**DEEP CONVOLUTION NEURAL NETWORK FOR BIG DATA MEDICAL IMAGE CLASSIFICATION**

**ABSTRACT**

Deep learning is one of the most unexpected machine learning techniques which is being used in many applications like image classiﬁcation, image analysis, clinical archives and object recognition. With an extensive utilization of digital images as information in the hospitals, the archives of medical images are growing exponentially. Digital images play a vigorous role in predicting the patient disease intensity and there are vast applications of medical images in diagnosis and investigation. Due to recent developments in imaging technology, classifying medical images in an automatic way is an open research problem for researchers of computer vision. For classifying the medical images according to their relevant classes a most suitable classiﬁer is most important. Where we are proposing our model where the algorithm is trained for classifying medical images by deep learning technique. A pre-trained deep convolution neural network (GoogleNet) is used that which can classifies the various medical images for various body organs. This method of image classiﬁcation is beneﬁcial to predict the appropriate class or category of unknown images. The results of the experiment exhibit that our method is best suited to classify various medical images.

**Keywords:** Medical image classiﬁcation, pre-trained DCNN, convolution neural network, deep learning

**CHAPTER 1**

**INTRODUCTION**

Image classification is the primary domain, in which deep neural networks play the most important role of medical image analysis. The image classification accepts the given input images and produces output classification for identifying whether the disease is present or not. In our model we mainly classify the different types of organs and predict the accuracy.

One of the most imperative problems faced in the domain area of image recognition is the classification of medical images. The major intention of medical image classification is to classify medical images into several elements to assist medical practitioners or physicists in diagnosing disease. Hence, medical image classification is split into two steps. The first and foremost step of medical image classification is to extract the essential features from the acquired input image. The second step in medical image classification is utilizing the features to construct models that classify the image data set. In the recent past, medical practitioners customarily utilized their specialized experience to extract features so that classification of medical images could be performed into several classes. However, this manual medical image classification was found to be highly cumbersome and time consuming.

Medical image classification involves the process of segregating medical-related information into a useful form. Classification of medical images is based on placing image pixels with similar values into groups. With the placement of similar values into groups, common pixels are identified and are denoted by these pixels. Hence, a correctly classified image usually denotes the areas on the ground that share specific features as specified in the classification scheme.

Image classification is where a computer can analyse an image and identify the ‘class’ the image falls under. (Or a probability of the image being part of a ‘class’.) A class is essentially a label, for instance, ‘car’, ‘animal’, ‘building’ and so on. For example, you input an image of a sheep. Image classification is the process of the computer analysing the image and telling you it’s a sheep. (Or the probability that it’s a sheep.) For us, classifying images is no big deal. But it’s a perfect example of Moravec’s paradox when it comes to machines. (That is, the things we find easy are difficult for AI.)

Early image classification relied on raw pixel data. This meant that computers would break down images into individual pixels. The problem is that two pictures of the same thing can look very different. They can have different backgrounds, angles, poses, and etcetera. This made it quite the challenge for computers to correctly ‘see’ and categorise images.

Deep learning is a type of machine learning; a subset of artificial intelligence (AI) that allows machines to learn from data. Deep learning involves the use of computer systems known as neural networks. In neural networks, the input filters through hidden layers of nodes. These nodes each process the input and communicate their results to the next layer of nodes. This repeats until it reaches an output layer, and the machine provides its answer.

There are different types of neural networks based on how the hidden layers work. Image classification with deep learning most often involves convolutional neural networks, or CNNs. In CNNs, the nodes in the hidden layers don’t always share their output with every node in the next layer (known as convolutional layers). Deep learning allows machines to identify and extract features from images. This means they can learn the features to look for in images by analysing lots of pictures. So, programmers don’t need to enter these filters by hand.

Image classification has a few uses — and vast potential as it grows in reliability. Here are just a few examples of what makes it useful. Self-driving cars use image classification to identify what’s around them. I.e. trees, people, traffic lights and so on. Image classification can also help in healthcare. For instance, it could analyse medical images and suggest whether they classify as depicting a symptom of illness. Or, for example, image classification could help people organise their photo collections.

**CHAPTER 2**

**LITERATURE REVIEW**

**[1] Q. Zhu, B. Du, and P. Yan:** Accurate segmentation of the prostate from magnetic resonance (MR) images provides useful information for prostate cancer diagnosis and treatment. However, automated prostate segmentation from 3D MR images still faces several challenges. The complex background texture and large variation in size, shape and intensity distribution of the prostate itself make segmentation even further complicated. Since large-scale dataset is one of the critical components for the success of deep learning, lack of sufficient training data makes it difficult to fully train complex CNNs. To tackle the above challenges, in this paper boundary-weighted domain adaptive neural network (BOWDA-Net) is proposed. To make the network more sensitive to the boundaries during segmentation, a boundary-weighted segmentation loss (BWL) is proposed.

