**Automatic Thyroid Ultrasound Image Classification Using Feature Fusion Network**

**ABSTRACT**

**ABSTRACT**

Nowadays, diagnosis of thyroid nodules is mainly based on clinical methods, which requires a lot of manpower and medical resources. Therefore, this work proposes an automated thyroid ultrasound nodule diagnosis method that combines convolutional neural networks and image texture features. The main steps include: Firstly, ultrasound thyroid nodule dataset is established by collecting positive and negative samples, standardizing of images and segmentation of nodule area. Secondly, through texture features extraction, feature selection and data dimensionality reduction, texture features model is obtained; Thirdly, by transfer learning, deep neural network is used to obtain feature model of the nodule in images; Then, texture features model and convolutional neural network feature model are combined to form a new nodule feature model called Feature Fusion Network; Finally, Feature Fusion Network is applied to train and improve performance than single network, and a deep neural network diagnosis model that can adapt to the characteristics of thyroid nodules is built. In order to test this method, 1874 groups of clinical ultrasound thyroid nodules are collected. Harmonic average F-score based on Precision and Recall is used as an evaluation indicator. Experimental results show that Feature Fusion Network can distinguish between benign and malignant thyroid nodules with an F-score of 92.52%. Compared with traditional machine learning methods and convolutional neural networks, performance of this work is better.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PGNO |
| 1 | Introduction   * 1. Software requirements   2. Hardware requirements |  |
| 2 | Feasibility study  2.1 Economic feasibility  2.2 Technical feasibility  2.3 Social feasibility |  |
| 3 | Literature survey |  |
| 4 | System analysis  4.1 Existing system  4.1.1 Disadvantages of existing system  4.2 Proposed system  4.2.1 Advantages of proposed system  4.3 Functional requirements  4.4 Non-Functional requirements |  |
| 5 | System design  5.1 System architecture  5.2 UML diagrams |  |
| 6 | Implementation  6.1 Modules  6.2 Sample code |  |
| 7 | Software environment |  |
| 8 | System testing  8.1 Testing strategies  8.2 Test cases |  |
| 9 | Screens |  |
| 10 | Conclusion |  |
| 11 | References |  |

LIST OF FIGURE

|  |  |  |
| --- | --- | --- |
| FIG.NO | FIG.NAME | PG.NO |
| 5.1.1 | System architecture |  |
| 5.1.2 | Dataflow diagram |  |
| 5.2.1 | Usecase diagram |  |
| 5.2.2 | Class diagram |  |
| 5.2.3 | Activity diagram |  |
| 5.2.4 | Sequence diagram |  |
| 5.2.5 | Collaboration diagram |  |
| 5.2.6 | Component diagram |  |
| 5.2.7 | Deployment diagram |  |

**INTRODUCTION**

**1.INTRODUCTION**

With the increase of people’s life pressure, prevalence of thyroid nodules has increased year by year in the world. It has become one of the most important diseases and is threatening human health [1]. Therefore, early diagnosis of thyroid nodules is very important [2]. The diagnostic methods of thyroid nodules mainly include ultrasound examination, CT examination, aspiration biopsy and pathological examination. CT examination requires nuclear scanning, which is harmful to patients and is expensive. Needle biopsy and pathological examination are more commonly used and reliable methods, but these two methods are very traumatic to thyroid tissue. Also, their diagnosis process is cumbersome, which will occupy more medical resources. Ultrasonography is currently the common imaging method for diagnosing thyroid diseases. It has the advantages of simplicity, good reproducibility, non-invasive, fast and low price. Usually, doctors can only judge benign and malignant based on clinical experience, which are highly subjective and easily affected. Therefore, the ability to accurately and quickly identify and diagnose the pathology of ultrasound thyroid nodules has become an increasingly urgent need.

In recent years, application of artificial intelligence technology in the medicine has gradually increased, especially in imaging [3]–[5] and signal [6]. How to use information of ultrasound images to establish a computer-assisted automated thyroid diagnosis system is an important direction of current research [7], [8]. The commonly applied method of assisting medical diagnosis is to use features extraction engineering and classifiers for classification. For example, Zheng et al. [9] used LR (Logistic Regression) to screen out indicators that had a greater impact on judging benign and malignant thyroid. This regression models can achieve pathological classification of images. Liu et al. extracted local texture features of thyroid nodules from region of interest, and applied KNN (K-NearestNeighbor) algorithm to obtain diagnosis results. Choi and Choi took thresholds and 3D connected region labeling methods to assist doctors in detection by classifiers based on genetic planning. These technologies are based on computer theoretical systems and establish accurate computer diagnosis methods. However, it depends on the completeness of feature textures information and selection of a suitable classifier.

**1.1 SOFTWARE REQUIREMENTS**

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

**Platform –** In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer’s architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

**APIs and drivers –** Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

**Web browser –** Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

**1)Visual Studio Community Version**

**2)Nodejs ( Version 12.3.1)**

**3)Python IDEL ( Python 3.7 )**

**1.2 HARDWARE REQUIREMENTS**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

**Architecture –** All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

**Processing power –** The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

**Memory –** All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

**Secondary storage –** Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

**Display adapter –** Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

**Peripherals –** Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

**1)Operating System : Windows Only**

**2)Processor : i5 and above**

**3)Ram : 4gb and above**

**4)Hard Disk : 50 GB**

**FEASIBILITY STUDY**

**2. FEASIBILITY STUDY**

The feasibility of the project is analyzed 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

**2.1 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.

### **2.2 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.

**2.3 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.

**LITERATURE SURVEY**

**3.LITERATURE SURVEY**

**3.1 Geographic influences in the global rise of thyroid cancer:**

<https://www.nature.com/articles/s41574-019-0263-x>

**ABSTRACT:** The incidence of thyroid cancer is on the rise, and this disease is projected to become the fourth leading type of cancer across the globe. From 1990 to 2013, the global age-standardized incidence rate of thyroid cancer increased by 20%. This global rise in incidence has been attributed to several factors, including increased detection of early tumours, the elevated prevalence of modifiable individual risk factors (for example, obesity) and increased exposure to environmental risk factors (for example, iodine levels). In this Review, we explore proven and novel hypotheses for how modifiable risk factors and environmental exposures might be driving the worldwide increase in the incidence of thyroid cancer. Although overscreening and the increased diagnosis of possibly clinically insignificant disease might have a role in certain parts of the world, other areas could be experiencing a true increase in incidence due to elevated exposure risks. In the current era of personalized medicine, national and international registry data should be applied to identify populations who are at increased risk for the development of thyroid cancer..

