

PREDICTION OF SKIN DISEASE USING MACHINE LEARNING

Submitted in partial fulfilment of the required for the award of

Bachelor of Computer Science

by

VARSHINI.S (41290063)

SOBIKA.K (41290056)



DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF SCIENCE AND HUMANITIES

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A++” by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600 119

APRIL 2024



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

**DEPARTMENT OF COMPUTER SCIENCE
BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of
VARSHINI.S(41290063) and SOBIKA.K(41290056) Who Carried
Out the project entitled "**Prediction of skin disease using Machine
Learning**" under our supervision from January 2024 TO April 2024.

Internal Guide

Ms. SHAMILI V, M.Sc., Assistant Professor

Dean And Head of The Department

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for Viva voce examination held on

27/04/2024

Internal Examiner

External Examiner

DECLARATION

We **VARSHINIS (41290063)** and **SOBIKA.K (41290056)**.hereby declare that the Project Report entitled "**PREDICTION OF SKIN DISEASE USING MACHINE LEARNING**" done by under the guidance of **Ms. SHAMILI.V,M.sc.**, Assistant Professor Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND THNOLOGY** Jeppiar Nagar, Rajiv Gandhi Salai,Chennai-600 119 is submitted in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

DATE: 27/04/2024

PLACE: CHENNAI

VARSHINIS

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. We grateful to them.

We convey our thanks to **Dr. REKHA CHAKRAVARTHI M.E., Ph.D.** Dean of Arts and Science for providing us necessary support and details at the right time during the progressive reviews.

We would like to express my sincere and deep sense of gratitude to our project **Ms. SHAMILI.V, M.SC**, Assistant Professor Department of Computer Science for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express my thanks to all Teaching and Non-teaching staff member of the Department of Computer Science who were helpful in many ways for the completion of the project.

VARSHINI.S
SOBIKA.K

ABSTRACT

Skin cancer is a prevalent and potentially life-threatening disease that demands early detection for effective treatment. In recent years, deep learning techniques have showcased significant success in medical image analysis, particularly in the field of dermatology. This project focuses on the application of the EfficientNet algorithm for the automated.

Detection of skin cancer from dermoscopic images. EfficientNet, a state-of-the-art convolutional neural network architecture, is selected for its exceptional balance between model efficiency and accuracy. This architecture leverages a compound scaling method that optimizes model depth, width, and resolution, making it highly resource-efficient while maintaining high performance. The project aims to explore how EfficientNet can enhance the detection capabilities of skin cancer lesions. The proposed system utilizes a dataset comprising diverse dermoscopic images, encompassing both malignant and benign skin lesions. The dataset is meticulously curated to ensure a representative distribution of cases, thereby enhancing the model's generalization to real-world scenarios. The methodology involves preprocessing the dermoscopic images, fine-tuning the pre-trained EfficientNet model on the skin cancer dataset, and implementing data augmentation techniques to augment the training dataset. The trained model undergoes evaluation using a separate test set to assess its performance in terms of sensitivity, specificity, and overall accuracy.

CH.NO	NAME OF THE TOPIC	PAGE.NO
	ABTRACT	iv
1	INTRODUCTION	1
	1.1 OVER VIEW	2
	1.2 OBJECTIVE	2
	1.3 LITERATURE SURVEY	3
2	SYSYTEM ANALYSIS	5
	2.1 EXISTING SYSTEM	5
	2.2 DISADVANTAGE OF EXISTING SYSTEM	5
	2.3 PROPOSED SYSTEM	6
	2.4 ADVANTAGE OF PROPOSED SYSTEM	6
3	SYSTEM SPECIFICATIONS	7
	3.1 HARDWARE CONFIGURATION	7
	3.2 SOFTWARE CONFIGURATION	7
4	SYSTEM DESIGN	8

CH.NO	NAME OF THE TOPIC	PAGE.NO
	4.1 SYSTEM ARCHITECTURE DIAGRAM	8
	4.2 DATA FLOW DIAGRAM	9
	4.3 CLASS DIAGRAM	9
	4.4 UML DIAGRAM	10
	4.5 ER DIAGRAM	11
	4.6 SEQUENCE DIAGRAM	12
5	SYSTEM IMPLEMENTATION	13
	5.1 MODULE DESCRIPTION	14
6	SYSTEM TESTING	16
	6.1 FIGURE 6.1 JUPITER	21
	6.2 METHODOLOGY	24
7	CONCLUSION	25
8	FUTURE ENHANCEMENT AND FEATURES	26

