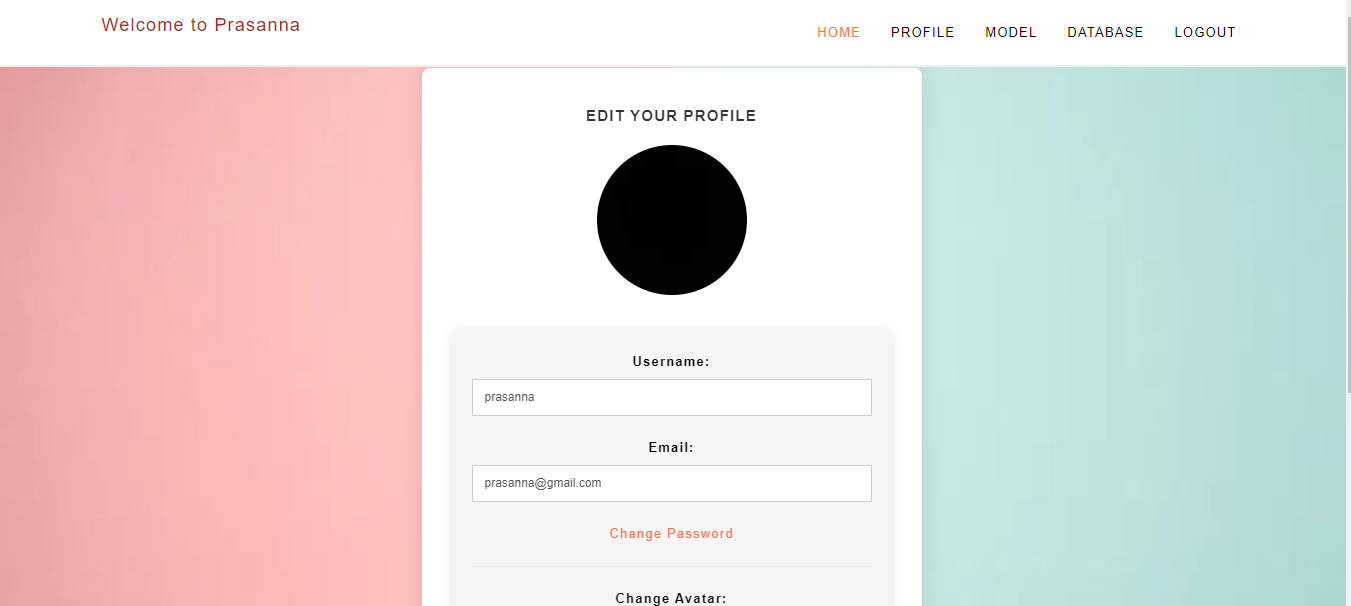
****

****

****

****

****

****

**Conclusion:**

The NeuroFusion approach significantly enhances brain tumor diagnosis by leveraging advanced Convolutional Neural Network (CNN) architectures. By integrating these sophisticated models, NeuroFusion not only improves the accuracy and efficiency of tumor detection but also provides a robust framework for analyzing complex brain imaging data. This innovation leads to more precise and timely diagnoses, ultimately contributing to better patient outcomes and paving the way for future advancements in neuroimaging and artificial intelligence.

**Future Work:**

* We can deploy the model in any cloud-based system.
* We can implement more than three architectures
* We can connect this model to the hardware.