**3.2 Improving the accuracy of early diagnosis of thyroid nodule type based on the SCAD method**

<https://www.researchgate.net/publication/304931466_Improving_the_Accuracy_of_Early_Diagnosis_of_Thyroid_Nodule_Type_Based_on_the_SCAD_Method>

**ABSTRACT:** Although early diagnosis of thyroid nodule type is very important, the diagnostic accuracy of standard tests is a challenging issue. We here aimed to find an optimal combination of factors to improve diagnostic accuracy for distinguishing malignant from benign thyroid nodules before surgery. In a prospective study from 2008 to 2012, 345 patients referred for thyroidectomy were enrolled. The sample size was split into a training set and testing set as a ratio of 7:3. The former was used for estimation and variable selection and obtaining a linear combination of factors. We utilized smoothly clipped absolute deviation (SCAD) logistic regression to achieve the sparse optimal combination of factors. To evaluate the performance of the estimated model in the testing set, a receiver operating characteristic (ROC) curve was utilized. The mean age of the examined patients (66 male and 279 female) was 40.9 ± 13.4 years (range 15-90 years). Some 54.8% of the patients (24.3% male and 75.7% female) had benign and 45.2% (14% male and 86% female) malignant thyroid nodules. In addition to maximum diameters of nodules and lobes, their volumes were considered as related factors for malignancy prediction (a total of 16 factors). However, the SCAD method estimated the coefficients of 8 factors to be zero and eliminated them from the model. Hence a sparse model which combined the effects of 8 factors to distinguish malignant from benign thyroid nodules was generated. An optimal cut off point of the ROC curve for our estimated model was obtained (p=0.44) and the area under the curve (AUC) was equal to 77% (95% CI: 68%-85%). Sensitivity, specificity, positive predictive value and negative predictive values for this model were 70%, 72%, 71% and 76%, respectively. An increase of 10 percent and a greater accuracy rate in early diagnosis of thyroid nodule type by statistical methods (SCAD and ANN methods) compared with the results of FNA testing revealed that the statistical modeling methods are helpful in disease diagnosis. In addition, the factor ranking offered by these methods is valuable in the clinical context.

**3.3 A review of medical image detection for cancers in digestive system based on artificial intelligence**

<https://pubmed.ncbi.nlm.nih.gov/31530047/>

**ABSTRACT:** At present, cancer imaging examination relies mainly on manual reading of doctors, which requests a high standard of doctors' professional skills, clinical experience, and concentration. However, the increasing amount of medical imaging data has brought more and more challenges to radiologists. The detection of digestive system cancer (DSC) based on artificial intelligence (AI) can provide a solution for automatic analysis of medical images and assist doctors to achieve high-precision intelligent diagnosis of cancers. Areas covered: The main goal of this paper is to introduce the main research methods of the AI based detection of DSC, and provide relevant reference for researchers. Meantime, it summarizes the main problems existing in these methods, and provides better guidance for future research. Expert commentary: The automatic classification, recognition, and segmentation of DSC can be better realized through the methods of machine learning and deep learning, which minimize the internal information of images that are difficult for humans to discover. In the diagnosis of DSC, the use of AI to assist imaging surgeons can achieve cancer detection rapidly and effectively and save doctors' diagnosis time. These can lay the foundation for better clinical diagnosis, treatment planning and accurate quantitative evaluation of DSC.

**3.4 An intelligent platform for ultrasound diagnosis of thyroid nodules**

<https://www.researchgate.net/publication/343484594_An_intelligent_platform_for_ultrasound_diagnosis_of_thyroid_nodules>

**ABSTRACT:** This paper proposed a non-segmentation radiological method for classification of benign and malignant thyroid tumors using B mode ultrasound data. This method aimed to combine the advantages of morphological information provided by ultrasound and convolutional neural networks in automatic feature extraction and accurate classification. Compared with the traditional feature extraction method, this method directly extracted features from the data set without the need for segmentation and manual operations. 861 benign nodule images and 740 malignant nodule images were collected for training data. A deep convolution neural network VGG-16 was constructed to analyze test data including 100 malignant nodule images and 109 benign nodule images. A nine fold cross validation was performed for training and testing of the classifier. The results showed that the method had an accuracy of 86.12%, a sensitivity of 87%, and a specificity of 85.32%. This computer-aided method demonstrated comparable diagnostic performance with the result reported by an experienced radiologist based on American college of radiology thyroid imaging reporting and data system (ACR TI-RADS) (accuracy: 87.56%, sensitivity: 92%, and specificity: 83.49%). The automation advantage of this method suggested application potential in computer-aided diagnosis of thyroid cancer.

**3.5 Automatic thyroid nodule recognition and diagnosis in ultrasound imaging with the YOLOv2 neural network**

<https://www.researchgate.net/publication/330232796_Automatic_thyroid_nodule_recognition_and_diagnosis_in_ultrasound_imaging_with_the_YOLOv2_neural_network>

**ABSTRACT:** Background In this study, images of 2450 benign thyroid nodules and 2557 malignant thyroid nodules were collected and labeled, and an automatic image recognition and diagnosis system was established by deep learning using the YOLOv2 neural network. The performance of the system in the diagnosis of thyroid nodules was evaluated, and the application value of artificial intelligence in clinical practice was investigated. Methods The ultrasound images of 276 patients were retrospectively selected. The diagnoses of the radiologists were determined according to the Thyroid Imaging Reporting and Data System; the images were automatically recognized and diagnosed by the established artificial intelligence system. Pathological diagnosis was the gold standard for the final diagnosis. The performances of the established system and the radiologists in diagnosing the benign and malignant thyroid nodules were compared. Results The artificial intelligence diagnosis system correctly identified the lesion area, with an area under the receiver operating characteristic (ROC) curve of 0.902, which is higher than that of the radiologists (0.859). This finding indicates a higher diagnostic accuracy (p = 0.0434). The sensitivity, positive predictive value, negative predictive value, and accuracy of the artificial intelligence diagnosis system for the diagnosis of malignant thyroid nodules were 90.5%, 95.22%, 80.99%, and 90.31%, respectively, and the performance did not significantly differ from that of the radiologists (p > 0.05). The artificial intelligence diagnosis system had a higher specificity (89.91% vs 77.98%, p = 0.026). Conclusions Compared with the performance of experienced radiologists, the artificial intelligence system has comparable sensitivity and accuracy for the diagnosis of malignant thyroid nodules and better diagnostic ability for benign thyroid nodules. As an auxiliary tool, this artificial intelligence diagnosis system can provide radiologists with sufficient assistance in the diagnosis of benign and malignant thyroid nodules.

**SYSTEM ANALYSIS**

**4.SYSTEM ANALYSIS**

**4.1 EXISTING SYSTEM:**

With the development of deep learning, some researchers are studying convolutional neural networks to diagnose thyroid ultrasound nodules. For example, Moran et al. established S-Detect technology based on the GoogLeNet. They cooperated with clinical sonographers for joint diagnosis to improve diagnostic performance. Xie et al. decomposed nodules into 9 views to learn 3D features. They built a multi-view knowledge-based collaborative model for each view and input three images into ResNet-50 network for training to represent appearance, voxel, and shape specificity. In summary, convolutional neural network usually does not require too much pre-processing operations, and has advantages of convenience and simplicity. However, it is very dependent on feature completeness of training data due to lack of sufficient prior theoretical support. In this case, direction and details of feature training are usually unknown. How to further improve diagnosis accuracy is still urgently needed.