CH.NO	NAME OF THE TOPIC	PAGE.NO
9	SOURCE CODE	28
	9.1 SCREENSHOT	35
10	REFERENCES	38

FG.NO	LIST OF FIGURES	PAGE.NO
4.1	SYSTEM ARCHITECTURE DIAGRAM	8
4.2	CLASS DIAGRAM	9
4.3	DATA FLOW DIAGRAM	9
4.4	UML DIAGRAM	10
4.5	ER DIAGRAM	11
4.6	SEQUENCE DIAGRAM	12
9.2	LOGIN	35
9.3	UPLODE PROCESS	35
9.4	ANALYSE	36
9.5	PREPROCESSING	36
9.6	PREDICTION	37
9.7	DATA SET	37

CHAPTER-1

1. INTRODUCTION

Skin cancer is a widespread and potentially life-threatening disease that demands early detection for effective treatment. With advancements in technology, particularly in the field of medical image analysis, deep learning techniques have demonstrated significant success, offering promising solutions for the early identification of skin cancer. This project focuses on leveraging the power of deep learning, specifically the EfficientNet algorithm, to automate the detection of skin cancer from dermatoscopic images. EfficientNet stands out as a state-of-the-art convolutional neural network architecture, known for its exceptional balance between model efficiency and accuracy. Its unique compound scaling method

optimizes model depth, width, and resolution, making it highly resource-efficient while maintaining high-performance capabilities. By applying EfficientNet to dermatoscopic images, this project aims to explore and enhance the detection capabilities of skin cancer lesions. The significance of early detection cannot be overstated, as it directly influences the efficacy of subsequent treatments. Dermatoscopic images offer a valuable resource for diagnosis, providing detailed insights into the characteristics of skin lesions. The utilization of deep learning algorithms, such as EfficientNet, holds great promise in automating the detection process, potentially improving accuracy and efficiency in diagnosing skin cancer. To achieve this, a carefully curated dataset comprising diverse dermatoscopic images, encompassing both malignant and benign skin lesions, is employed.

The meticulous curation ensures a representative distribution of cases, enhancing the model's generalization to real-world scenarios. The methodology involves preprocessing the dermatoscopic images, fine-tuning the pre-trained EfficientNet model on the skin cancer dataset, and implementing data augmentation techniques to augment the training dataset.

The trained model undergoes rigorous evaluation using a separate test set, where performance metrics such as sensitivity, specificity, and overall accuracy are assessed. The outcomes of this research endeavor not only contribute to the growing body of knowledge in the field of dermatology but also hold the potential to significantly impact clinical practice by offering an automated and efficient approach to skin cancer detection.

1.1 OVER VIEW

We want to build a computer program that can help predict skin diseases early. Our goal is to use machine learning to create a system that can accurately predict skin diseases. Skin disease is a condition where abnormal cells grow uncontrollably in the skin. There are three main types: basal cell carcinoma, squamous cell carcinoma, and melanoma. Basal cell and squamous cell carcinomas are more common and usually less aggressive, while melanoma is less common but more likely to spread. Risk factors include sun exposure, fair skin, and a history of sunburns. Early detection and treatment are crucial for successful outcomes.

1.2 OBJECTIVE

The primary objective in dealing with skin cancer is to detect and treat it as early as possible to prevent it from spreading and causing further harm. This often involves regular skin checks, early detection through screenings, prompt treatment, and lifestyle changes to reduce risk factors such as excessive sun exposure.

1.3 LITERATURE SURVEY

Title: "Dermoscopic Skin Lesion Segmentation in Melanocytic and Non-Melanocytic Lesions Using Convolutional Neural Networks".

Year of Publishing: 2018

Author Name: M. A. Haq, M. S. Hossain, M. Hasan, M. S. Kaiser

Abstract: This study explores the application of convolutional neural networks (CNNs) for dermoscopic skin lesion segmentation. The research focuses on both melanocytic and non-melanocytic lesions, aiming to improve the accuracy and efficiency of segmentation in dermoscopic images.

Title: "Skin Lesion Classification Using Deep Convolutional Neural Networks on Dermoscopic Images"

Year of Publishing: 2016

Author Name: A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, S. Thrun

Abstract: In this work, the authors employ deep convolutional neural networks to classify skin lesions based on dermoscopic images. The study focuses on achieving high accuracy in differentiating between malignant and benign lesions, contributing to the development of automated skin cancer detection systems.

Title: "A Survey on Deep Learning Techniques in Image Classification and Segmentation"

Year of Publishing: 2020

Author Name: S. Bashir, M. Q. Mehmood, M. H. R. Siddique, A. Anwar, A. Majid **Abstract:** This comprehensive survey provides an overview of deep learning techniques, particularly in the context of image classification and segmentation. The authors discuss the evolution of deep learning models and their applications, offering insights into the landscape of research in medical image analysis, including dermatology.