**Summary:** In this paper boundary-weighted domain adaptive neural network (BOWDA-Net) is proposed. To make the network more sensitive to the boundaries during segmentation, a boundary-weighted segmentation loss (BWL) is proposed. . Furthermore, an advanced boundary-weighted transfer leaning approach is introduced to address the problem of small medical imaging datasets. We evaluate our proposed model on the publicly available MICCAI 2012 Prostate MR Image Segmentation (PROMISE12) challenge dataset.

**[2] Q.Zhu, B.Du, P.Yan, H.Lu, and L.Zhang:** Bladder wall segmentation from Magnetic Resonance (MR) images plays an important role in diagnosis. Since the thickness of the bladder wall is a key indication of bladder cancer. There are several methods that have been used for bladder wall segmentation, such as level sets and Active Shape Model (ASM). However, the weak boundaries, the artifacts inside bladder lumen and the complex background outside the bladder wall make the bladder wall segmentation very challenging. To overcome these difficulties and obtain accurate bladder walls, in this paper, a shape prior constrained particle swarm optimization (SPC-PSO) model is proposed to segment the inner and outer boundaries of the bladder wall. The bladder walls are divided into two categories: strong boundaries and weak boundaries by the proposed model.

**Summary**: In this paper, a shape prior constrained particle swarm optimization (SPC-PSO) model is proposed to segment the inner and outer boundaries of the bladder wall. The bladder walls are divided into two categories: strong boundaries and weak boundaries by the proposed model. For the strong boundaries, the proposed model can reserve it. For the weak boundaries, the model applies the shape prior to guide the process of segmentation**.**

**[3] Q. Zhu, B. Du, B. Turkbey, P. Choyke, and P. Yan:** Segmentation of the prostate from Magnetic Resonance Imaging (MRI) plays an important role in prostate cancer diagnosis. However, the lack of clear boundary and significant variation of prostate shapes and appearances make the automatic segmentation very challenging. In the past several years, approaches based on deep learning technology have made significant progress on prostate segmentation. However, those approaches mainly paid attention to features and contexts within each single slice of a 3D volume. As a result, this kind of approaches faces many difficulties when segmenting the base and apex of the prostate due to the limited slice boundary information. To tackle this problem, in this paper, we propose a deep neural network with bidirectional convolutional recurrent layers for MRI prostate image segmentation.

**Summary:** A deep neural network is proposed with bidirectional convolutional recurrent layers for MRI prostate image segmentation. In addition to utilizing the interslice contexts and features, the proposed model also treats prostate slices as a data sequence and utilizes the interslice contexts to assist segmentation. The experimental results show that the proposed approach achieved significant segmentation improvement compared to other reported methods.

**[4] K.Kranthi Kumar and T.V.Gopal:** Content Based Image Retrieval (CBIR) is a prominent research area in effective retrieval and management process for large image databases. Which was a bottleneck in reducing semantic gap issue to solve, many approaches have been proposed. Among them, Relevance Feedback (RF) is a technique absorbed into CBIR systems to improve retrieval accuracy using user given feedback. One of the traditional methods to enact relevance feedback is Feature Reweighting (FRW), it is useful technique to enhance retrieval performance based on the acquired feedback from user. The assumption for previous FRW approaches are that the length of feature vectors for images are fixed and use only the information from the set of images send back in the early query result for feature reweighting. **Summary:** In this article, examined systematically the proposed system with various weight update strategies and compared output retrieval results and proposed a new self-order feature reweighting approach in CBIR to reduce semantic gap using relevance feedback which we experimented with COREL database with 25 different categories and each category containing 100 number of relevant images.

**[5] R.Ashraf, K.B.Bajwa, and T.Mahmood:** Segmentation of the images is considered as a solution but there isn’t any technique which can guarantee the object extraction in a robust way. Another limitation of the segmentation is that, most of the image segmentation techniques are very slow and still their results are not reliable. To overcome these problems a Bandelets transform based image representation technique is presented in this paper, which reliably returns the information about the major objects found in an image. For image retrieval purposes Support Vector Machine are applied and the performance of the system is evaluated on three standard data sets used in the domain of content based image retrieval.

**Summary:** To overcome the problems which are in the existing method, a Bandelets transform based image representation technique is presented in this paper, which reliably returns the information about the major objects found in an image. For image retrieval purposes Support Vector Machine are applied and the performance of the system is evaluated on three standard data sets used in the domain of content based image retrieval.

