DEEP LEARNING MODELS EARLY DETECTION AND CLASSIFICATION OF SKIN CANCER

# A PROJECT REPORT

***Submitted by***

|  |  |
| --- | --- |
| **DHANALAKSHMI K** | **(621320104010)** |
| **SHREYA S B** | **(621319104049)** |

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

# IN

**COMPUTER SCIENCE AND ENGINEERING**

# KONGUNADU COLLEGE OF ENGINEERING AND TECHNOLOGY

**(AUTONOMOUS)**

# THOTTIAM, TRICHY

**ANNA UNIVERSITY :: CHENNAI 600 025**

# MAY - 2024

C:\Users\user\Downloads\logo_new.png**KONGUNADU COLLEGE OF ENGINEERING AND TECHNOLOGY,**

**(AUTONOMOUS)**

**Tholurpatti (Po), Thottiam (Tk), Trichy (Dt) – 621 215 COLLEGE VISION & MISSION STATEMENT**

**VISION**

“To become an Internationally Renowned Institution in Technical Education, Research and Development by Transforming the Students into Competent Professionals with Leadership Skills and Ethical Values.”

**MISSION**

* Providing the Best Resources and Infrastructure.
* Creating Learner centric Environment and continuous –Learning.
* Promoting Effective Links with Intellectuals and Industries.
* Enriching Employability and Entrepreneurial Skills.
* Adapting to Changes for Sustainable Development.

**COMPUTER SCIENCE AND ENGINEERING VISION & MISSION STATEMENT**

**VISION**

To produce competent software professionals, academicians, researchers, and entrepreneurs with moral values through quality education in the field of Computer Science andEngineering.

**MISSION**

* Enrich the students' knowledge and computing skills through an innovative teaching- learning process with state- of- art- infrastructure facilities.
* Endeavour the students to become an entrepreneur and employable through adequate industry-institute interaction.
* Inculcating leadership skills, professional communication skills with moral and ethical values to serve the society and focus on students' overall development

**PROGRAM OUTCOMES (PO’s)**

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identity, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using the first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and the ural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis, and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and teamwork**: Function effectively as an individual, and as a member or leader in diverse teams, and multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-Long Learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO I:** Graduates shall be professionals with expertise in the fields of Software Engineering, Networking, Data Mining, and Cloud Computing and shall undertake Software Development, Teaching and Research.

**PEO II:** Graduates will analyze problems, design solutions, and develop programs with sound Domain Knowledge.

**PEO III:** Graduates shall have professional ethics, team spirit, lifelong learning, good oral and written communication skills, and adopt the corporate culture, core values and leadership skills.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO1:** Professional skills: Students shall understand, analyze and develop computer applications in the field of Data Mining/Analytics, Cloud Computing, Networking, etc., to meet the requirements of industry and society.

**PSO2:** Competency: Students shall qualify at the State, National, and International level competitive examination for employment, higher studies, and research.

# KONGUNADU COLLEGE OF ENGINEERING ANDTECHNOLOGY (AUTONOMOUS)

**ANNA UNIVERSITY :: CHENNAI 600 025**

# BONAFIDE CERTIFICATE

Certified that this project report **“DEEP LEARNING MODELS FOR EARLY DETECTION AND CLASSIFICATION OF SKIN CANCER”** is the bonafide work of **“DHANALAKSHMI K (621320104010), SHERYA S B (621320104049),**

**VIJAYABARATH D (621319104063)”** who carried out the project work under my supervision.

# SIGNATURE SIGNATURE

Dr.C.Saravanabhavan, M.Tech., Ph.D., Mrs.L.Nivetha, M.E.,

# HEAD OF THE DEPARTMENT SUPERVISOR

Department of Computer Science and Professor,

Engineering, Department of Computer Science Kongunadu College of Engineering and and Engineering,

Technology, Thottiam, Trichy. Kongunadu College of Engineering

and Technology, Thottiam, Trichy.

Submitted for the Project Viva-Voce examination held on

# Internal Examiner External Examiner

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks to our beloved Chairman

**Dr.PSK.R.PERIASWAMY** for providing immense facilities in our institution.

We proudly render our thanks to our Principal **Dr.R.ASOKAN, M.S., M.Tech., Ph.D.,** for the facilities and the encouragement was given by him to the progress and completion of our project.

We proudly render our immense gratitude and sincere thanks to our Head of the Department of Computer Science and Engineering **Dr.C.SARAVANABHAVAN, M.Tech., Ph.D.,** for his effective leadership, encouragement, and, guidance in the project.

We are highly indebted to provide our heart full thanks to our supervisor **Mrs.L.NIVETHA, M.E.,** for his valuable suggestion during execution of our project work and for continued encouragement in conveying us for making many constructive comments for improving comments the operation of this project report.

We are highly indebted to provide our heart full thanks to our project coordinator **Ms. V.DHANALAKSHMI, M.E.,**for his valuable ideas, constant encouragement, and supportive guidance throughout the project.

We wish to extend our sincere thanks to all teaching and non-teaching staff of the Computer Science and Engineering department for their valuable suggestion, cooperation, and encouragement in the successful completion of this project.

We wish to acknowledge the help received from various departments and various individuals during the preparation and editing stages of the manuscript.

# ABSTRACT

The Deep learning algorithms for skin cancer detection aiming to improve diagnostic accuracy for skin cancer lesions.. Existing methodologies predominantly rely on visual inspection exhibiting limitations in less accuracy. The utilization of deep learning models in this context results in early diagnosis of skin cancer . Proposed methodologies contains a set of algorithms such as Convolutional Neural Network (CNN), Support Vector Machine (SVM), and decision tree. The algorithms accuracy is improved when applied to diverse skin tones. The proposed system introduces an inclusive dataset that encompasses diverse skin tones, including darker skin types. Thereby mitigating the color bias present in existing datasets. the dataset to represent various skin types comprehensively. The proposed system aims to significantly enhance the accuracy of skin cancer detection through deep learning methodologies and it represents a step towards more accuracy of skin cancer detection, which provides an advancement in improved healthcare sectors.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF CONTENT** |  |
| **CHAPTER No.** | **TITLE** | **PAGE No.** |
|  | **ABSTRACT** | **viii** |
|  | **LIST OF FIGURES** | **xi** |
|  | **LIST OF ABBREVIATIONS** | **xii** |
| **1** | **INTRODUCTION** | **1** |
| **2** | **LITERATURE SURVEY** | **2** |
| **3** | **SYSTEM ANALYSIS** | **5** |
|  | 3.1 EXISTING SYSTEM | 5 |
|  | 3.1.1 DISADVANTAGES | 5 |
|  | 3.2 PROPOSED SYSTEM | 5 |
|  | * + 1. ADVANTAGES     2. DISADVANTAGES | 6  6 |
| **4** | **SYSTEM REQUIREMENTS** | **7** |
|  | 4.1 HARDWARE REQUIREMENTS | **7** |
|  | 4.2 SOFTWARE REQUIREMENTS | 7 |
|  | 4.3 SOFTWARE DESCRIPTION | 7 |
| **5** | **SYSTEM DESIGN** | **13** |
|  | 5.1 SYSTEM BLOCK DIAGRAM | 13 |
|  | 5.2 ARCHITECTURE DIAGRAM | 14 |
|  | 5.3 ARCHITECTURE DIAGRAM EXPLANATION | 15 |
| 6 | **SYSTEM IMPLEMENTATION** | **16** |
|  | 6.1 MODULES | 16 |
|  | 6.2 MODULE DESCRIPTION | 16 |
|  | 6.2.1 TEST MODULE | 16 |
|  | 6.2.2 TRAIN MODULE | 17 |
|  | 6.2.3 SEGMENTATION MODULE | 17 |
|  | 6.2.4 BACKEND MODULE | 18 |
| **7** | **ALGORITHM** | 19 |
|  | 7.1 ALGORITHM DETAILS | 19 |
| **8** | **RESULT AND DISCUSSION** | 22 |
| **9** | **CONCLUSION** | 25 |
|  | APPENDIX | 27 |
|  | REFERENCES | 56 |