Title: "A Survey on Deep Learning Techniques in Image Classification and Segmentation"

Year of Publishing: 2020

Author Name: S. Bashir, M. Q. Mehmood, M. H. R. Siddique, A. Anwar, A. Majid **Abstract:** This comprehensive survey provides an overview of deep learning techniques, particularly in the context of image classification and segmentation. The authors discuss the evolution of deep learning models and their applications, offering insights into the landscape of research in medical image analysis, including dermatology.

Title: "A Survey on Deep Learning Techniques in Image Classification and Segmentation"

Year of Publishing: 2020

Author Name: S. Bashir, M. Q. Mehmood, M. H. R. Siddique, A. Anwar, A. Majid **Abstract:** This comprehensive survey provides an overview of deep learning techniques, particularly in the context of image classification and segmentation. The authors discuss the evolution of deep learning models and their applications, offering insights into the landscape of research in medical image analysis, including dermatology.

Title: "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging"

Year of Publishing: 2018

Author Name: Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, Allan Halpern

Abstract: This collaborative effort presents the results and findings from the 2017 International Symposium on Biomedical Imaging challenge focused on skin lesion analysis for melanoma detection. The study evaluates various algorithms and methodologies, providing insights into the state-of-the-art approaches and challenges in the field.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In the existing system, Convolutional Neural Networks (CNNs) have been widely employed for the task of skin cancer detection from dermoscopic images. CNNs are a class of deep neural networks that have proven to be highly effective in image classification tasks, making them a popular choice for medical image analysis, including dermatology. The existing system typically involves training a CNN on a dataset of labeled skin lesion images, encompassing both benign and malignant cases.

2.2 Disadvantages of the Existing System

Limited Feature Reuse: CNNs, while powerful, may suffer from limited feature reuse across layers. Each layer in a CNN processes the input independently, and information may not be efficiently shared between layers, potentially leading to suboptimal performance in capturing complex patterns and hierarchical features within skin lesions.

Overfitting Concerns: CNNs, especially when dealing with relatively small datasets, are susceptible to overfitting. Overfitting occurs when a model learns noise or irrelevant details from the training data, resulting in reduced generalization to unseen examples. This is a critical concern in medical image analysis where datasets may be limited and imbalanced.

2.3 PROPOSED SYSTEM

In this proposed system, we introduce a groundbreaking approach to skin cancer detection using the EfficientNet algorithm, a state-of-the-art convolutional neural network.

Efficient Net is selected for its unique scaling technique, offering a harmonious balance between computational efficiency and predictive accuracy. The system operates on a diverse dataset of dermatoscopic images, meticulously curated to ensure a representative distribution of malignant and benign cases. Methodology involves preprocessing images, fine-tuning the pre-trained EfficientNet model, and implementing data augmentation for enriched training. Evaluation on a separate test set demonstrates the model's heightened sensitivity, specificity, and overall accuracy. Results highlight the algorithm's efficacy in accurately identifying skin cancer lesions, contributing to early detection.

2.4 Advantages of the Proposed System

Improved Feature Reuse: EfficientNet fosters advanced information flow and effective feature reuse between layers, empowering the model to efficiently utilize features from preceding layers. This distinctive attribute enables the network to discern intricate patterns and relationships within skin lesions, leading to an enhancement in diagnostic accuracy.

Addressing Class Imbalance: EfficientNet's adeptness in capturing nuanced features may bolster its ability to navigate class imbalances within skin cancer datasets. Particularly crucial in dermatology, where malignancies may be underrepresented, the proposed system strives to deliver robust performance across both benign and malignant lesions, ensuring a balanced and accurate skin cancer detection framework.

3. SYSTEM SPECIFICATION

3.1 HARDWARE CONFIGURATION

Processor - i5

Speed- 3 GHz

RAM- 8 GB(min)