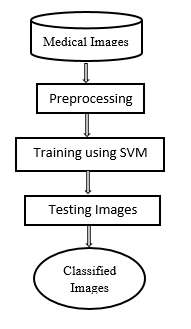
**[6] G.Wu, W.Lu, G.Gao, C.Zhao, and J.Liu:** Deep learning has been successfully applied to visual tracking due to its powerful feature learning characteristic. However, existing deep learning trackers rely on single observation model and focus on the holistic representation of the tracking object. When occlusion occurs, the trackers suffer from the contaminated features obtained in occluded areas. In this paper, we propose a regional deep learning tracker that observes the target by multiple sub-regions and each region is observed by a deep learning model. In particular, we devise a stable factor, modelled as a hidden variable of the Factorial Hidden Markov Model, to characterize the stability of these sub-models. The stability indicator not only provides a confidence degree for the response score of each model during inference stage, but also determines the online training criteria for each deep learning model.

**Summary:** When occlusion occurs, the trackers suffer from the contaminated features obtained in occluded areas. In this paper, we propose a regional deep learning tracker that observes the target by multiple sub-regions and each region is observed by a deep learning model. In particular, we devise a stable factor, modelled as a hidden variable of the Factorial Hidden Markov Model, to characterize the stability of these sub-models. The stability indicator not only provides a confidence degree for the response score of each model during inference stage, but also determines the online training criteria for each deep learning model.

**CHAPTER 3**

**EXISTING METHOD**

This model emphasizes an existing method that which is designed using the machine learning architecture which is used to classify the various medical images. With an extensive utilization of digital images as information in the hospitals, the archives of medical images are growing exponentially. Digital images play a vigorous role in predicting the patient disease intensity and there are vast applications of medical images in diagnosis and investigation. To make this in the easier way Support Vector Machine (SVM) is used that which can classifies the various medical images for various body organs. The block diagram of the existing method is shown in the below figure.



**Fig 1. Block diagram of existing method**

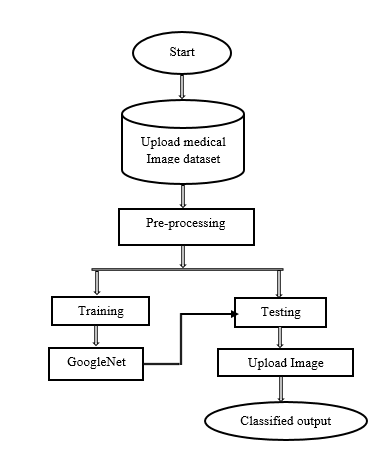
**Disadvantages:**

* Low efficiency.
* Time consuming.
* High complexities.
* No accurate classification

**CHAPTER 4**

**PROPOSED METHOD**

The proposed model emphasizes a deep network architecture which is used to classify the various medical images. With an extensive utilization of digital images as information in the hospitals, the archives of medical images are growing exponentially. Digital images play a vigorous role in predicting the patient disease intensity and there are vast applications of medical images in diagnosis and investigation. Hence, we are proposing our model where the algorithm is trained for classifying medical images by deep learning technique. A pre-trained deep convolution neural network (GoogleNet) is used that which can classifies the various medical images for various body organs. The block diagram of the proposed model is shown in the below figure.



**Fig 2. Block diagram of proposed method**

**ADVANTAGES**

* High efficiency.
* Time Saving.
* Inexpensive.
* Low complexities.

**APPLICATIONS**

* Useful for the hospitals to detect brain strokes using this application.

**SYSTEM SPECIFICATIONS**

**HARDWARE & SOFTWARE REQUIREMENTS**

# **H/W Configuration:**

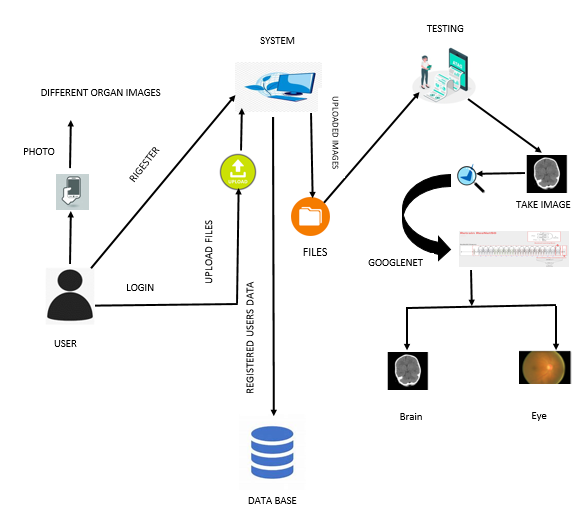
# Processor : I3/Intel Processor

* Hard Disk : 160GB
* RAM : 8Gb

**S/W Configuration:**

* Operating System : Windows 7/8/10 .
* Server side Script : HTML, CSS & JS.
* IDE : Pycharm.
* Libraries Used : Numpy, IO, OS, Flask, keras.
* Technology : Python 3.6+.