**4.1.1 DISADVANTAGES OF EXISTING SYSTEM:**

1. However, it is very dependent on feature completeness of training data due to lack of sufficient prior theoretical support.
2. In this case, direction and details of feature training are usually unknown.
3. How to further improve diagnosis accuracy is still urgently needed

# 4.2 Proposed System:

This research suggests an automated thyroid ultrasonography nodule diagnosis technique that incorporates picture texture data and convolutional neural networks. The key actions consist of: By gathering both positive and negative samples, normalising the pictures, and segmenting the nodule area, the ultrasonography thyroid nodule dataset is first created. Second, a texture features model is created through the extraction of texture features, feature selection, and data dimensionality reduction; Thirdly, a deep neural network is employed to obtain a feature representation of the nodule in photos by transfer learning; Then, a new nodule feature model called Feature Fusion Network is created by fusing the texture and convolutional neural network feature models; A deep neural network diagnosis model that can adapt and feature fusion network are used to train and increase performance over single networks..

# 4.2.1 Advantages of proposed system:

1. Compared with traditional machine learning methods and convolutional neural networks, performance of this work is better.

### **4.3 FUNCTIONAL REQUIREMENTS**

1.Data Collection

2.Data Preprocessing

3.Training And Testing

4.Modiling

5.Predicting

### **4.4 NON FUNCTIONAL REQUIREMENTS**

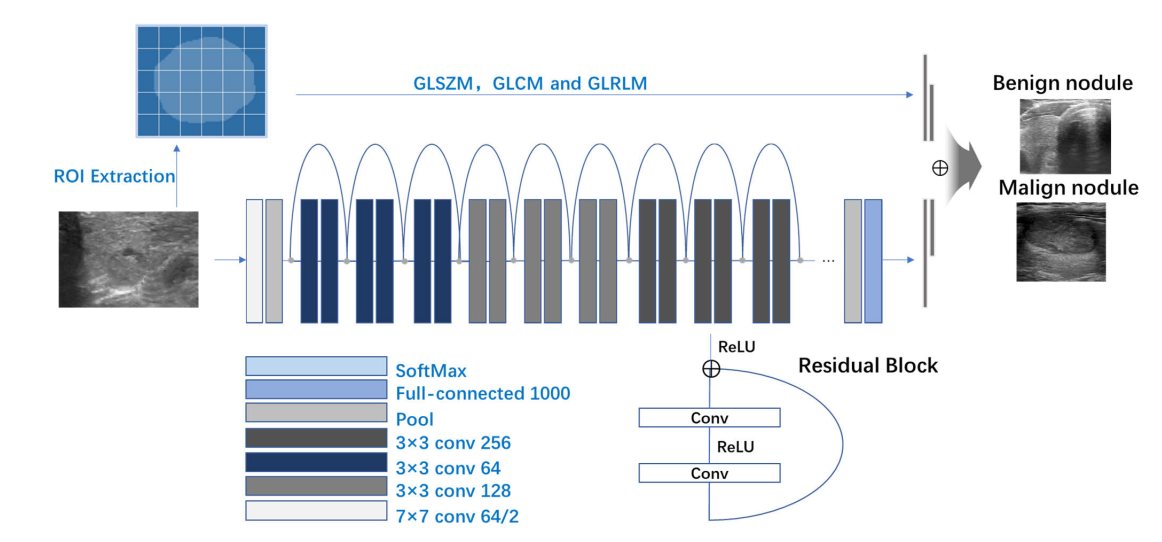
NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**SYSTEM DESIGN**

**5. SYSTEM DESIGN**

**5.1 SYSTEM ARCHITECTURE:**

****

**Fig.5.1.1 System architecture**

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

**Import libraries**

**VERIFY**

**NO PROCESS**

**Yes NO**

**Exploring the dataset – Thyroid ultrasound data**

**Image processing**

**Data visualization**

**Splitting the data into train & test**

**Building the model - Feature Fusion ResNet, Feature Fusion VGG16, VGG16 with Feed Forward Network Transfer Learning, ResNet50, VGG16, MobileNet V2 and GAN**

**KNN, LR and Voting Classifier**

**Training the model**

**Signup & signin**

**User input**

**End process**

**Final outcome**

**Fig.5.1.3 Dataflow diagram**

**5.2 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 in 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.



**Fig.5.2.1 Usecase diagram**

**Class diagram:**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.



**Fig.5.2.2 Class diagram**

**Activity diagram:**

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

Importing libraries

Exploring the dataset

Image preprocessing

Data visualization

Splitting the data to train & test

Building the model - Feature Fusion ResNet, Feature Fusion VGG16, VGG16 with Feed Forward Network Transfer Learning, ResNet50, VGG16, MobileNet V2 and GAN

KNN, LR and Voting Classifier

Training the model

User signup & signin

User input

Final outcome

**Fig.5.2.3 Activity diagram**

**Sequence diagram:**

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".



**Fig.5.2.4 Sequence diagram**

**Collaboration diagram:**

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

****

**Fig.5.2.5 Collaboration diagram**

**Component diagram:**

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.



**Fig.5.2.6 Component diagram**

**Deployment diagram:**

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

****

**Fig.5.2.7 Deployment diagram**

**IMPLEMENTATION**

1. **IMPLEMENTATION**

MODULES:

* Data exploration: using this module we will load data into system
* Processing: Using the module we will read data for processing
* Splitting data into train & test: using this module data will be divided into train & test

Model generation: Building the model - Feature Fusion ResNet, Feature Fusion VGG16, VGG16 with Feed Forward Network Transfer Learning, ResNet50, VGG16, MobileNet V2 and GAN

KNN, LR and Voting Classifiers. Algorithms accuracy calculted

* User signup & login: Using this module will get registration and login
* User input: Using this module will give input for prediction
* Prediction: final predicted displayed

**ALGORITHMS:**

Feature Fusion ResNet: Feature fusion refers to the fusion of feature vectors of training images extracted from shared weight network layer and feature vectors composed of other numerical data, so that the proposed model can utilize features as many as possible for the further classification. ResNet is an artificial neural network that introduced a so-called “identity shortcut connection,” which allows the model to skip one or more layers. This approach makes it possible to train the network on thousands of layers without affecting performance.

Feature Fusion VGG16: VGG-16 is a convolutional neural network that is 16 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

VGG16 with Feed Forward Network Transfer Learning: VGG16 is a 16 layer transfer learning architecture and is quite similar to earlier architectures as it's foundation is based on CNN only but the arrangement is a bit different. The standard input image size which was taken by the researchers for this architecture was 224\*224\*3 where 3 represents the RGB channel.