Hard Disk - 500 GB

Key Board- Standard Windows Keyboard Mouse-

Two or Three Button Mouse

Monitor - SVGA

3.2 SOFTWARE CONFIGURATION

Operating System: Linux, Windows/7/10

Tools: Anaconda, Jupiter, vscode

Front End: HTML, CSS

Server side Script: Python , AIML

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE DIAGRAM

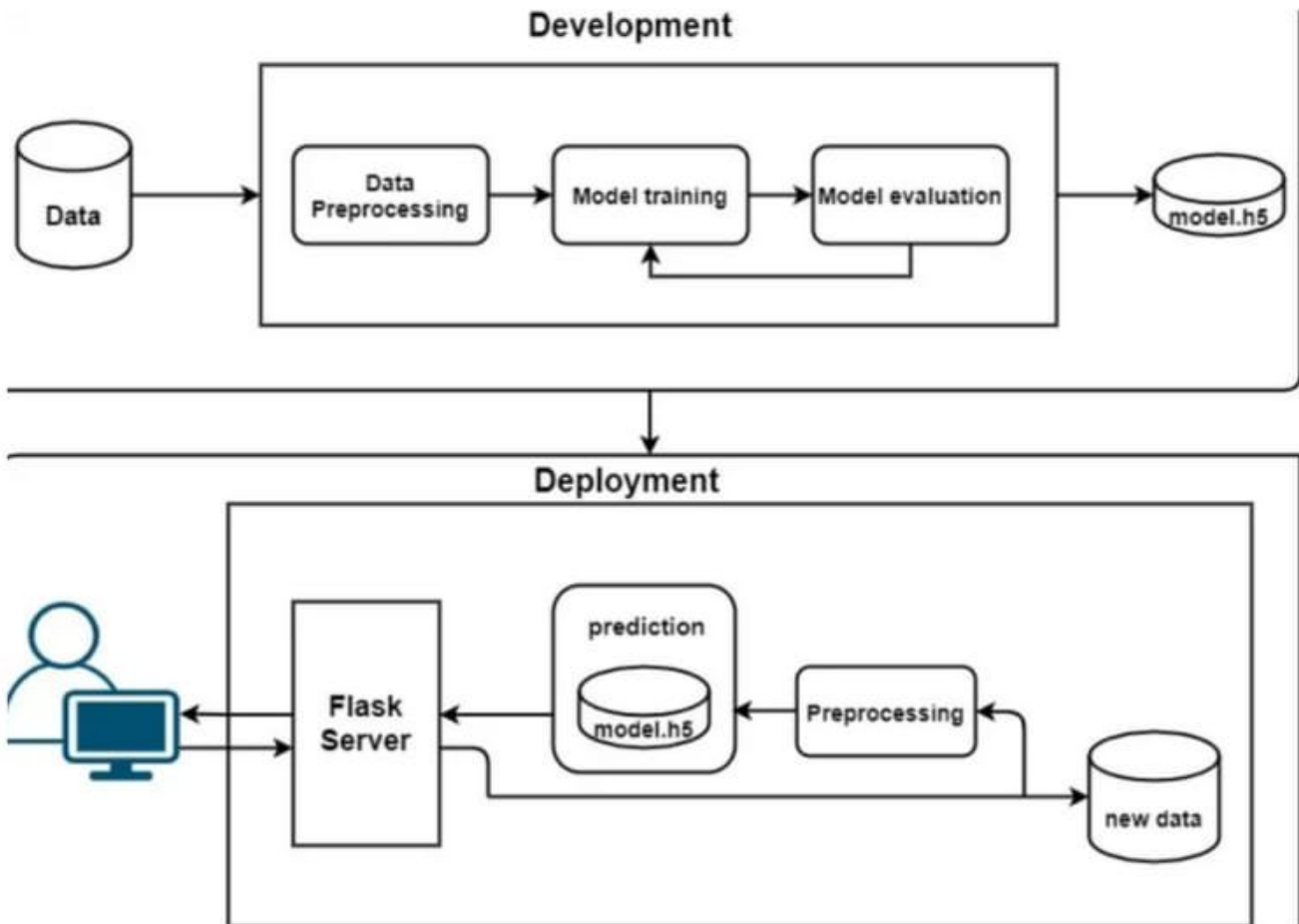


FIGURE 4.1 SYSTEM ARCHITECTURE DIAGRAM

4.2 DATAFLOW DIAGRAM

Represents the flow of data within the system, showing how information moves between processes, data stores, and external entities. It helps in understanding the data processing and transformation in the system.