**ARCHITECTURE**



**SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS**

Installing Python:

1. To download and install Python visit the official website of Python <https://www.python.org/downloads/> and choose your version.



1. Once the download is complete, run the exe for install Python. Now click on Install Now.
2. You can see Python installing at this point.
3. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

Installing PyCharm:

1. To download PyCharm visit the website <https://www.jetbrains.com/pycharm/download/> and Click the "DOWNLOAD" link under the Community Section.



1. Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
2. On the next screen, Change the installation path if required. Click “Next”.
3. On the next screen, you can create a desktop shortcut if you want and click on “Next”.
4. Choose the start menu folder. Keep selected Jet Brains and click on “Install”.
5. Wait for the installation to finish.
6. Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.
7. After you click on "Finish," the Following screen will appear.



9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. The prompt will get open, with specified path, type “pip install package name” which you want to install (like numpy, pandas, seaborn, scikit-learn, matplotlib.pyplot)

Ex: pip install numpy



**MODULES**

**System**

**User**

**1. System:**

* 1. **Create Dataset:**

The dataset containing images of the desired objects to be recognize is split into training and testing dataset with the test size of 20-30%.

**1.2 Pre-processing:**

Resizing and reshaping the images into appropriate format to train our model. **1.3Training:**

Use the pre-processed training dataset is used to train our model using CNN algorithm.

**2. User:**

**2.1 Register**

The user needs to register and the data stored in MySQL database**.**

**2.2 Login**

A registered user can login using the valid credentials to the website to use a application.

**2.1 About-Project**

In this application, we have successfully created an application which takes to classify the images.

**2.2 Upload Image**

The user has to upload an image which needs to be classify the images.

**2.3 Prediction**

The results of our model is displayed as either Rice Blast, Leaf Blight, Healthy & Brown Spot.

**2.6 Logout**

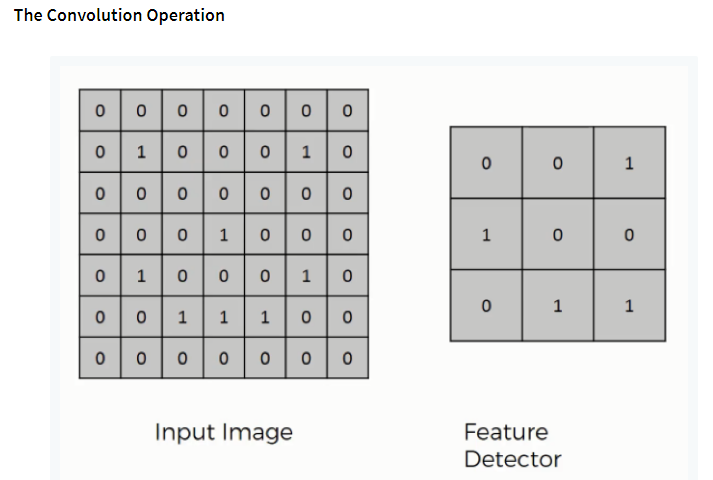
Once the prediction is over, the user can logout of the application.

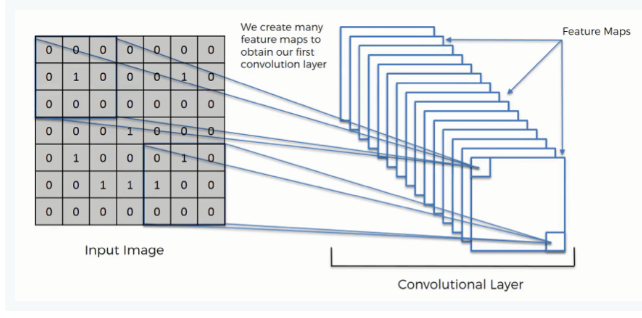
**ALGORITHM**

**1. Convolutional Neural Network**

**Step1: convolutional operation**

The first building block in our plan of attack is convolution operation. In this step, we will touch on feature detectors, which basically serve as the neural network's filters. We will also discuss feature maps, learning the parameters of such maps, how patterns are detected, the layers of detection, and how the findings are mapped out.