ResNet50: ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

VGG16: VGG-16 is a convolutional neural network that is 16 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

MobileNet V2: MobileNet-v2 is a convolutional neural network that is 53 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

GAN: A generative adversarial network (GAN) is a machine learning (ML) model in which two neural networks compete with each other to become more accurate in their predictions. GANs typically run unsupervised and use a cooperative zero-sum game framework to learn.

KNN: The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

LR: Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of certain classes based on some dependent variables. In short, the logistic regression model computes a sum of the input features (in most cases, there is a bias term), and calculates the logistic of the result.

Voting Classifiers: A voting classifier is a machine learning estimator that trains various base models or estimators and predicts on the basis of aggregating the findings of each base estimator. The aggregating criteria can be combined decision of voting for each estimator output.

**6.2 SAMPLE CODE:**

|  |
| --- |
| import numpy as np |
|  | import pandas as pd |
|  | from glob import glob |
|  | from imgaug import augmenters |
|  | from random import seed, randint |
|  | from scipy.misc import imread |
|  |  |
|  | data\_path = "./data.csv" |
|  | images\_dir = "/data/images-cv" |
|  | test\_images\_dir = "/data/images-test" |
|  |  |
|  | random\_seed = 3 |
|  | total\_folds = 10 |
|  |  |
|  |  |
|  | def feature\_classes(feature): |
|  | df = pd.read\_csv(data\_path) |
|  | df.fillna(0, inplace=True) |
|  | df.Calcs1.replace(0, "None", inplace=True) |
|  | if feature == "composition": |
|  | return list(pd.get\_dummies(df.Composition, prefix="", prefix\_sep="").columns) |
|  | if feature == "echogenicity": |
|  | return list(pd.get\_dummies(df.Echogenicity, prefix="", prefix\_sep="").columns) |
|  | if feature == "shape": |
|  | return ["wider", "taller"] |
|  | if feature == "calcification": |
|  | return list(pd.get\_dummies(df.Calcs1, prefix="", prefix\_sep="").columns) |
|  | if feature == "margin": |
|  | return list(pd.get\_dummies(df.MargA, prefix="", prefix\_sep="").columns) |
|  | return [] |
|  |  |
|  |  |
|  | def fold\_pids(fold, test=True): |
|  | # get patient IDs for given training fold in 10-fold cross-validation |
|  | # if test=False, test patient IDs are returned |
|  | df = pd.read\_csv(data\_path) |
|  | all\_files = glob(os.path.join(images\_dir, "\*.PNG")) |
|  | val\_ids = validation\_ids(fold, df[["ID", "Cancer"]]) |
|  | pids = [] |
|  | for f\_path in all\_files: |
|  | pid = fname2pid(f\_path) |
|  | if (test and pid in val\_ids) or (not test and pid not in val\_ids): |
|  | pids.append(pid) |
|  | return pids |
|  |  |
|  |  |
|  | def test\_pids(): |
|  | # get patient IDs for test cases |
|  | test\_files = sorted(glob(os.path.join(test\_images\_dir, "\*.PNG"))) |
|  | pids = [] |
|  | for f\_path in test\_files: |
|  | pids.append(fname2pid(f\_path)) |
|  | return pids |
|  |  |
|  |  |
|  | def train\_pids(): |
|  | # get patient IDs for training cases |
|  | train\_files = sorted(glob(images\_dir + "/\*.PNG")) |
|  | pids = [] |
|  | for f\_path in train\_files: |
|  | pids.append(fname2pid(f\_path)) |
|  | return pids |
|  |  |
|  |  |
|  | def train\_data(): |
|  | # get images and labels for training cases |
|  | df = pd.read\_csv(data\_path) |
|  | df.fillna(0, inplace=True) |
|  | df.Calcs1.replace(0, "None", inplace=True) |
|  |  |
|  | df\_cancer = df[["ID", "Cancer"]] |
|  | df\_compos = pd.concat([df.ID, pd.get\_dummies(df.Composition)], axis=1) |
|  | df\_echo = pd.concat([df.ID, pd.get\_dummies(df.Echogenicity)], axis=1) |
|  | df\_shape = df[["ID", "Shape"]] |
|  | df\_shape["Shape"] = df\_shape.apply(lambda row: 1 if row.Shape == "y" else 0, axis=1) |
|  | df\_calcs = pd.concat([df.ID, pd.get\_dummies(df.Calcs1)], axis=1) |
|  | df\_margin = pd.concat([df.ID, pd.get\_dummies(df.MargA)], axis=1) |
|  |  |
|  | train\_files = sorted(glob(os.path.join(images\_dir, "\*.PNG"))) |
|  | X\_train = [] |
|  |  |
|  | # labels for malignancy and 5 TI-RADS features |
|  | y\_train\_cancer = [] |
|  | y\_train\_compos = [] |
|  | y\_train\_echo = [] |
|  | y\_train\_shape = [] |
|  | y\_train\_calcs = [] |
|  | y\_train\_margin = [] |
|  |  |
|  | for f\_path in train\_files: |
|  | pid = fname2pid(f\_path) |
|  | X\_train.append( |
|  | np.expand\_dims( |
|  | np.array(imread(f\_path, flatten=False, mode="F")).astype(np.float32), |
|  | axis=-1, |
|  | ) |
|  | ) |
|  | y\_train\_cancer.append( |
|  | df\_cancer[df\_cancer.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  | y\_train\_compos.append( |
|  | df\_compos[df\_compos.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  | y\_train\_echo.append( |
|  | df\_echo[df\_echo.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  | # for shape feature, only assign positive label to transversal view |
|  | if "trans" in f\_path: |
|  | y\_train\_shape.append( |
|  | df\_shape[df\_shape.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | else: |
|  | y\_train\_shape.append(np.array([0]).astype(np.float32)) |
|  | y\_train\_calcs.append( |
|  | df\_calcs[df\_calcs.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  | y\_train\_margin.append( |
|  | df\_margin[df\_margin.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  |  |
|  | X\_train = np.array(X\_train) |
|  |  |
|  | # normalize |
|  | X\_train /= 255. |
|  | X\_train -= 0.5 |
|  | X\_train \*= 2. |
|  |  |
|  | y\_train = { |
|  | "out\_cancer": np.array(y\_train\_cancer), |
|  | "out\_compos": np.array(y\_train\_compos), |
|  | "out\_echo": np.array(y\_train\_echo), |
|  | "out\_shape": np.array(y\_train\_shape), |
|  | "out\_calcs": np.array(y\_train\_calcs), |
|  | "out\_margin": np.array(y\_train\_margin), |
|  | } |
|  |  |
|  | return X\_train, y\_train |
|  |  |
|  |  |
|  | def test\_data(): |
|  | # get images and labels for test cases |
|  | df = pd.read\_csv(data\_path) |
|  | df.fillna(0, inplace=True) |
|  | df.Calcs1.replace(0, "None", inplace=True) |
|  |  |
|  | df\_cancer = df[["ID", "Cancer"]] |
|  | df\_compos = pd.concat([df.ID, pd.get\_dummies(df.Composition)], axis=1) |
|  | df\_echo = pd.concat([df.ID, pd.get\_dummies(df.Echogenicity)], axis=1) |
|  | df\_shape = df[["ID", "Shape"]] |
|  | df\_shape["Shape"] = df\_shape.apply(lambda row: 1 if row.Shape == "y" else 0, axis=1) |
|  | df\_calcs = pd.concat([df.ID, pd.get\_dummies(df.Calcs1)], axis=1) |
|  | df\_margin = pd.concat([df.ID, pd.get\_dummies(df.MargA)], axis=1) |
|  |  |
|  | test\_files = sorted(glob(test\_images\_dir + "/\*.PNG")) |
|  |  |
|  | X\_test = [] |
|  |  |
|  | # labels for malignancy and 5 TI-RADS features |
|  | y\_test\_cancer = [] |