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 5.1 | Block Diagram of the proposed system | 13 |
| 5.2 | Architecture Diagram | 14 |
| 7.1 | Working of CNN | 19 |
| 7.2 | Mechanism of pooling layers | 21 |
| 8.1 | Developed Web Page | 22 |
| 8.2 | Interior Page | 22 |
| 8.3 | Page to Upload the image | 23 |
| 8.4 | Result Page | 23 |
| 8.5 | Result Page | 24 |

|  |  |
| --- | --- |
|  | **LIST OF ABBREVIATIONS** |
| ISIC | **I**nternational **S**kin **I**maging **C**ollaboration |
| CNN | **C**onvolutional **N**eural **N**etwork |
| LDED | **L**ocal **D**ouble **E**llipse **D**escriptor |
| ALDS | **A**utomatic **L**esion **D**etection **S**ystem |
| HTML | **H**yper **T**ext **M**arkup **L**anguage |
| CSS | **C**ascading **S**tyle **S**heets |
| DOM | **D**ocument **O**bject **M**odel |
| API | **A**pplication **P**rogram **I**nterface |
| ECMA | **E**uropean **C**omputer **M**anufacturers **A**ssociation |

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| NLTK | **N**atural **L**anguage **T**oolkit |
| JSON | **J**ava**S**cript **O**bject **N**otation |
| HTML | **H**ypertext **M**arkup **L**anguage |
| CSS | **C**ascading **S**tyle **S**heet |
| AI | **A**rtificial **I**ntelligence |
| ML | **M**achine **L**earning |
| JS | **J**ava**S**cript |
| DL | **D**eep **L**earning |
| RNN | **R**ecurrent **N**eural **N**etwork |
| LSTM | **L**ong **S**hort **T**erm **M**emory |
| GUI | **G**raphic **U**ser **I**nterface |
| NLP | **N**atural **L**anguage **P**rocessing |
| HTTP | **H**ypertext **T**ransfer **p**rotocol |
| UI | **U**ser **I**nterface |
| UX | **U**ser **E**xperience |

**CHAPTER 1**

**INTRODUCTION**

* 1. **OVERVIEW**

Skin cancer is an alarming disease among mankind. People in the USA, Canada, and Australia have been diagnosed at the highest increasing rate over the past few decades. Skin cancer happens due to the uneven development of melanocytic skin cells. Among all skin cancers, malignant and benign are the deadliest. A malignant tumor is a type of cancerous tumor that spreads and expands in a patient’s body. Many malignant skin growths have symptoms that can be identified as precursors. A precursor is a group of aberrant cells that may develop into cancer. Precancerous is another term for a precursor. Some precancerous skin growths have a minimal chance of developing into cancer, whereas others have a very high chance. The importance of detecting and treating cancer in early malignant skin growth cannot be overstated. For this reason, CNN is proposed in the systems of this study to detect skin cancer. For the first time, this project achieved an accuracy of 80.3% using deep CNN on the International Skin Imaging Collaboration (ISIC) dataset. Although this project worked on the same dataset, this time the project is based CNN symptomatic feature extraction. After segmenting the dermoscopic images using the feature extraction approach, the characteristics of the skin cells are retrieved in this work. Finally, this project got an accuracy of 89.5%. The project has worked in many ways with CNN architecture to detect skin cancer. Through this, the project achieved a training accuracy of 93.7% using a publicly available data set.

**CHAPTER** **2**

**LITERATURE SURVEY**

**TOPIC : AN AUTOMATED COMPUTER-AIDED DIAGNOSIS OF SKIN LESIONS DETECTION AND CLASSIFICATION FOR DERMOSCOPY IMAGES.**

**AUTHORS :**R.Suganya

**SUMMARY :**

In the diagnosis of skin melanoma by analyzing histopathological images, the detection of melanocytes in the epidermis area is an important step. This paper [1] proposes a novel computer-aided technique for the segmentation of melanocytes in skin histopathological images. To reduce the local intensity variant, a CNN algorithm is applied for the initial segmentation of the image. A local region recursive segmentation algorithm is then proposed to filter out the candidate nuclei regions based on the domain’s prior knowledge. To distinguish the melanocytes from other keratinocytes in the epidermis area, a novel descriptor, named local double ellipse descriptor (LDED), is proposed to measure the local features of the candidate regions. The LDED uses two parameters: region ellipticity and local pattern characteristics to distinguish the melanocytes from the candidate nuclei regions. Experimental results on 28 different histopathological images of skin tissue with different zooming factors show that the proposed technique provides superior performance. Merits : CNN has the highest accuracy among all algorithm that predicts images. Demerits : Shows difficulty in identifying melanocytes.

## TOPIC : A NOVEL HYBRID SYSTEM FOR SKIN LESION DETECTION.

**AUTHORS :** Chiem, A. Al-Jumaily, and R. N. Khushaba.

## SUMMARY :

Melanoma is the deadliest form of skin cancer. Incidence rates of melanoma have been increasing, especially among non-Hispanic white males and females, but survival rates are high if detected early. Due to the costs for dermatologists to screen every patient, there is a need for an automated system to assess a patient's risk of melanoma using images of their skin lesions captured using a standard digital camera [2]. One challenge in implementing such a system is locating the skin lesion in the digital image. A novel texture-based skin lesion segmentation algorithm is proposed. A set of representative texture distributions are learned from an illumination-corrected photograph and the texture distinctiveness metric is calculated for each distribution. Next, regions in the image are classified as normal skin or lesions based on the occurrence of representative texture distributions. The proposed segmentation framework is tested by comparing lesion segmentation results and melanoma classification results to results using other state-of-art algorithms. The proposed framework has higher segmentation accuracy compared to all other tested algorithms.

**Merits :**Texture-based analysis is utilized above can be helpful when the object in an image are more characterized by texture rather than intensity.

**Demerits :** Classification of skin cell image as benign and malignant.

## TOPICS : AUTOMATIC LESION DETECTION SYSTEM (ALDS) FOR SKIN CANCER CLASSIFICATION USING SVM AND NEURAL CLASSIFIERS.

**AUTHORS :** M. A. Farooq, M. A. M. Azhar, and R. H. Raza

## SUMMARY :

Malignant melanomas are asymmetrical and have irregular borders, notched edges, and color variations, so analyzing the shape, color, and texture of the skin lesion is important for the early detection and prevention of melanoma. This paper proposes the two major components of a noninvasive real-time automated skin lesion analysis system for the early detection and prevention of melanoma. The first component is a real-time alert to help users prevent skin burns caused by sunlight; a novel equation to compute the time for skin to burn is thereby introduced. The second component is an automated image analysis module, which contains image acquisition, hair detection and exclusion, lesion segmentation, feature extraction, and classification. The proposed system [3] uses the PH2 Dermoscopy image database from Pedro Hispano Hospital for development and testing purposes. The image database contains a total of 200 dermoscopy images of lesions, including benign, atypical, and melanoma cases. The experimental results show that the proposed system is efficient, achieving classification of the benign, atypical, and melanoma images with an accuracy of 96.3%, 95.7%, and 97.5%, respectively.