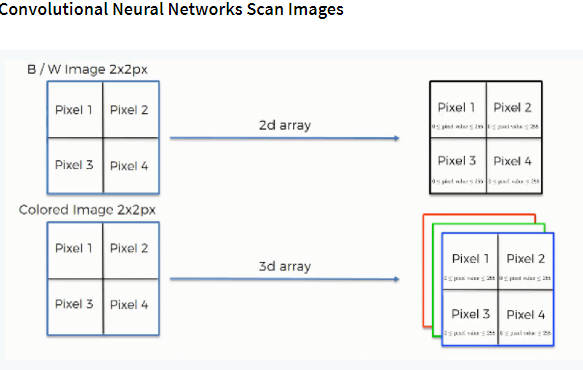




**Step (1b): ReLU Layer**

The second part of this step will involve the Rectified Linear Unit or Relook. We will cover Relook layers and explore how linearity functions in the context of Convolutional Neural Networks.

Not necessary for understanding CNN's, but there's no harm in a quick lesson to improve your skills.



**Step 2: Pooling Layer**

In this part, we'll cover pooling and will get to understand exactly how it generally works. Our nexus here, however, will be a specific type of pooling; max pooling. We'll cover various approaches, though, including mean (or sum) pooling. This part will end with a demonstration made using a visual interactive tool that will definitely sort the whole concept out for you.

**Step 3: Flattening**

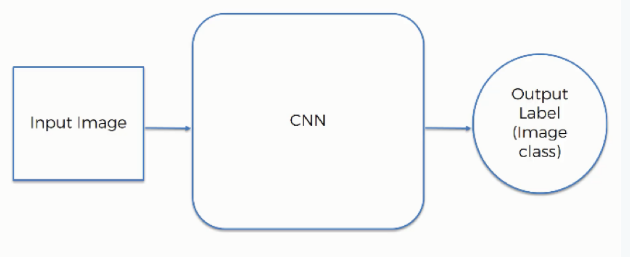
This will be a brief breakdown of the flattening process and how we move from pooled to flattened layers when working with Convolutional Neural Networks.

**Step 4: Full Connection**

In this part, everything that we covered throughout the section will be merged together. By learning this, you'll get to envision a fuller picture of how Convolutional Neural Networks operate and how the "neurons" that are finally produced learn the classification of images.

**Summary**

In the end, we'll wrap everything up and give a quick recap of the concept covered in the section. If you feel like it will do you any benefit (and it probably will), you should check out the extra tutorial in which Soft ax and Cross-Entropy are covered. It's not mandatory for the course, but you will likely come across these concepts when working with Convolutional Neural Networks and it will do you a lot of good to be familiar with them.



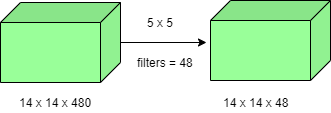
**GOOGLENET:**

Google Net (or Inception V1) was proposed by research at Google (with the collaboration of various universities) in 2014 in the research paper titled “Going Deeper with Convolutions”. This architecture was the winner at the ILSVRC 2014 image classification challenge. It has provided a significant decrease in error rate as compared to previous winners AlexNet (Winner of ILSVRC 2012) and ZF-Net (Winner of ILSVRC 2013) and significantly less error rate than VGG (2014 runner up). This architecture uses techniques such as 1×1 convolutions in the middle of the architecture and global average pooling.

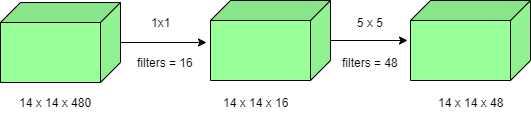
**Features of GoogleNet:**

The GoogLeNet architecture is very different from previous state-of-the-art architectures such as AlexNet and ZF-Net. It uses many different kinds of methods such as 1×1 convolution and global average pooling that enables it to create deeper architecture. In the architecture, we will discuss some of these methods:

* **1×1 convolution:** The inception architecture uses 1×1 convolution in its architecture. These convolutions used to decrease the number of parameters (weights and biases) of the architecture. By reducing the parameters we also increase the depth of the architecture. Let’s look at an example of a 1×1 convolution below:
  + For Example, If we want to perform 5×5 convolution having 48 filters without using 1×1 convolution as intermediate:



* Total Number of operations : (14 x 14 x 48) x (5 x 5 x 480) = 112.9 M
  + With 1×1 convolution :



* (14 x 14 x 16) x (1 x 1 x 480) + (14 x 14 x 48) x (5 x 5 x 16) = 1.5M + 3.8M = 5.3M which is much smaller than 112.9M.
* **Global Average Pooling:**