|  | y\_test\_compos = [] |
|  | y\_test\_echo = [] |
|  | y\_test\_shape = [] |
|  | y\_test\_calcs = [] |
|  | y\_test\_margin = [] |
|  |  |
|  | for f\_path in test\_files: |
|  | pid = fname2pid(f\_path) |
|  | X\_test.append( |
|  | np.expand\_dims( |
|  | np.array(imread(f\_path, flatten=False, mode="F")).astype(np.float32), |
|  | axis=-1, |
|  | ) |
|  | ) |
|  | y\_test\_cancer.append( |
|  | df\_cancer[df\_cancer.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  | y\_test\_compos.append( |
|  | df\_compos[df\_compos.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  | y\_test\_echo.append( |
|  | df\_echo[df\_echo.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  | if "trans" in f\_path: |
|  | y\_test\_shape.append( |
|  | df\_shape[df\_shape.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | else: |
|  | y\_test\_shape.append(np.array([0]).astype(np.float32)) |
|  | y\_test\_calcs.append( |
|  | df\_calcs[df\_calcs.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  | y\_test\_margin.append( |
|  | df\_margin[df\_margin.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  |  |
|  | X\_test = np.array(X\_test) |
|  |  |
|  | # normalize |
|  | X\_test /= 255. |
|  | X\_test -= 0.5 |
|  | X\_test \*= 2. |
|  |  |
|  | y\_test = [ |
|  | np.array(y\_test\_cancer), |
|  | np.array(y\_test\_compos), |
|  | np.array(y\_test\_echo), |
|  | np.array(y\_test\_shape), |
|  | np.array(y\_test\_calcs), |
|  | np.array(y\_test\_margin), |
|  | ] |
|  |  |
|  | return X\_test, y\_test |
|  |  |
|  |  |
|  | def fold\_data(fold): |
|  | # get images and labels for given fold |
|  | df = pd.read\_csv(data\_path) |
|  | df.fillna(0, inplace=True) |
|  | df.Calcs1.replace(0, "None", inplace=True) |
|  |  |
|  | df\_cancer = df[["ID", "Cancer"]] |
|  | df\_compos = pd.concat([df.ID, pd.get\_dummies(df.Composition)], axis=1) |
|  | df\_echo = pd.concat([df.ID, pd.get\_dummies(df.Echogenicity)], axis=1) |
|  | df\_shape = df[["ID", "Shape"]] |
|  | df\_shape["Shape"] = df\_shape.apply(lambda row: 1 if row.Shape == "y" else 0, axis=1) |
|  | df\_calcs = pd.concat([df.ID, pd.get\_dummies(df.Calcs1)], axis=1) |
|  | df\_margin = pd.concat([df.ID, pd.get\_dummies(df.MargA)], axis=1) |
|  |  |
|  | all\_files = glob(images\_dir + "/\*.PNG") |
|  | val\_ids = validation\_ids(fold, df\_cancer) |
|  |  |
|  | X\_train = [] |
|  | X\_test = [] |
|  |  |
|  | # labels for malignancy and 5 TI-RADS features |
|  | y\_train\_cancer = [] |
|  | y\_train\_compos = [] |
|  | y\_train\_echo = [] |
|  | y\_train\_shape = [] |
|  | y\_train\_calcs = [] |
|  | y\_train\_margin = [] |
|  | y\_test\_cancer = [] |
|  | y\_test\_compos = [] |
|  | y\_test\_echo = [] |
|  | y\_test\_shape = [] |
|  | y\_test\_calcs = [] |
|  | y\_test\_margin = [] |
|  |  |
|  | for f\_path in all\_files: |
|  | pid = fname2pid(f\_path) |
|  | image = np.expand\_dims( |
|  | np.array(imread(f\_path, flatten=False, mode="F")).astype(np.float32), |
|  | axis=-1, |
|  | ) |
|  | if pid in val\_ids: |
|  | X\_test.append(image) |
|  | y\_test\_cancer.append( |
|  | df\_cancer[df\_cancer.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | y\_test\_compos.append( |
|  | df\_compos[df\_compos.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | y\_test\_echo.append( |
|  | df\_echo[df\_echo.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  | if "trans" in f\_path: |
|  | y\_test\_shape.append( |
|  | df\_shape[df\_shape.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | else: |
|  | y\_test\_shape.append(np.array([0]).astype(np.float32)) |
|  | y\_test\_calcs.append( |
|  | df\_calcs[df\_calcs.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | y\_test\_margin.append( |
|  | df\_margin[df\_margin.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | else: |
|  | X\_train.append(image) |
|  | y\_train\_cancer.append( |
|  | df\_cancer[df\_cancer.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | y\_train\_compos.append( |
|  | df\_compos[df\_compos.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | y\_train\_echo.append( |
|  | df\_echo[df\_echo.ID == pid].as\_matrix().flatten()[1:].astype(np.float32) |
|  | ) |
|  | if "trans" in f\_path: |
|  | y\_train\_shape.append( |
|  | df\_shape[df\_shape.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | else: |
|  | y\_train\_shape.append(np.array([0]).astype(np.float32)) |
|  | y\_train\_calcs.append( |
|  | df\_calcs[df\_calcs.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  | y\_train\_margin.append( |
|  | df\_margin[df\_margin.ID == pid] |
|  | .as\_matrix() |
|  | .flatten()[1:] |
|  | .astype(np.float32) |
|  | ) |
|  |  |
|  | X\_train = np.array(X\_train) |
|  | X\_test = np.array(X\_test) |
|  |  |
|  | # normalize |
|  | X\_train /= 255. |
|  | X\_train -= 0.5 |
|  | X\_train \*= 2. |
|  | X\_test /= 255. |
|  | X\_test -= 0.5 |
|  | X\_test \*= 2. |
|  |  |
|  | y\_train = { |
|  | "out\_cancer": np.array(y\_train\_cancer), |
|  | "out\_compos": np.array(y\_train\_compos), |
|  | "out\_echo": np.array(y\_train\_echo), |
|  | "out\_shape": np.array(y\_train\_shape), |
|  | "out\_calcs": np.array(y\_train\_calcs), |
|  | "out\_margin": np.array(y\_train\_margin), |
|  | } |
|  |  |
|  | y\_test = [ |
|  | np.array(y\_test\_cancer), |
|  | np.array(y\_test\_compos), |
|  | np.array(y\_test\_echo), |
|  | np.array(y\_test\_shape), |
|  | np.array(y\_test\_calcs), |
|  | np.array(y\_test\_margin), |
|  | ] |
|  |  |
|  | return X\_train, y\_train, X\_test, y\_test |
|  |  |
|  |  |
|  | def augment(X): |
|  | # data augmentation |
|  | seq = augmenters.Sequential( |
|  | [ |
|  | augmenters.Fliplr(0.5), |
|  | augmenters.Flipud(0.5), |
|  | augmenters.Affine(rotate=(-15, 15)), |
|  | augmenters.Affine(shear=(-15, 15)), |
|  | augmenters.Affine(translate\_percent={"x": (-0.2, 0.2), "y": (-0.2, 0.2)}), |
|  | augmenters.Affine(scale=(0.9, 1.1)), |
|  | ] |
|  | ) |
|  | return seq.augment\_images(X) |
|  |  |
|  |  |
|  | def validation\_ids(fold, df\_cancer): |
|  | # get patient IDs in given fold |
|  | pid\_set = set() |
|  | all\_files = glob(images\_dir + "/\*.PNG") |
|  | for f\_path in all\_files: |
|  | pid = fname2pid(f\_path) |
|  | pid\_set.add(pid) |
|  |  |
|  | val\_ids = [] |
|  |  |
|  | # set random seed to get the same split every time |
|  | seed(random\_seed) |
|  | # stratified split |
|  | malignant\_fold = 0 |
|  | for pid in sorted(pid\_set): |
|  | label = df\_cancer[df\_cancer.ID == pid].as\_matrix().flatten()[1:] |
|  | if label == 1: |
|  | if fold == np.mod(malignant\_fold, total\_folds): |
|  | val\_ids.append(pid) |
|  | malignant\_fold += 1 |
|  | else: |
|  | if fold == randint(0, total\_folds - 1): |
|  | val\_ids.append(pid) |
|  |  |
|  | return val\_ids |
|  |  |
|  |  |
|  | def fname2pid(fname): |
|  | # get patient ID from image file name |
|  | return fname.split("/")[-1].split(".")[0].lstrip("0") |