**Merits :** It produces results with a high degree of accuracy.

**Demerits :** High computational cost.

## CHAPTER 3

**SYSTEM ANALYSIS**

## 3.1 EXISTING SYSTEM

Skin cancer diagnosis requires a skin biopsy. A sample of skin tissue is taken by your doctor and submitted to a lab. A pathologist examines the material under a microscope in the lab. A pathologist searches for unusual cells that could be cancerous. If cancer is present, the biopsy sample offers crucial details regarding cancer's stage.

## 3.1.1 Disadvantages

The risk of recurrence plays a major disadvantage in manual detection in case, if it spreads within the detection then there will be no chance for diagnosis. Many melanomas can be cured with surgery if caught early. Melanoma that has metastasized has a possibility of curing radiotherapy but after several stages, it can’t be treated. The time required risk for the detection takes too long.

## 3.2 PROPOSED SYSTEM

Using deep learning algorithms like Convolutional Neural Network (CNN), which incorporates image processing, this project offered a web-based detection of skin cell detection. The test results will be shown on the website after the photos of the skin cells from the test report have been processed using image processing.

## Advantages

* + - * The outcomes will be shown promptly. The time commitment will be cut down.
      * All of the remote members may access the website if it is hosted.

## Disadvantages

* Accuracy needs to be improved.
* Network facility essential for progress

## CHAPTER 4 SYSTEM DESIGN

**SYSTEM REQUIREMENT**

## 

## 4.1 HARDWARE REQUIREMENT

* + - **Processor :** i5 intel core
    - **RAM :** 4GB
    - **Keyboard :** Standard keyboard

## Hard disk : 4GB

* + - **Monitor :** 15.6-inch color monitor

## 

## 4.2 SOFTWARE REQUIREMENT

* + - **Operating system** : Windows OS
    - **Editor** : Sublime Text Editor
    - **Front End** : HTML, CSS, JS
    - **Back End** : Python
    - **Programming Language** : Python

## 

## 4.3 SOFTWARE DESCRIPTION

Frond End: Python

Python is an interpreted high-level programming language for general- purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional, and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open- source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non- profit Python Software Foundation. Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a complimenting the Python culture. "Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favor of "there should be one and preferably only one obvious way to do it".

Python's developers strive to avoid premature optimization and reject patches to non-critical parts of C Python that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. C Python is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python developers is to keep it fun to use.

A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluencyin the language, and that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic. Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonists, Pythonistas, and Pythoners. 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. 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.

Python’s initial development was spearheaded by Guido van Rossum in the late 1980s. Today, it is developed by the Python Software Foundation. Because Python is a multi paradigm language, Python programmers can accomplish their tasks using different styles of programming: object-oriented, imperative, functional, or reflective. Python can be used in Web development, numeric programming, game development, serial port access, and more.

Two attributes make development time in Python faster than in other programming languages:

1. Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice

programmers.

1. Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.

## Front End : HTML

The Hyper Text Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content to the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

## Front End : CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of content and presentation, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the. CSS file to be cached to improve the page load speed between the pages that share the file and its formatting Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on- screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device. The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

## Front End : JavaScript

JavaScript (JS) is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third- party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices. JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMA Script standard. It has dynamic typing, prototype-based object- orientation, and first-class functions. It is a multi- paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). The ECMA Script standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O. JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js. Although Java and JavaScript are similar in name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

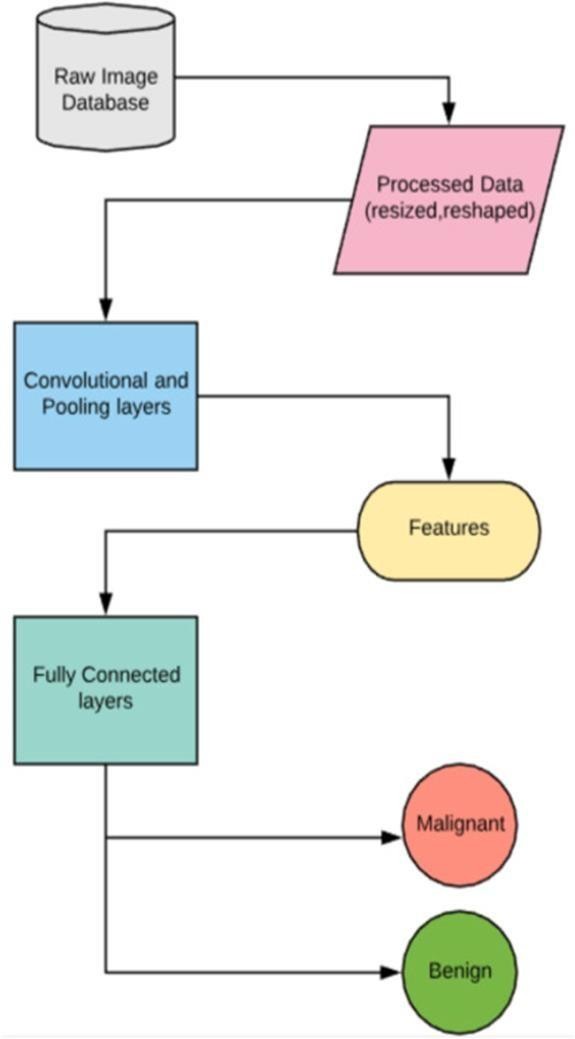
## Back End: Flask

Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre- existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies, and several common framework-related tools. Applications that use the Flask framework include Pinterest and LinkedIn.

## CHAPTER 5 SYSTEM DESIGN

* 2. **5.1 SYSTEM BLOCK DIAGRAM**

## Figure 5.1 Block Diagram of the proposed system



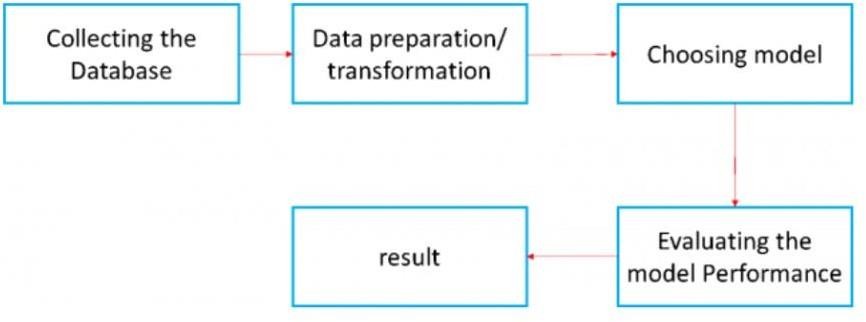
* 1. **5.2 ARCHITECTURE DIAGRAM**

The Architecture diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and detailed modeling translating the models into programming code.

## Figure 5.2: Architecture Diagram

* 1. **5.3 ARCHITECTURE DIAGRAM EXPLANATION**

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.



* Provide the project's web pages with the microscopic photos as input.
* Along with the input image choose the gender and age.
* After certain preprocessing, segmentation, and feature extraction the output of the given image is displayed on the web page within a few seconds.
* So, the time consumption is less as compared to the existing system

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

## 6.1 MODULES

A module is a separate unit of software or hardware. Typical characteristics of modular components include portability, which allows them to be used in a variety of systems, and interoperability which allows them to function with the components of other systems. The modules used in this project are

* + - Test Module
    - Train Module
    - Segmentation Module
    - Backend Module

## 6.2 MODULES DESCRIPTION

**6.2.1 Test Module**

In Deep learning, model testing is referred to as the process where the performance of a fully trained model is evaluated on a testing set. Python provides the unit test module to test the unit of source code. The unit test plays an essential role when the project needs huge code, and it provides the facility to check whether the output is correct or not. Normally, print the value and match it with the reference output or check the output manually. You test cases with positive inputs, inputs with zero, negative inputs, and positive and negative inputs. If the output of our function/method being tested would be equal to the outputs defined in the unit test for all the input cases, your unit would pass the test otherwise it would fail.