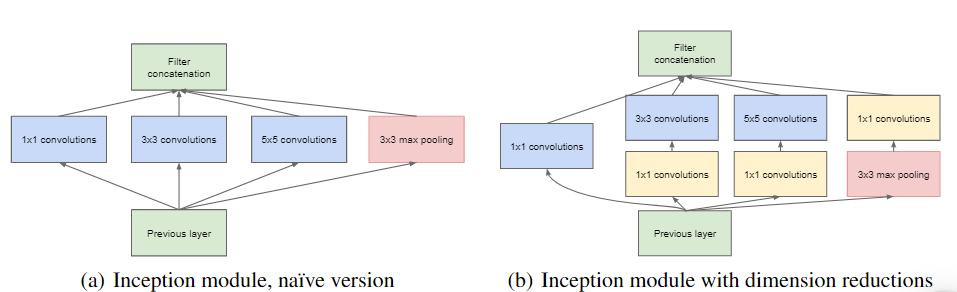
In the previous architecture such as AlexNet, the fully connected layers are used at the end of the network. These fully connected layers contain the majority of parameters of many architectures that causes an increase in computation cost.

In GoogLeNet architecture, there is a method called global average pooling is used at the end of the network. This layer takes a feature map of 7×7 and averages it to 1×1. This also decreases the number of trainable parameters to 0 and improves the top-1 accuracy by 0.6%

* **Inception Module:**

The inception module is different from previous architectures such as AlexNet, ZF-Net. In this architecture, there is a fixed convolution size for each layer.

In the Inception module 1×1, 3×3, 5×5 convolution and 3×3 max pooling performed in a parallel way at the input and the output of these are stacked together to generated final output. The idea behind that convolution filters of different sizes will handle objects at multiple scale better.