**SOFTWARE ENVIRONMENT**

**7.SOFTWARE ENVIRONMENT**

PYTHON LANGUAGE:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language. For example, x = 10 Here, x can be anything such as String, int, etc.

## Features in Python:

There are many features in Python, some of which are discussed below as follows:

### **1. Free and Open Source**

[Python](https://www.geeksforgeeks.org/python-programming-language/)language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword. [Download Python](https://www.python.org/downloads/) Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

### **2. Easy to code**

Python is a [high-level programming language](https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/). Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

### 3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

### **4. Object-Oriented Language**

One of the key features of [Python is Object-Oriented programming](https://www.geeksforgeeks.org/python-oops-concepts/). Python supports object-oriented language and concepts of classes, object encapsulation, etc.

### **5. GUI Programming Support**

Graphical User interfaces can be made using a module such as [PyQt5](https://www.geeksforgeeks.org/pyqt5-qaction/), PyQt4, wxPython, or [Tk in python](https://www.geeksforgeeks.org/python-gui-tkinter/). PyQt5 is the most popular option for creating graphical apps with Python.

### **6. High-Level Language**

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

### **7. Extensible feature**

Python is an **Extensible** language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

### 8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to [interpret](https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/)Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

### **9. Python is a Portable language**

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as [Linux](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/), Unix, and Mac then we do not need to change it, we can run this code on any platform.

### **10. Python is an Integrated language**

Python is also an Integrated language because we can easily integrate Python with other languages like C, [C++](http://www.geeksforgeeks.org/c-plus-plus/), etc.

### **11. Interpreted Language:**

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, [Java](https://www.geeksforgeeks.org/java/), etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called **bytecode**.

### **12. Large Standard Library**

Python has a large [standard library](https://www.geeksforgeeks.org/libraries-in-python/) that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as [regular expression](https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/)s, [unit-testing](https://www.geeksforgeeks.org/unit-testing-software-testing/), web browsers, etc.

### **13. Dynamically Typed Language**

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

### **14. Frontend and backend development**

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like javascript. Backend is the strong forte of Python it’s extensively used for this work cause of its frameworks like [Django](https://www.geeksforgeeks.org/django-tutorial/)and [Flask](https://www.geeksforgeeks.org/flask-creating-first-simple-application/).

### 15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

**LIBRARIES/PACKGES :-**

**Tensorflow**

TensorFlow is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library for dataflow and differentiable programming](https://en.wikipedia.org/wiki/Library_(computing)) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache 2.0](https://en.wikipedia.org/wiki/Apache_License) [open-source license](https://en.wikipedia.org/wiki/Open-source_license) on November 9, 2015.

**Numpy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the [sample plots](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

**SYSTEM TESTING**

**8.SYSTEM TESTING**

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

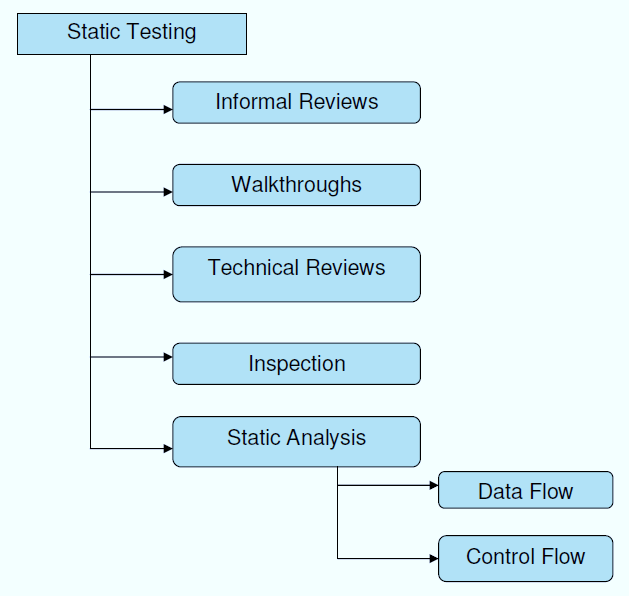
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

**8.1Software Testing Strategies:**

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

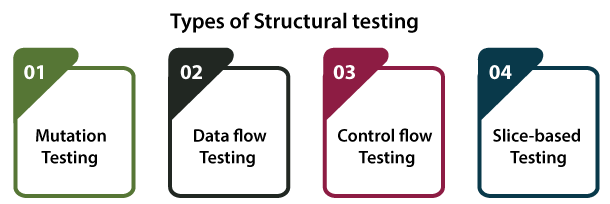
Static Testing:

The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.