## 6.2.2 Train Module

A smart algorithm that can obtain sensitivity from a provided set of training data. A step-by-step procedure for adjusting the connection weights of an artificial neural network. In supervised training, the desired (correct) output for each input vector of a training set is presented to the network, and many iterations through the training data may be required to adjust the weights. In unsupervised training, the weights are adjusted without specifying the correct output for any of the input vectors. Training data is the data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict. If you are using supervised learning or some hybrid that includes that approach, your data will be enriched with data labeling or annotation. programmers input data and the desired behavior to produce the logic of the machine. It's tested repeatedly to see if the learned logic remains consistent. If it does, it ensures that the system understands the logic and develops a model according to the desired behavior.

## 6.2.3 Segmentation Module

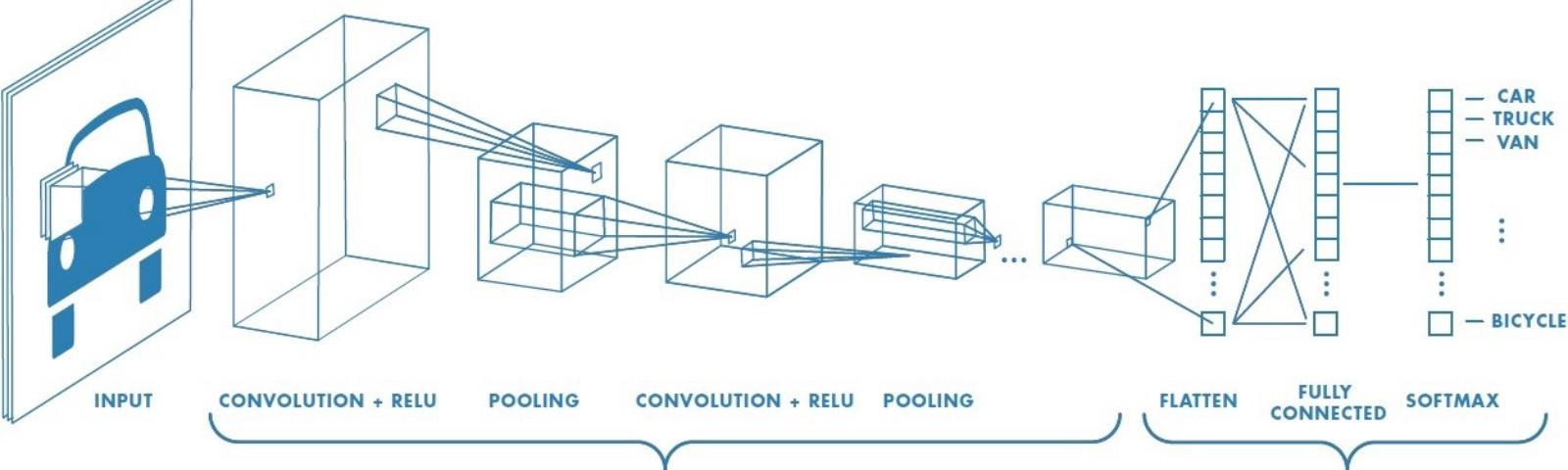
Segmentation refers to the process of partitioning a digital image into multiple segments. In other words, a grouping of pixels into different groups is known as Segmentation. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics The division of an image into meaningful structures, image segmentation, is often an essential step in image analysis, object representation, visualization, and many other image processing tasks. But segmentation of a satellite image into differently textured regions (groups) is a difficult problem. One does not know a prior what types of textures exist in a satellite image, how many textures there are, and what regions have certain textures. A region of interest (ROI) is a subset of an image or a dataset identified for a particular purpose. In other words, region of interest (ROI) can be defined as a portion of an image which is needed to be filtered or to be performed some other operation on.

## 6.2.4 Design and Backend Module

Python is one of the most powerful and best multi-purpose languages used for Web Development, as well as data analysis. Back-end developers are the experts who build and maintain the mechanisms that process data and perform actions on websites. Unlike front-end developers, who control everything you can see on a website, back-end developers are involved in data storage, security, and other server-side functions that you cannot see on the server. This is the computer that receives requests. This is the application running on the server that listens for requests, retrieves information from the database, and sends a response. Databases are used to organize and persist data. The back end, also called the server side, consists of the server which provides data on request, the application that channels it, and the database which organizes the information. For example, when a customer browses shoes on a website, they are interacting with the front end. The source code of a website is easily accessible from your browser. In Chrome, look for Developer Tools, in Firefox look for Web Developer in your menu. The source code's file extensions and URLs can tell you.

## CHAPTER 7 ALGORITHM

**7.1 ALGORITHM DETAILS**

IDP leverages a deep learning network known as CNN (Convolutional Neural Networks) to learn patterns that naturally occur in photos. IDP is then able to adapt as new data is processed, using Image net, one of the biggest databases of labeled images, which has been instrumental in advancing computer vision. Deep Learning models, with their multi-level structures, as shown above, are very helpful in extracting complicated information from input images. Convolutional neural networks are also able to drastically reduce computation time by taking advantage of GPU for computation, which many networks fail to utilize. Deep learning algorithms run data through several “layers” of neural network algorithms, each of which passes a simplified representation of the data to the next layer. Most machine learning algorithms work well on datasets that have up to a few hundred features, or columns.

## Figure 7.1: Working Of CNN

**Convolution**

In deep learning, a convolutional neural network (CNN or Conv Net) is a class of deep neural networks, that are typically used to recognize patterns present in images but they are also used for spatial data analysis, computer vision, natural language processing, signal processing, and various other purposes.

## Pooling

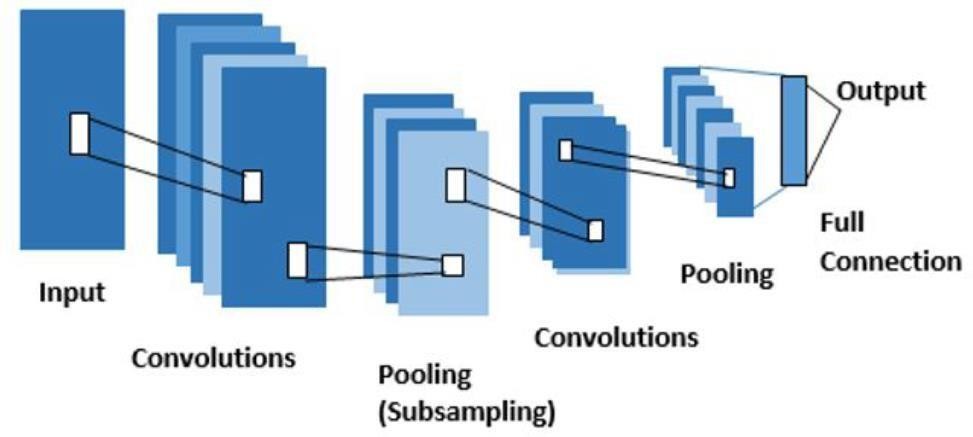
Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarises the features present in a region of the feature map generated by a convolution layer. The pooling operation involves sliding a two-dimensional filter over each channel of the feature map and summarising the features lying within the region covered by the filter. In a convolutional neural network, pooling layers are applied after the convolutional layer. The pooling layer is used to reduce the size of the feature.