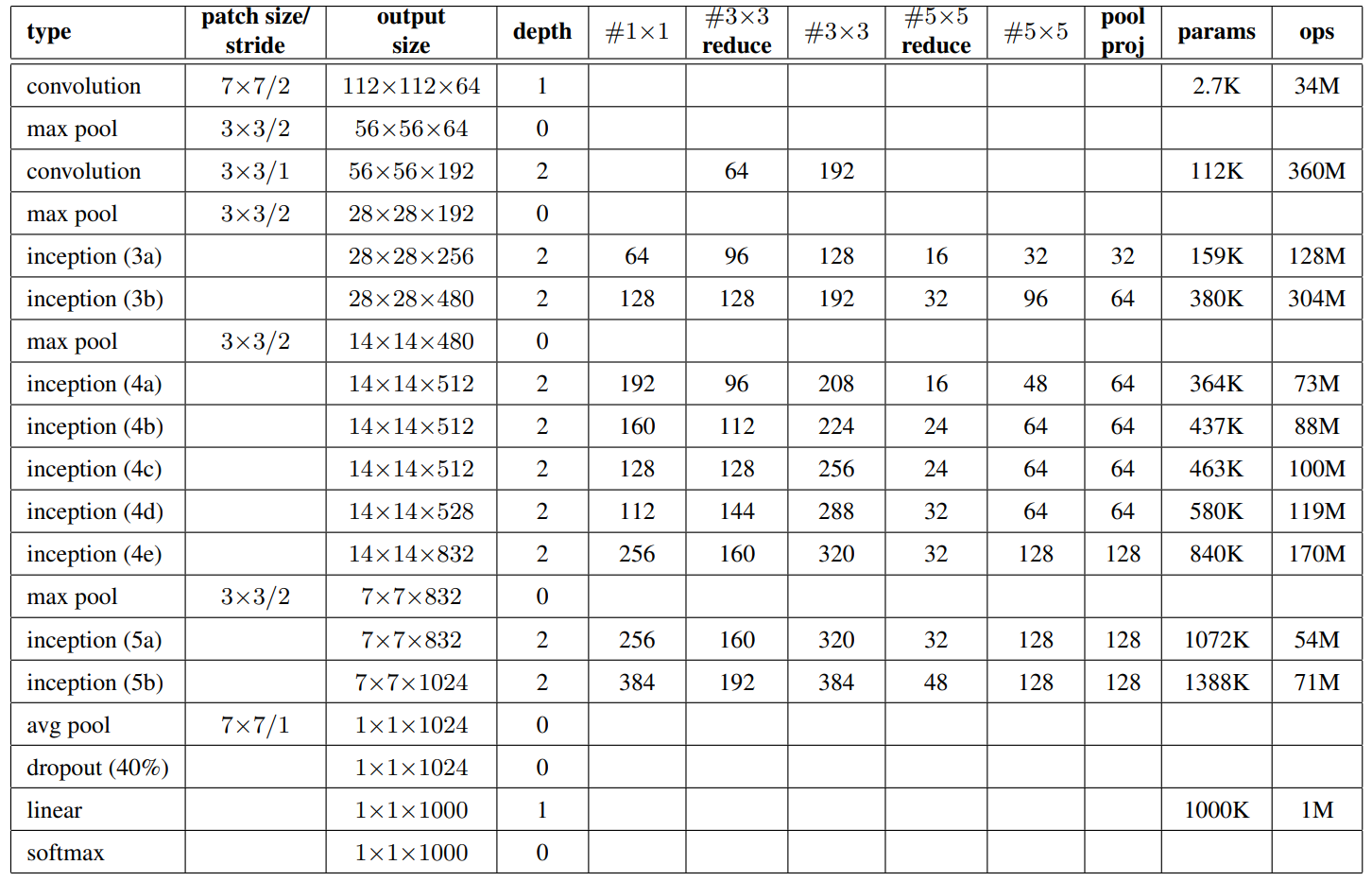


* **Auxiliary Classifier for Training:**

Inception architecture used some intermediate classifier branches in the middle of the architecture, these branches are used during training only. These branches consist of a 5×5 average pooling layer with a stride of 3, a 1×1 convolutions with 128 filters, two fully connected layers of 1024 outputs and 1000 outputs and a Softmax classification layer. The generated loss of these layers added to total loss with a weight of 0.3. These layers help in combating gradient vanishing problem and also provide regularization.

**Model Architecture:**

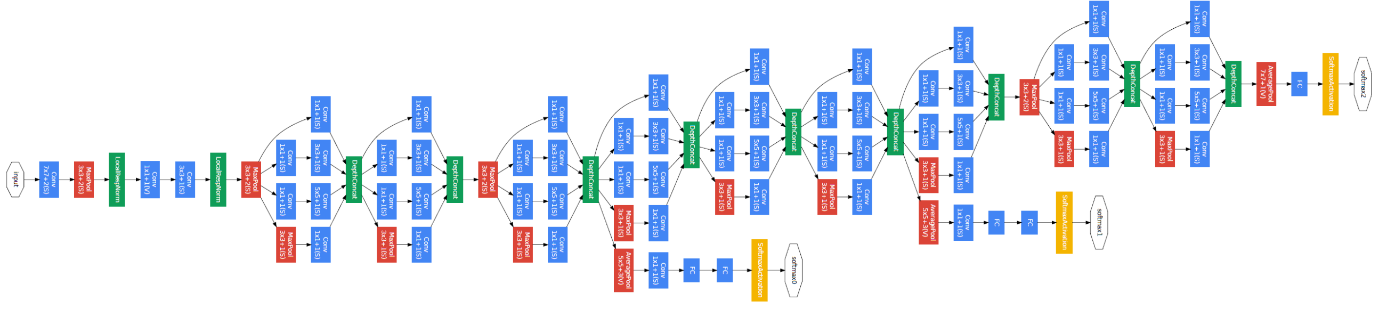
Below is Layer by Layer architectural details of GoogLeNet.



The overall architecture is 22 layers deep. The architecture was designed to keep computational efficiency in mind. The idea behind that the architecture can be run on individual devices even with low computational resources. The architecture also contains two auxiliary classifier layer connected to the output of Inception (4a) and Inception (4d) layers.

The architectural details of auxiliary classifiers as follows:

* An average pooling layer of filter size 5×5 and stride 3.
* A 1×1 convolution with 128 filters for dimension reduction and ReLU activation.
* A fully connected layer with 1025 outputs and ReLU activation
* Dropout Regularization with dropout ratio = 0.7
* A Softmax classifier with 1000 classes output similar to the main Softmax classifier.



This architecture takes image of size 224 x 224 with RGB colour channels. All the convolutions inside this architecture uses Rectified Linear Units (ReLU) as their activation functions.