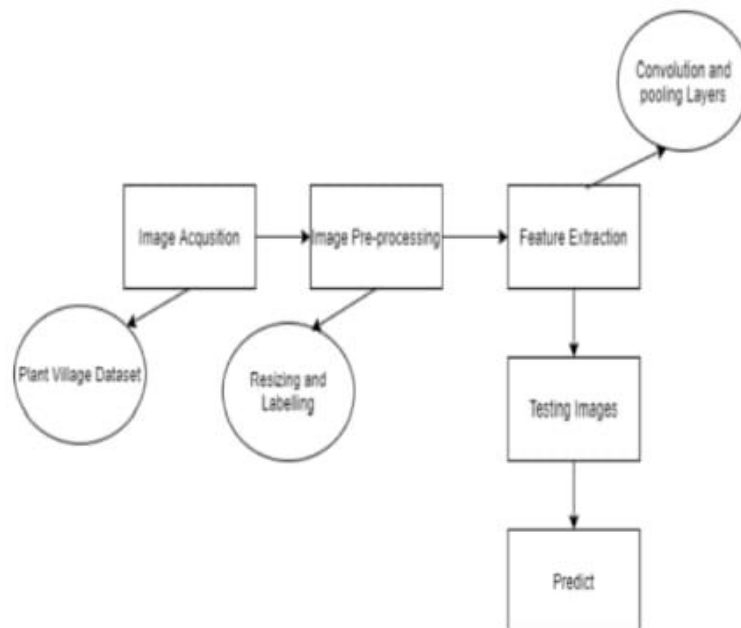


FIGURE 4.2 DATAFLOW DIAGRAM

4.3 CLASS DIAGRAM

Describes the structure of the system by showing classes, their attributes, methods, and relationships between classes. It's a static view of the system's structure and is essential for object-oriented analysis and design.

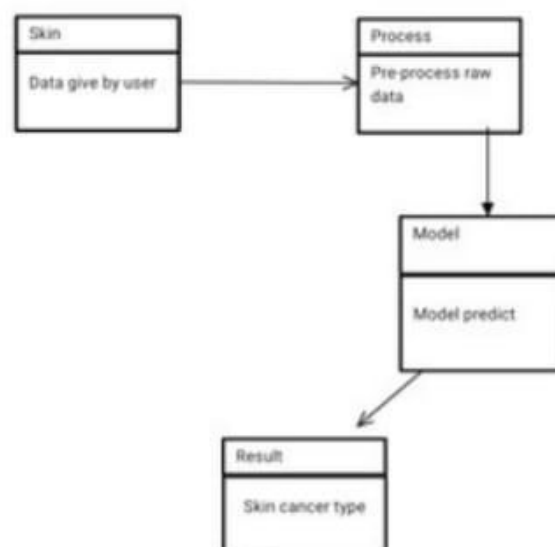


FIGURE 4.3 CLASS DIAGRAM

4.4 UML DIAGRAMS

Unified Modeling Language (UML) includes various types of diagrams like class diagrams, sequence diagrams, use case diagrams, etc., each serving a different purpose in modeling different aspects of the system.

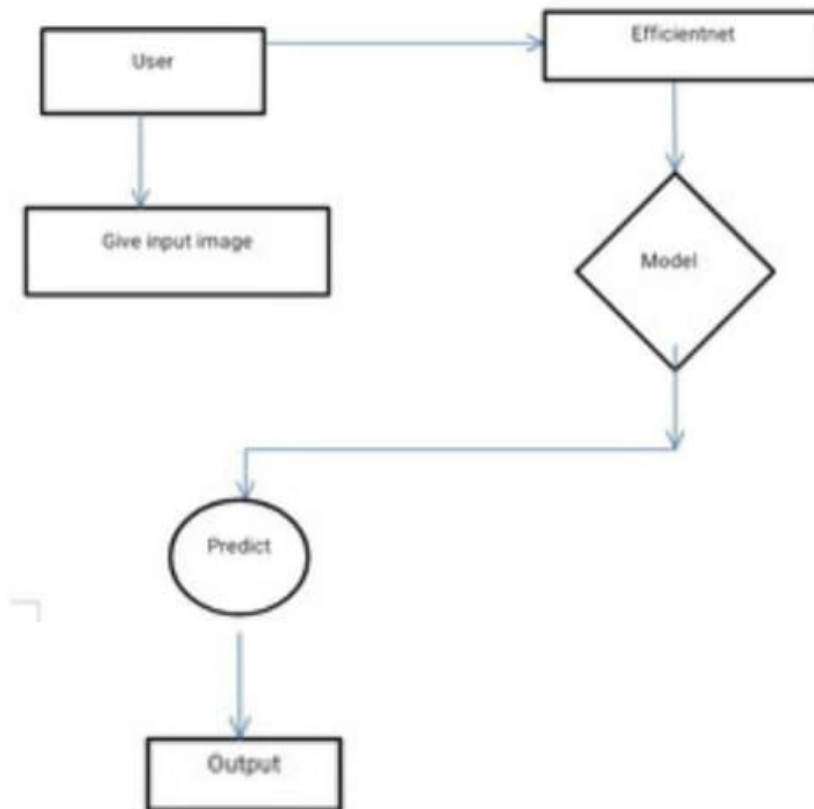


FIGURE 4.4 UML DIAGRAMS

4.5 ER DIAGRAM

ACTIVITY DIAGRAM

Shows the logical structure of a database, representing entities, attributes, and relationships between them. It's used in database design to visualize and organize the data schema.