## Flatten

Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fedas input to the fully connected layer to classify the image. A tensor flattens operation is a common operation inside convolutional neural networks. This is because convolutional layer outputs that are passed to fully connected layers must be flatted out before the fully connected layer will accept the input. A flatten operation is a specific type of reshaping operation where all of the axes are smooshed or squashed together. To flatten a tensor, this project needs to have at least two axes. This makes it so that the project is starting with something that is not already flat.

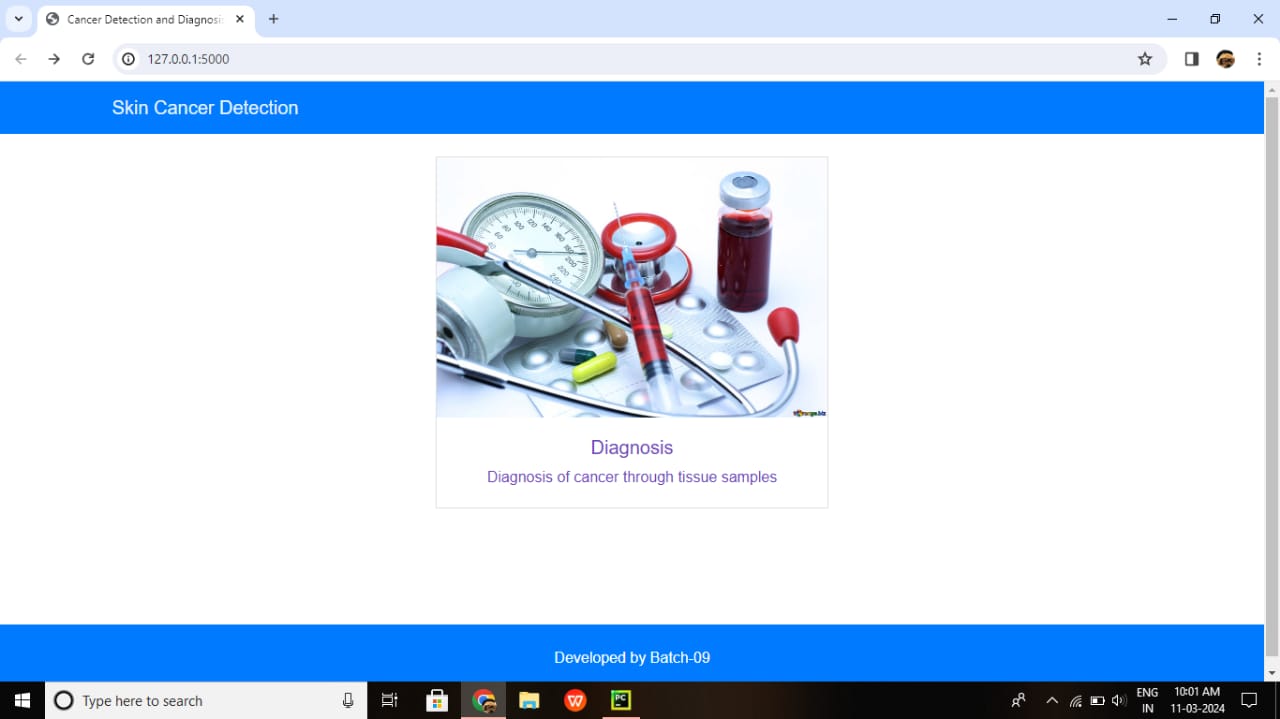
## Fully connected Layer

Each function consists of a neuron (or a perceptron). In fully connected layers, the neuron applies a linear transformation to the input vector through a weights matrix. A non-linear transformation is then applied to the product through a non-linear activation function f. A neuron in a fully connected layer is connected to every neuron in the preceding layer, and so can change if any of the neurons from the preceding layer changes. A neuron in a convolutional layer, however, is only connected to "nearby" neurons from the preceding layer within the width of the convolutional kernel. A fully Connected Layer simply, feeds forward neural networks. Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer. A fully connected layer is the last part of a convolutional neural network. It can be a single Dense layer or a complex multilayer perceptron. The inputs of this layer come from the flattened layer. Therefore, the flattened layer can be seenas the input layer of this component.



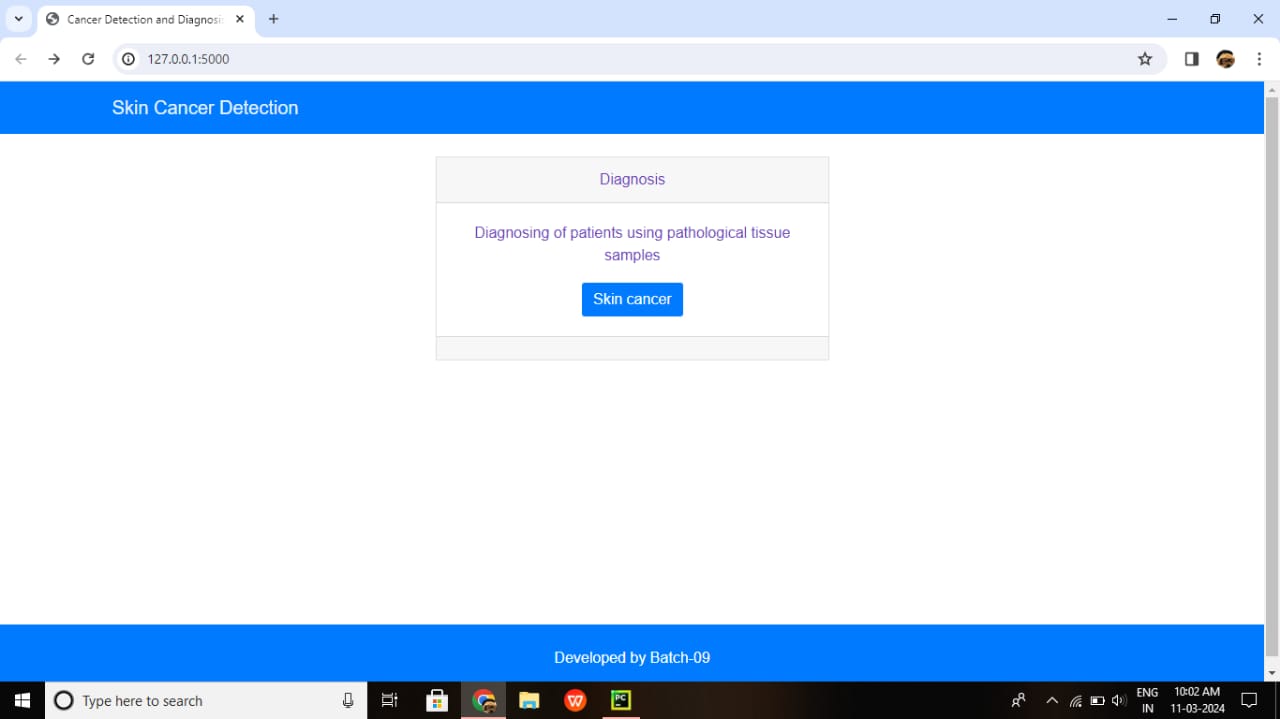
## Figure 7.2: Mechanism of Pooling Layers