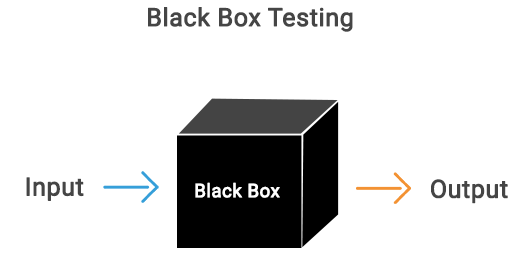
**Structural Testing:**

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software’s structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system’s behavior by comparing the tests’ outcomes with the results of previous iterations (control flow testing).



**Behavioral Testing:**

The final stage of testing focuses on the software’s reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user’s point of view. QA engineers usually have some specific information about a business or other purposes of the software (‘the black box’) to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.

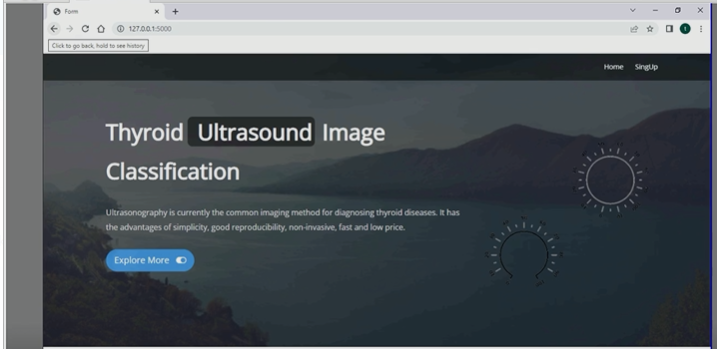


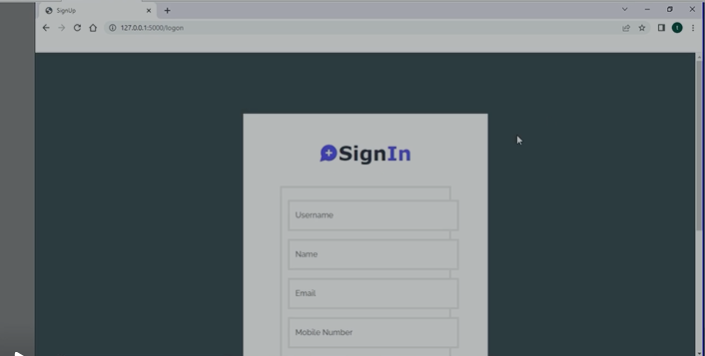
**8.2 TEST CASES:**

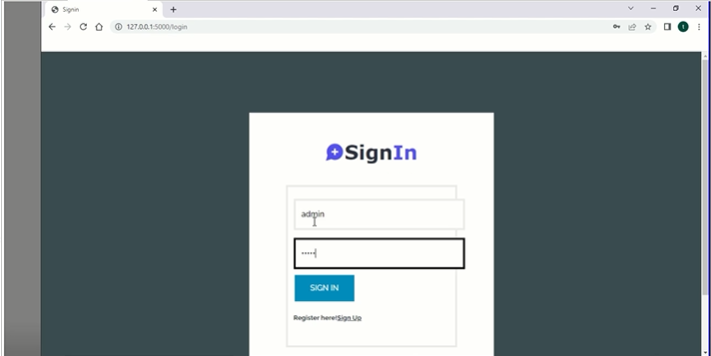
|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **INPUT** | **If available** | **If not available** |
| 1 | User signup | User get registered into the application | There is no process |
| 2 | User signin | User get login into the application | There is no process |
| 3 | Enter input for prediction | Prediction result displayed | There is no process |