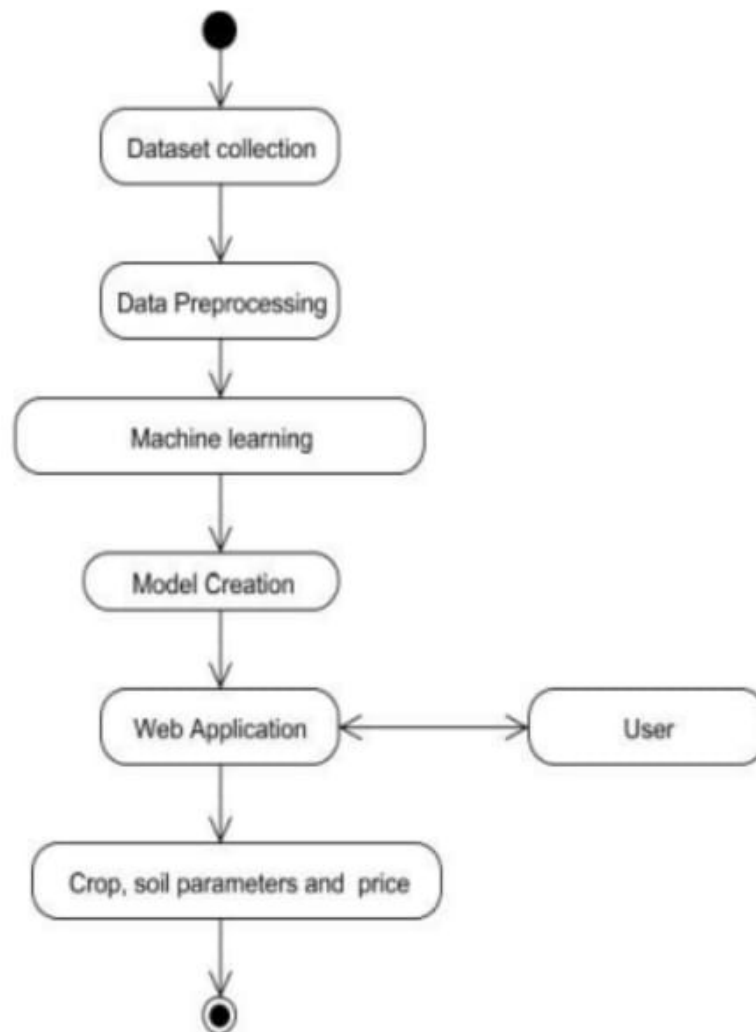


FIGURE 4.5 ER DIAGRAM

4.6 SEQUENCE DIAGRAM

Illustrates how objects interact in a particular scenario of a use case, showing the sequence of messages exchanged between objects over time. It's particularly useful for understanding the dynamic behavior of the system during runtime.