**CHAPTER 8** **RESULTS AND DISCUSSION**

****

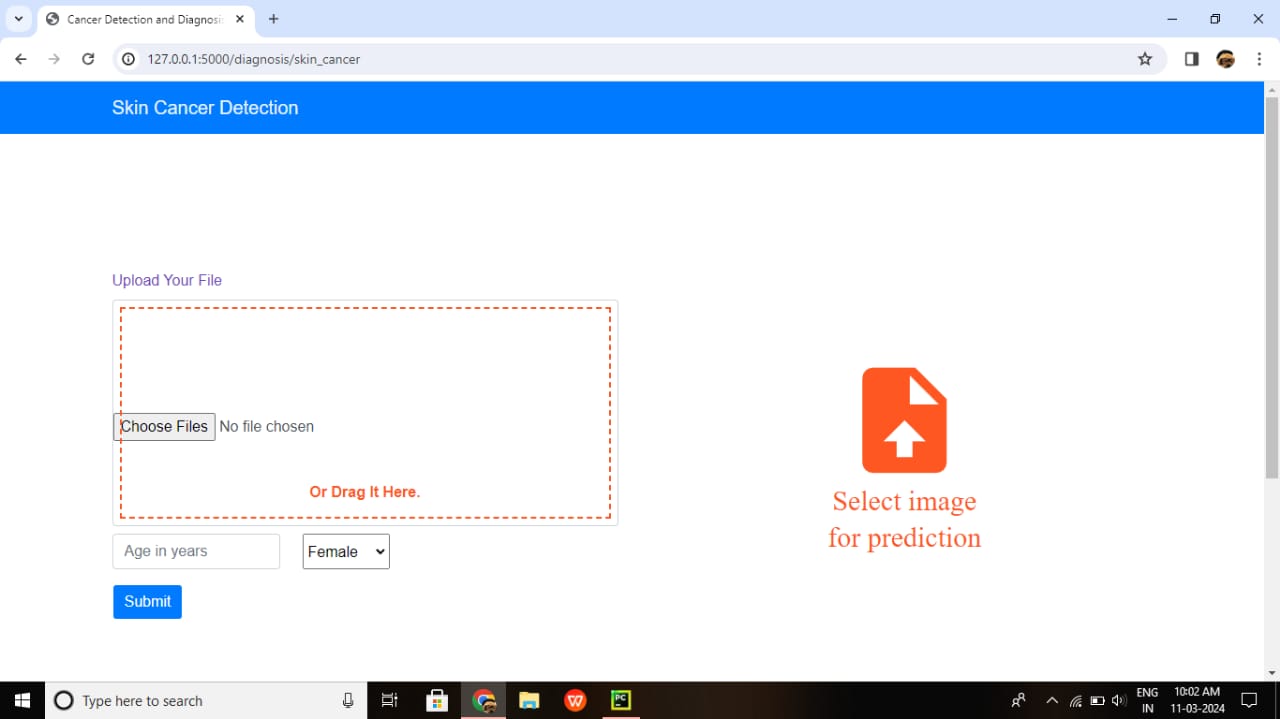
## Figure 8.1: Developed web page

* Give the microscopic images as the input for web pages .



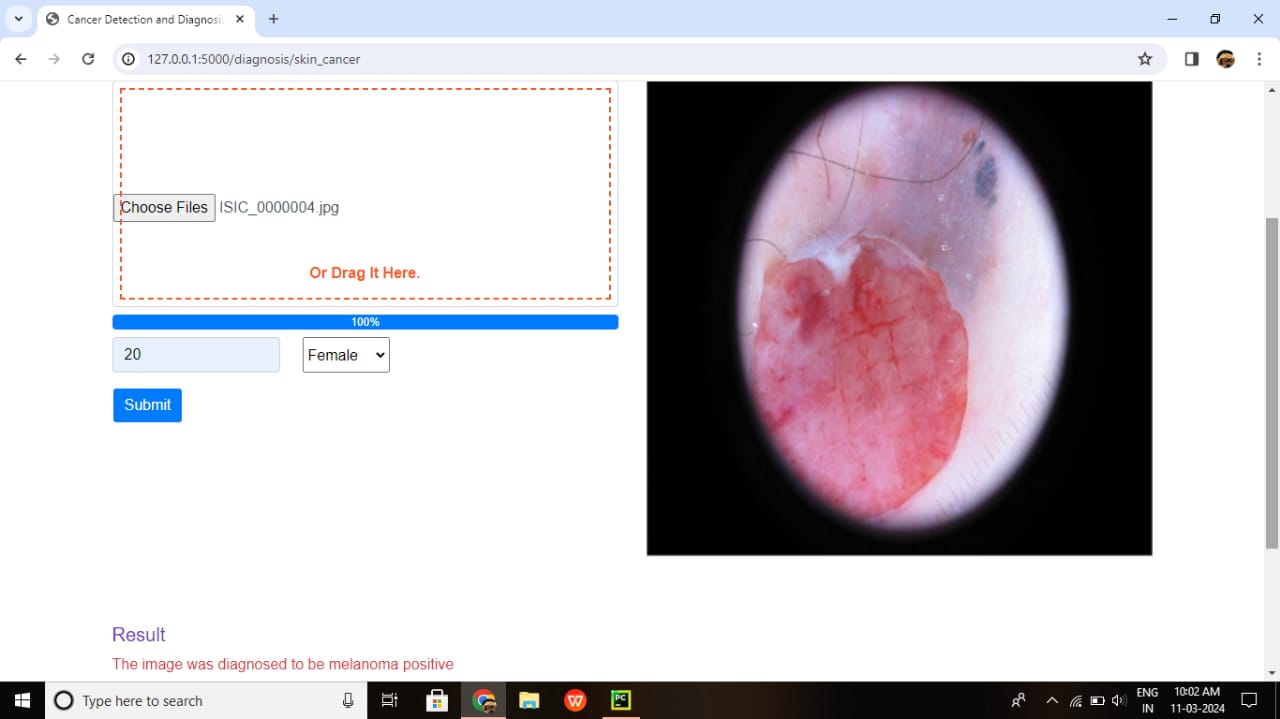
## Figure 8.2: Interior Page

* + - Interior page to help users understand the functions of webpage.



## Figure 8.3: Page to upload the image

* + - Along with the input image choose the gender and age.



## Figure 8.4: Result is Displayed on the Web page as positive

* + - After certain preprocessing, segmentation, and feature extraction the output of the given image is displayed on the webpage within a few seconds.



## Figure 8.5: Result is Displayed on the Web page as Negative

* + - The time consumption is less as compared to the existing system

## CHAPTER 9

**CONCLUSION**

This project works on neural network techniques for skin cancer detection and classification. Skin cancer detection requires multiple stages, such as preprocessing and image segmentation, followed by feature extraction and classification. This review focused on CNNs, for the classification of lesion images. Each algorithm has its advantages and disadvantages. Proper selection of the classification technique is the core point for best results. However, CNN gives better results than other types of neural networks when classifying image data because it is more closely related to computer vision than others. Most of the research related to skin cancer detection focuses on whether a given lesion image is cancerous. However, when a patient asks if a particular skin cancer symptom appears on any part of their body, the current research cannot provide an answer. Thus far, the research has focused on the narrow problem of classification of the signal image. Future research can include full-body photography to seek the answer to the question that typically arises. Autonomous full-body photography will automate and speed up the image acquisition phase. As a result, deep learning- based methods such as Convolutional Neural Networks (CNN) have been used to solve the great majority of skin cancer classification problems in recent years and obtained satisfactory results. The biggest advantage of Deep Learning algorithms as discussed before is that they try to learn high-level features from data in an incremental manner. This eliminates the need for domain expertise and hardcore feature extraction. Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example.

## Future Enhancement

The idea of auto-organization has recently emerged within the area of deep learning. Auto-organization refers to the process of unsupervised learning, which aims to identify features and discover relations or patterns in the image samples of the dataset. Under the umbrella of convolutional neural networks, auto- organization techniques increase the level of feature representation that is retrieved by expert systems. Currently, auto-organization is a model that is still in research and development. However, its study can improve the accuracy of image processing systems in the future, particularly in the area of medical imaging, where the smallest details of features are extremely crucial for the correct diagnosis of disease. New cell-based therapy for melanoma more effective than existing treatment, trial finds. Researchers in the Netherlands found that a therapy that uses a patient's immune cells slowed the progression of metastatic melanoma.