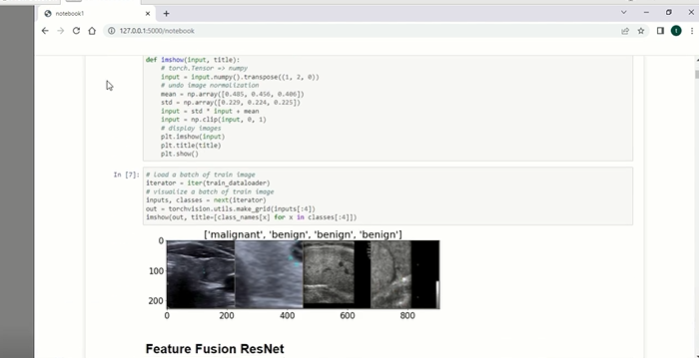
**SCREENS**

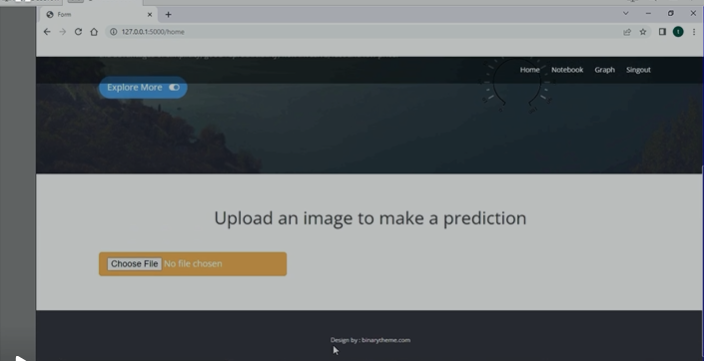
1. **SCREENSHOTS**

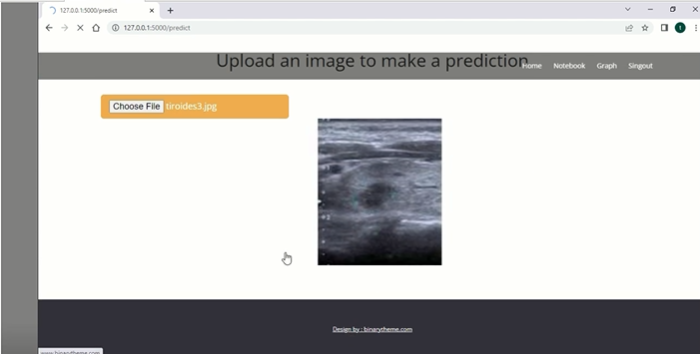
****

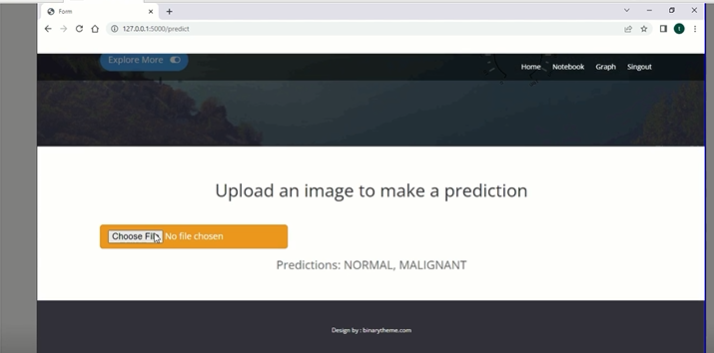
****

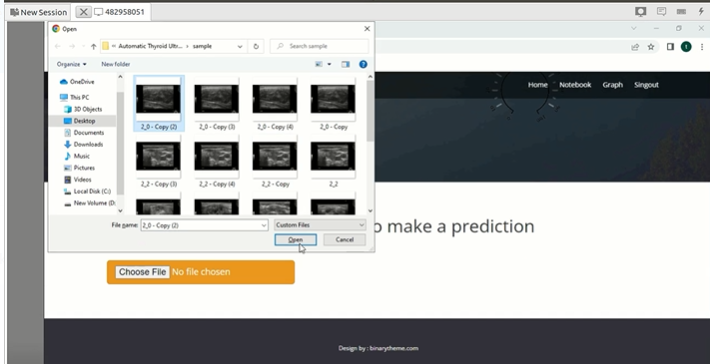
****

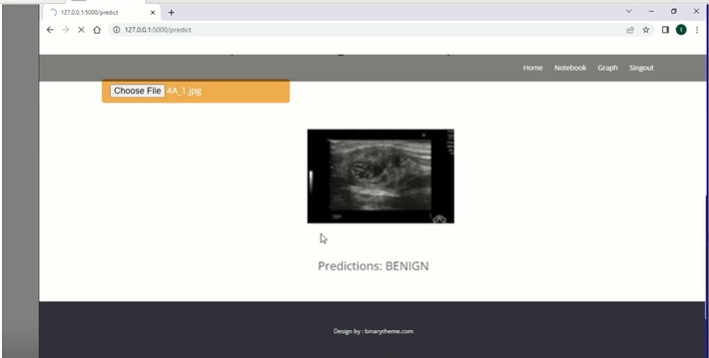
****

****

****

****

****

****

**CONCLUSION**

**10.CONCLUSION**

Since clinical diagnosis of benign and malignant thyroid nodules by ultrasound is a subjective and tedious process, this work aims to assist doctors in making clinical diagnosis of thyroid nodules, thereby improving the accuracy and efficiency of diagnosis. Firstly, it is necessary to preprocess the clinically collected data, including cropping, enhancement, and extraction of regions of interest. Then, feature engineering is applied to obtain texture features of nodules based on the nodules area, and feature dimensionality reduction is realized through the correlation between features and nodules. Finally, a deep neural network model is established, and texture features from the previous step are merged to achieve the goal of further improving network performance. Under assessment of 1874 cases with thyroid nodules, this method obtained the best performance, which has clinical potential. This work combines the advantages of feature engineering and deep neural networks, and proposes a novel way of fusing features. Although this work mainly validates the diagnostic performance of ultrasound imaging of thyroid nodules, this work can also be applied to various domains under the transfer learning and fusion feature structure, such as breast nodules, lung nodules and other tumor diagnosis. It is worth mentioning that the method of fusing features is mainly to introduce more features and information for deep neural network, so that the network can converge more accurately and quickly. This is also a future direction for new fusion information. This work has certain inspirations for computeraided diagnosis, application of deep convolutional networks, and image analysis.

**BIBILOGRAPHY**

**11. REFERENCES**

[1] J. Kim, J. E. Gosnell, and S. A. Roman, ‘‘Geographic influences in the global rise of thyroid cancer,’’ Nature Rev. Endocrinol., vol. 16, no. 1, pp. 17–29, Jan. 2020.

[2] H. R. Shahraki, S. Pourahmad, S. Paydar, and M. Azad, ‘‘Improving the accuracy of early diagnosis of thyroid nodule type based on the SCAD method,’’ Asian Pacific J. Cancer Prevention, vol. 17, no. 4, pp. 1861–1864, Jun. 2016.

[3] J. Xu, M. Jing, S. Wang, C. Yang, and X. Chen, ‘‘A review of medical image detection for cancers in digestive system based on artificial intelligence,’’ Expert Rev. Med. Devices, vol. 16, no. 10, pp. 877–889, Oct. 2019.

[4] H. Ye, J. Hang, X. Chen, D. Xu, J. Chen, X. Ye, and D. Zhang, ‘‘An intelligent platform for ultrasound diagnosis of thyroid nodules,’’ Sci. Rep., vol. 10, no. 1, Aug. 2020, Art. no. 13223.

[5] L. Wang, S. Yang, S. Yang, C. Zhao, G. Tian, Y. Gao, Y. Chen, and Y. Lu, ‘‘Automatic thyroid nodule recognition and diagnosis in ultrasound imaging with the YOLOv2 neural network,’’ World J. Surg. Oncol., vol. 17, no. 1, p. 12, Jan. 2019.

[6] C. Chen, L. Zhan, X. Pan, Z. Wang, X. Guo, H. Qin, F. Xiong, W. Shi, M. Shi, F. Ji, Q. Wang, N. Yu, and R. Xiao, ‘‘Automatic recognition of auditory brainstem response characteristic waveform based on bidirectional long short-term memory,’’ Frontiers Med., vol. 7, Jan. 2021, Art. no. 613708.

[7] D. Koundal, S. Gupta, and S. Singh, ‘‘Computer aided thyroid nodule detection system using medical ultrasound images,’’ Biomed. Signal Process. Control, vol. 40, pp. 117–130, Feb. 2018.

[8] D. Koundal, S. Gupta, and S. Singh, ‘‘Automated delineation of thyroid nodules in ultrasound images using spatial neutrosophic clustering and level set,’’ Appl. Soft Comput., vol. 40, pp. 86–97, Mar. 2016.

[9] Y. Zheng, S. Xu, Z. Zheng, L. Wu, L. Chen, and W. Zhan, ‘‘Ultrasonic classification of multicategory thyroid nodules based on logistic regression,’’ Ultrasound Quart., vol. 36, no. 2, pp. 149–157, Jun. 2020.

[10] W. Sun, S. Xie, J. Yu, L. Niu, and W. Sun, ‘‘Classification of thyroid nodules in ultrasound images using deep model based transfer learning and hybrid features,’’ in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Mar. 2017, pp. 919–923.