**STEPS FOR EXECUTING THE PROJECTS**

1. Install the required packages

2. Defining the custom model.

3. Loading the dataset.

4. Pre-Processing the dataset.

5. Training the custom model.

6. Training the pre-trained model with our own dataset using GoogleNet.

7. Performing prediction.

8. Create a Flask based User Interface.

**SYSTEM DESIGN**

**UML Diagrams**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful insists the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**Goals:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

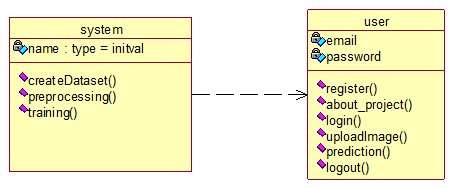
**Use Case Diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



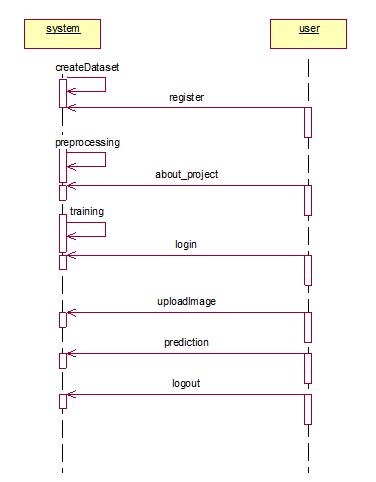
**Class Diagram:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



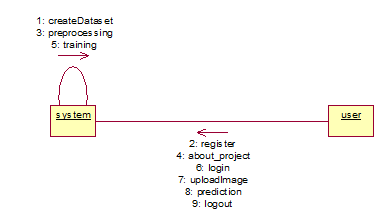
**Sequence Diagram:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and time diagrams.



**Collaboration Diagram:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



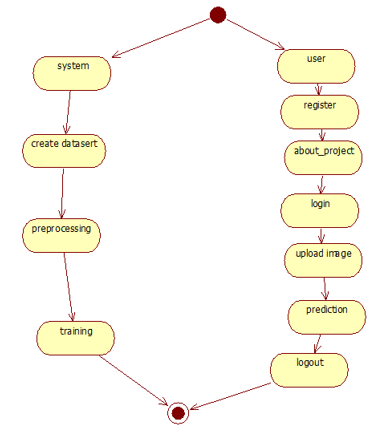
**Deployment diagram**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware’s used to deploy the application.



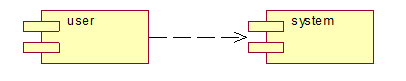
**Activity Diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**Component diagram**:

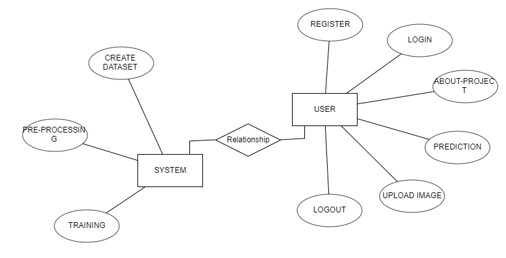
A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.



**ER Diagram:**

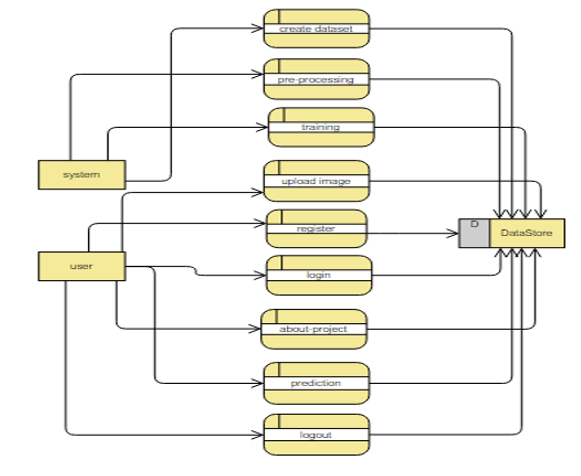
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

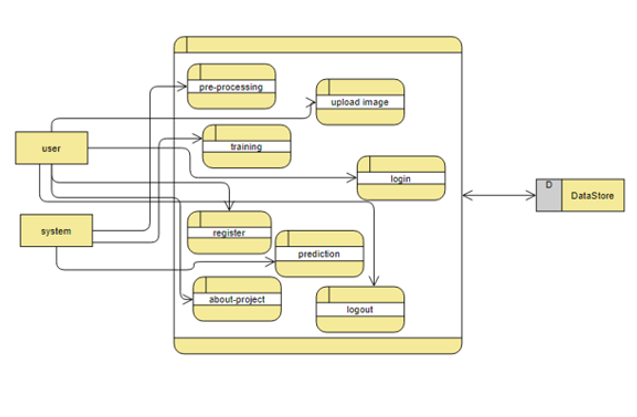
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.



**DFD Diagram:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.





**SYSTEM STUDY**

**Feasibility Study**

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* Economical Feasibility
* Technical Feasibility
* Social Feasibility

**Economical Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**Social Feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

### **SYSTEM TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**Types of Tests**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# **Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.