## APPENDIX

import base64 import ntpath import os import random

from io import BytesIO

import numpy as np import pandas as pd from PIL import Image

from flask import Flask, render\_template, request, jsonifyfrom torch import Tensor, load

from SkinCancer.Model import SkinCancerModel app = Flask( name )

def serve\_pil\_image(pil\_img

):img\_io = BytesIO() pil\_img.save(img\_io, format='JPEG', quality=100)img\_io.seek(0)

img\_i = base64.b64encode(img\_io.getvalue()).decode() return {'image': img\_i}

def age():

pass

@app.route('/diagnosis/skin\_cancer', methods=['GET', 'POST'])def skin\_cancer(age=None):

def get\_prediction(pil\_img, age, gender): pil\_img = pil\_img.resize([128, 128]) age\_t = Tensor([age]).reshape(1, 1) gender\_t = Tensor([gender]).reshape(1, 1) img\_t = Tensor(np.array(pil\_img)).reshape(1, 3, 128,

128)model = SkinCancerModel() model.load\_state\_dict(load('SkinCancer/model\_skin\_cancer\_epoch40.pt')) model.eval()

pred = np.round(model.forward(img\_t, gender\_t, age\_t).detach().numpy())

return int(pred)

if request.method == 'GET':

return render\_template('skin- cancer.html')else:

if 'use\_random' in request.form.keys():

if int(request.form.get('use\_random')) == 1:

random\_path = os.path.join('SkinCancer/RandomData/images', random.choice(os.listdir('SkinCancer/RandomData/images')))

file\_name = ntpath.basename(ntpath.splitext(random\_path)[0]) metadata = pd.read\_csv('SkinCancer/RandomData/meta\_data.csv')row = metadata.loc[(metadata['image\_id'] == file\_name)]

gender = row['sex'].item()

if gender == 'male':gender = 1 elif gender == 'female':gender = 0 else:

gender = -1

img = Image.open(random\_path).convert('RGB') res = serve\_pil\_image(img.resize([400, 400])) res['prediction'] = get\_prediction(img, age, gender)return jsonify(res)

else:

print('File: {}'.format(request.files.get('skin-cancer-diagnosis- image')))age = int(request.form.get('age'))

gender = request.form.get('gender')

img = Image.open(request.files.get('skin-cancer- diagnosis-image')).convert('RGB')

if gender == 'Female':gender = 0 elif gender == 'Male':gender = 1 else:

gender = -1

res = serve\_pil\_image(img.resize([400, 400])) res['prediction'] = get\_prediction(img, age,

gender)return jsonify(res)

return jsonify({'error': 1}) @app.route('/',

methods=['GET'])def home():

return render\_template('home.html')

if name == ' main ':app.run(debug=True)

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink- to-fit=no">

<title>Cancer Detection and Diagnosis</title>

<script src="{{ url\_for('static', filename='js/jquery.min.js')}}" type="text/javascript"></script>

<script src="{{ url\_for('static', filename='js/bootstrap.min.js')}}" type="text/javascript"></script>

<script src="{{ url\_for('static', filename='js/tether.min.js')}}" type="text/javascript"></script>

<script src="{{ url\_for('static', filename='js/custom.js')}}" type="text/javascript"></script>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"

integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg… [10:06 PM, 11/10/2022] Amuthan Kncet: <!DOCTYPE html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink- to-fit=no">

<title>Cancer Detection and Diagnosis</title>

<script src="{{ url\_for('static', filename='js/jquery.min.js')}}" type="text/javascript"></script>

<script src="{{ url\_for('static', filename='js/bootstrap.min.js')}}" type="text/javascript"></script>

<script src="{{ url\_for('static', filename='js/tether.min.js')}}" type="text/javascript"></script>

<script src="{{ url\_for('static', filename='js/dynamic.js')}}" type="text/javascript"></script>

<script src="{{ url\_for('static', filename='js/skin-cancer-diagnosis.js')}}" type="text/javascript"></script>

<!-- CSS -->

<link rel="stylesheet" href…

[10:06 PM, 11/10/2022] Amuthan Kncet: from flask import Flask, render\_template, request, jsonify, send\_file, make\_response import pandas as pd

from PIL import Image, ImageDraw import random

import os import ntpath import base64 import pickle

from io import BytesIO

from torch import Tensor, load import numpy as np

from SkinCancer.Model import SkinCancerModel app = Flask(\_name\_)

def serve\_pil\_image(pil\_img): img\_io = BytesIO()

pil\_img.save(img\_io, format='JPEG', quality=100) img\_io.seek(0)

img\_i = base64.b64encode(img\_io.getvalue()).decode() return {'image': img\_i}

@app.route('/diagnosis/skin\_cancer', methods=['GET', 'POST']) def skin\_cancer():

def get\_prediction(pil\_img, age, gender): pil\_img = pil\_img.resize([128, 128]) age\_t = Tensor([age]).reshape(1, 1)

img\_t = Tensor(np.array(pil\_img)).reshape(1, 3, 128, 12 model = SkinCancerModel()

model.load\_state\_dict(load('SkinCancer/model\_skin\_cancer\_epoch40.pt')

)

model.eval()

pred = np.round(model.forward(img\_t, gender\_t,

age\_t).detach().numpy())

return int(pred)

if request.method == 'GET':

return render\_template('skin-cancer.html')

else:

if 'use\_random' in request. form.keys():

if int(request.form.get('use\_random')) == 1: random\_path = os.

path.join('SkinCancer/RandomData/images',

random.choice(os.listdir('SkinCancer/RandomData/images'))) file\_name =

path.basename(ntpath.splitext(random\_path)[0])

metadata = pd.read\_csv('SkinCancer/RandomData/meta\_data.csv')

row = metadata.loc[(metadata['image\_id'] ==

file\_name)]

gender = row['sex'].item() if gender == 'male':

gender = 1

elif gender == 'female': gender = 0

else:

gender = -1

diagnosis-image')))

else:

img = Image.open(random\_path).convert('RGB') res = serve\_pil\_image(IMG.resize([400, 400])) res['prediction'] = get\_prediction(IMG, age, gender) return jsonify(res)

print('File: {}'.format(request.files.get('skin-cancer- age = int(request.form.get('age'))

gender = request. form.get('gender')

img = Image.open(request.files.get('skin-cancer-

diagnosis-image')).convert('RGB')

if gender == 'Female': gender = 0

elif gender == 'Male':

gender = 1

else:

gender = -1

res = serve\_pil\_image(img.resize([400, 400])) res['prediction'] = get\_prediction(img, age, gender) return jsonify(res)

return jsonify({'error': 1})

**Dataloader.py**

class6 = fnmatch.filter(data, '/home/aftaab/MylanDatasets/colorectal-histology-mnist/Patches/06\_MUCOSA/\*.tif')

class7 = fnmatch.filter(data, '/home/aftaab/MylanDatasets/colorectal-histology-mnist/Patches/07\_ADIPOSE/\*.tif')

class0 = fnmatch.filter(data, '/home/aftaab/MylanDatasets/colorectal-histology-mnist/Patches/08\_EMPTY/\*.tif')

train, test = train\_test\_split(data, test\_size=0.05, shuffle=True)

train, test = train\_test\_split(data, test\_size=0.1, shuffle=True)

class TrainDataset(Dataset):