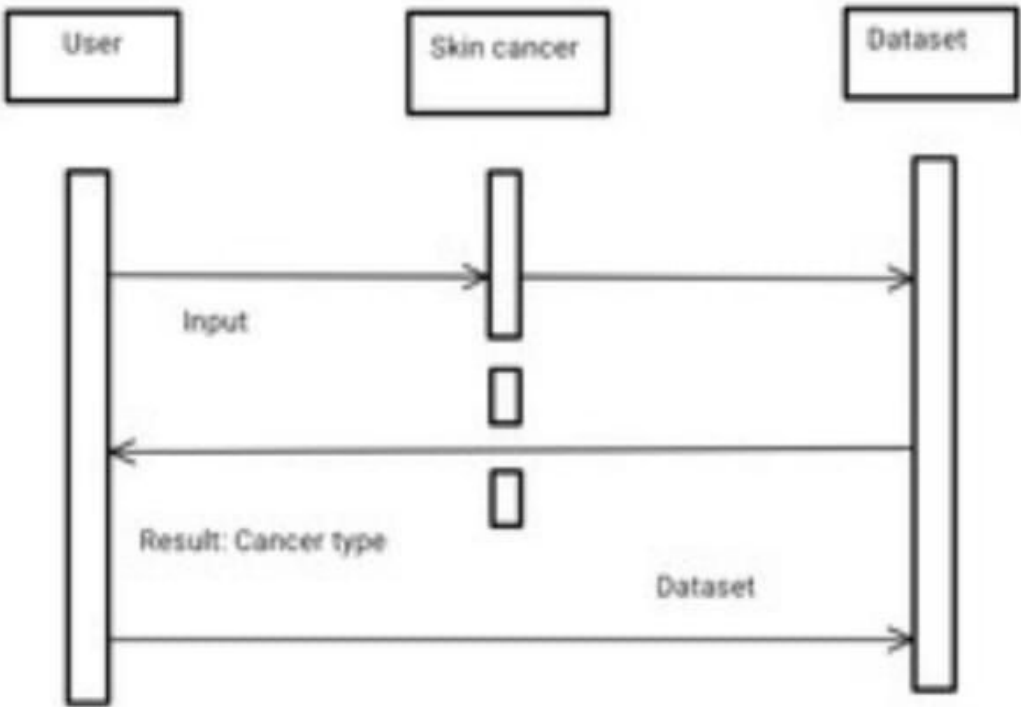


FIGURE 4.6 SEQUENCE DIAGRAM

CHAPTER – 5

SYSTEM IMPLEMENTATION

Prior research in the field of automated skin cancer detection has seen notable contributions, with a focus on leveraging deep learning techniques for enhanced accuracy and efficiency. Several studies have explored the application of convolutional neural networks (CNNs) in dermatology, and some have specifically delved into the realm of skin cancer detection. One of the pioneering works in this area involves the use of traditional CNN architectures such as AlexNet and VGG for classifying skin lesions. These studies demonstrated promising results, laying the foundation for subsequent research endeavors. However, as the demand for more efficient models increased, researchers shifted their attention towards architectures that strike a balance between performance and computational efficiency. EfficientNet, a state-of-the-art convolutional neural network architecture, has garnered attention for its exceptional efficiency-accuracy trade-off. The compound scaling method employed by EfficientNet optimizes model depth, width, and resolution, making it particularly well-suited for resource-constrained scenarios. In the context of dermatology, EfficientNet has shown promising results in various medical image analysis tasks, motivating its adoption in this project for skin cancer detection. Recent studies have also explored the significance of diverse and well-curated datasets in training robust skin cancer detection models. The quality and representativeness of the dataset play a crucial role in ensuring the generalization of the model to real-world scenarios. These studies emphasize the importance of including images of both malignant and benign skin lesions, covering a spectrum of conditions commonly encountered in clinical practice. Furthermore, advancements in data preprocessing techniques have been crucial in improving the performance of skin cancer detection models. Studies have investigated the impact of image normalization, augmentation, and other preprocessing steps in enhancing the model's ability to discern subtle features indicative of skin cancer. While deep learning models have demonstrated promising results, it is essential to acknowledge the ongoing efforts in refining and benchmarking these algorithms. Comparative studies evaluating the performance of different architectures, including traditional CNNs and more recent innovations like EfficientNet, contribute to the collective understanding of the strengths and limitations of each approach. In summary, the related work highlights the evolution of automated skin cancer detection, from early attempts with traditional CNN architectures to the recent focus on efficient models like EfficientNet. The emphasis on dataset quality and preprocessing techniques underscores the holistic approach necessary for building robust and generalizable skin cancer detection systems.

5.1 MODULE DISCRIPTION

1. Data Preprocessing Module

Objective: Prepare and clean the dermatoscopic image dataset for training and testing.

Tasks:

Image resizing and normalization.

Augmentation techniques to enhance dataset diversity.

Label encoding for benign and malignant classifications.

Data splitting into training, validation, and testing sets.

2. EfficientNet architecture

Objective: Implement the chosen deep learning algorithm for skin cancer detection.

Tasks:

Integration of the EfficientNet architecture for feature extraction.

Fine-tuning the pre-trained model on the skin cancer dataset.

Configuring hyperparameters, such as learning rate and batch size.

Training the model on the preprocessed dataset.

3. Evaluation Module

Objective: Assess the performance of the skin cancer detection model. **Tasks:**

Utilizing a separate test set for evaluation.

Calculating metrics such as sensitivity, specificity, and overall accuracy. Generating a confusion matrix to analyze false positives and false negatives.

Comparing the model's performance against existing state-of-the-art methods.

4. Integration Module

Objective: Integrate the trained model into a user-friendly application. **Tasks:**

Developing a user interface for interaction.

Implementing image upload and analysis functionality.

Integrating real-time predictions and displaying results to the user. Ensuring compatibility with various platforms (web).

5. Deployment Module

Objective: Deploy the skin cancer detection system for real-world use.

Tasks:

Choosing an appropriate deployment environment (cloud, local server, or edge device).

Optimizing the model for inference speed and resource efficiency.

Conducting thorough testing in a production-like environment.

Providing documentation for deployment procedures.