@@ -24,9 +24,9 @@ def \_\_init\_\_(self):

self.train = train

def \_\_getitem\_\_(self, index):

img = Image.open(train[index]).convert('RGB')

img = Image.open(train[index]).convert('L')

img = img.resize([150, 150])

img = np.array(img).reshape(3, 150, 150)

img = np.array(img).reshape(1, 150, 150)

if train[index] in class1:

y = 1

elif train[index] in class2:

@@ -44,7 +44,7 @@ def \_\_getitem\_\_(self, index):

else:

y = 0

img = Tensor(img/255.0).view(3, 150, 150)

img = Tensor(img/255.0).view(1, 150, 150)

return img, y

def \_\_len\_\_(self):

@@ -58,9 +58,9 @@ def \_\_init\_\_(self):

self.test = test

def \_\_getitem\_\_(self, index):

img = Image.open(train[index]).convert('RGB')

img = Image.open(train[index]).convert('L')

img = img.resize([150, 150])

img = np.array(img).reshape(3, 150, 150)

img = np.array(img).reshape(1, 150, 150)

if test[index] in class1:

y = 1

elif test[index] in class2:

@@ -78,7 +78,7 @@ def \_\_getitem\_\_(self, index):

else:

y = 0

img = Tensor(img/255.0).view(3, 150, 150)

img = Tensor(img/255.0).view(1, 150, 150)

return img, y

def \_\_len\_\_(self):

1. changes: 2 additions & 4 deletions6

**Model.py**

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.conv1 = nn.Conv2d(1, 3, 3, stride=2)

self.conv1 = nn.Conv2d(3, 3, 3, stride=2)

self.pool1 = nn.MaxPool2d(3, stride=2)

self.conv2 = nn.Conv2d(3, 4, 5, stride=2)

self.pool2 = nn.AvgPool2d(3, stride=2)

self.pool2 = nn.MaxPool2d(3, stride=2)

self.fc1 = nn.Linear(4 \* 6 \* 6, 24)

self.dropout = nn.Dropout(0.4)

self.fc2 = nn.Linear(24, 1)

@@ -22,21 +22,16 @@ def \_\_init\_\_(self):

def forward(self, x, g, a):

x = self.conv1(x)

x = self.pool1(x)

x = leaky\_relu(x)

x = relu(x)

x = self.conv2(x)

x = self.pool2(x)

x = leaky\_relu(x)

x = x.view(-1, 4 \* 6 \* 6)

x = self.fc1(x)

x = self.dropout(x)

x = sigmoid(self.fc2(x))

x = leaky\_relu(self.fc2(x))

a = a.view(-1, 1)

g = g.view(-1, 1)

print(x.size())

print(a.size())

print(g.size())

x = torch.cat((x, g, a), 1)

x = torch.cat((x, g, a), 1)

x = self.meta\_fc3(x)

x = leaky\_relu(x)

x = self.out(x)

return sigmoid(x)

1. changes: 4 additions & 5 deletions9

**Test.py**

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.conv1 = nn.Conv2d(1, 3, 3, stride=2)

self.conv1 = nn.Conv2d(3, 3, 3, stride=2)

self.pool1 = nn.MaxPool2d(3, stride=2)

self.conv2 = nn.Conv2d(3, 4, 5, stride=2)

self.pool2 = nn.AvgPool2d(3, stride=2)

self.pool2 = nn.MaxPool2d(3, stride=2)

self.fc1 = nn.Linear(4 \* 6 \* 6, 24)

self.dropout = nn.Dropout(0.4)

self.fc2 = nn.Linear(24, 1)

@@ -22,21 +22,16 @@ def \_\_init\_\_(self):

def forward(self, x, g, a):

x = self.conv1(x)

x = self.pool1(x)

x = leaky\_relu(x)

x = relu(x)

x = self.conv2(x)

x = self.pool2(x)

x = leaky\_relu(x)

x = x.view(-1, 4 \* 6 \* 6)

x = self.fc1(x)

x = self.dropout(x)

x = sigmoid(self.fc2(x))

x = leaky\_relu(self.fc2(x))

a = a.view(-1, 1)

g = g.view(-1, 1)

print(x.size())

print(a.size())

print(g.size())

x = torch.cat((x, g, a), 1)

x = torch.cat((x, g, a), 1)

x = self.meta\_fc3(x)

x = leaky\_relu(x)

x = self.out(x)

return sigmoid(x)

1. changes: 4 additions & 5 deletions9

**Train.py**

from src.SkinCancer.Model import Net

from src.SkinCancer.Dataloader import TrainDataset

from torch import save

import torch

import numpy as np

from torch.tensor import Tensor

import torch.nn as nn

import pandas as pd

from torch.optim import Adam

from torch.utils.data import DataLoader, WeightedRandomSampler

from torch.utils.data import DataLoader

model = Net()

optimizer = Adam(model.parameters())

criterion = nn.BCELoss()

train\_dataset = TrainDataset()

ground\_truth = pd.read\_csv('/home/aftaab/MylanDatasets/Skin Cancer/ground\_truth.csv')

class\_weights = [sum(ground\_truth['melanoma']), 1-sum(ground\_truth['melanoma'])]

sampler = WeightedRandomSampler(class\_weights, len(class\_weights))

train\_loader = DataLoader(train\_dataset, batch\_size=32, shuffle=True)

train\_loader = DataLoader(train\_dataset, batch\_size=128, shuffle=True)

def train(epoch):

for step, data in enumerate(train\_loader):

train\_x, train\_y, sm, am = data

sex\_t=sm.float()

age\_t=am.float()

train\_y=train\_y.float().view(-1,1)

sex\_t = sm.float()

age\_t = am.float()

train\_y = train\_y.float().view(-1, 1)

optimizer.zero\_grad()

y\_pred = model.forward(train\_x, sex\_t, age\_t)

loss = criterion(y\_pred, train\_y)

loss.backward()

optimizer.step()

print('Epoch {}, batch={}, loss={}'.format(epoch,step, loss.item()))

print('Epoch {}, batch={}, loss={}'.format(epoch, step, loss.item()))

for i in range(1, 4):

for i in range(1, 41):

train(i)

save(model.state\_dict(), 'model\_skin\_cancer\_epoch{}.pt'.format(i))

elif train[index] in class2:

@@ -44,7 +44,7 @@ def \_\_getitem\_\_(self, index):

else:

y = 0

img = Tensor(img/255.0).view(3, 150, 150)

img = Tensor(img/255.0).view(1, 150, 150)

return img, y

def \_\_len\_\_(self):

@@ -58,9 +58,9 @@ def \_\_init\_\_(self):

self.test = test

def \_\_getitem\_\_(self, index):

img = Image.open(train[index]).convert('RGB')

img = Image.open(train[index]).convert('L')

img = img.resize([150, 150])

img = np.array(img).reshape(3, 150, 150)

img = np.array(img).reshape(1, 150, 150)

if test[index] in class1:

y = 1

if i % 5 == 0:

save(model.state\_dict(), 'model\_skin\_cancer\_epoch{}.pt'.format(i))

## REFERENCES

1. R. Suganya, An automated computer-aided diagnosis of skin lesions detection and classification for dermoscopy images. April 2021
2. Chiem, A. Al-Jumaily, and R. N. Khushaba, A Novel Hybrid System for Skin Lesion Detection. Dec 2021.
3. M. A. Farooq, M. A. M. Azhar, and R. H. Raza, Automatic Lesion Detection System (ALDS) for Skin Cancer Classification Using SVM and Neural Classifiers, Oct 2020.
4. M. W. Rashad and M. Takruri, Automatic non-invasive recognition of melanoma using Support Vector Machines, Dec 2020.
5. H. T. Lau and A. Al-Jumaily, Automatically Early Detection of Skin Cancer: Study Based on Neural Network Classification, Dec 2019.