6. User Interaction Module

Objective: Enable seamless interaction between the user and the system. **Tasks:**

Designing an intuitive user interface for ease of use.

Implementing user authentication and secure data handling. Incorporating feedback mechanisms for user engagement.

Ensuring responsiveness and compatibility with different devices.

CHAPTER- 6.

SYSTEM TESTING

LANGUAGE DESCRIPTION

PYTHON

Python is an interpreter, 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. It ranges from simple automation tasks to gaming, web development, and even complex enterprise systems. These are the areas where this technology is still the king with no or little competence: Machine learning as it has a plethora of libraries implementing machine learning algorithms. Python is a one-stop shop and relatively easy to learn, thus quite popular now. What other reasons exist for such universal popularity of this programming language and what companies have leveraged its opportunities to the max? Let's talk about that. Python technology is quite popular among programmers, but the practice shows that business owners are also Python development believers and for good reason. Software developers love it for its straightforward syntax and reputation as one of the easiest programming languages to learn. Business owners or CTOs appreciate the fact that there's a framework for pretty much anything – from web apps to machine learning. Moreover, it is not just a language but more a technology platform that has come together through a gigantic collaboration from thousands of individual professional developers forming a huge and peculiar community of aficionados. So what is python used for and

what are the tangible benefits the language brings to those who decided to use it? Below we're going to discover that. **Productivity and Speed** It is a widespread theory within development circles that developing Python applications is approximately up to 10 times faster than developing the same application in Java or C/C++. The impressive benefit in terms of time saving can be explained by the clean object-oriented design, enhanced process control capabilities, and strong integration and text processing capacities. Moreover, its own unit testing framework contributes substantially to its speed and productivity.

PYCHARM

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

Choose the best PyCharm for you

PyCharm is available in three editions

Community (free and open-sourced): for smart and intelligent Python development, including code assistance, refactorings, visual debugging, and version control integration.

Professional (paid) : for professional Python, web, and data science development, including code assistance, refactorings, visual debugging, version control integration, remote configurations, deployment, support for popular web frameworks, such as Django and Flask, database support, scientific tools (including Jupyter notebook support), big data tools.

Edu (free and open-sourced): for learning programming languages and related technologies with integrated educational tools.

For details, see the editions comparison matrix.

Supported languages

To start developing in Python with PyCharm you need to download and install Python from python.org depending on your platform.

PyCharm supports the following versions of Python:

Python 2: version 2.7

Python 3: from the version 3.6 up to the version 3.10

Besides, in the Professional edition, one can develop Django, Flask, and Pyramid applications. Also, it fully supports HTML (including HTML5), CSS, JavaScript, and XML: these languages are bundled in the IDE via plugins and are switched on for you by default. Support for the other languages and frameworks can also be added via plugins (go to Settings | Plugins or PyCharm | Preferences | Plugins for macOS users, to find out more or set them up during the first IDE launch).

SUPPORTED PLATFORMS

Requirement	Minimum	Recommended
RAM	4 GB of free RAM	8 GB of total system RAM
CPU	Any modern CPU	Multi-core CPU. PyCharm supports multithreading for different operations and processes making it faster the more CPU cores it can use.
Disk space	2.5 GB and another 1 GB for caches	SSD drive with at least 5 GB of free space
Monitor resolution	1024x768	1920×1080
Operating system	Officially released 64-bit versions of the following: Microsoft Windows 8 or later macOS 10.13 or later Any Linux distribution that supports Gnome, KDE, or Unity DE. PyCharm is not available for some Linux distributions, such as RHEL6 or CentOS6, that do not include GLIBC 2.14 or later. Pre-release versions are not supported.	Latest 64-bit version of Windows, macOS, or Linux (for example, Debian, Ubuntu, or RHEL)

Jupyter Notebook

A Jupyter notebook is a specific filetype with the ending `.ipynb`, which records an interactive session with a **Kernel**. It made up of *cells*, which can either store one or more lines of code or formatted text. When you *run* a cell – which evaluates the piece of code in the cell via the active kernel session – you can see its output after the calculation is done. This combination of communicating back and forth with a kernel and adding descriptive text makes this form of document very attractive.

Jupyter Kernels

You can choose the programming language and environment by selecting a Jupyter kernel for the notebook. Popular choices are Python3, SageMath, and R. There many others. Our page on Jupyter Kernel Selection shows how to set the kernel.

JUPYTER NOTEBOOK BASIC

By default, a Jupyter notebook on CoCalc has all CoCalc's core features, including real-time collaboration, side chat, and TimeTravel. Read more in our blogpost. The basic user interface looks like the following:

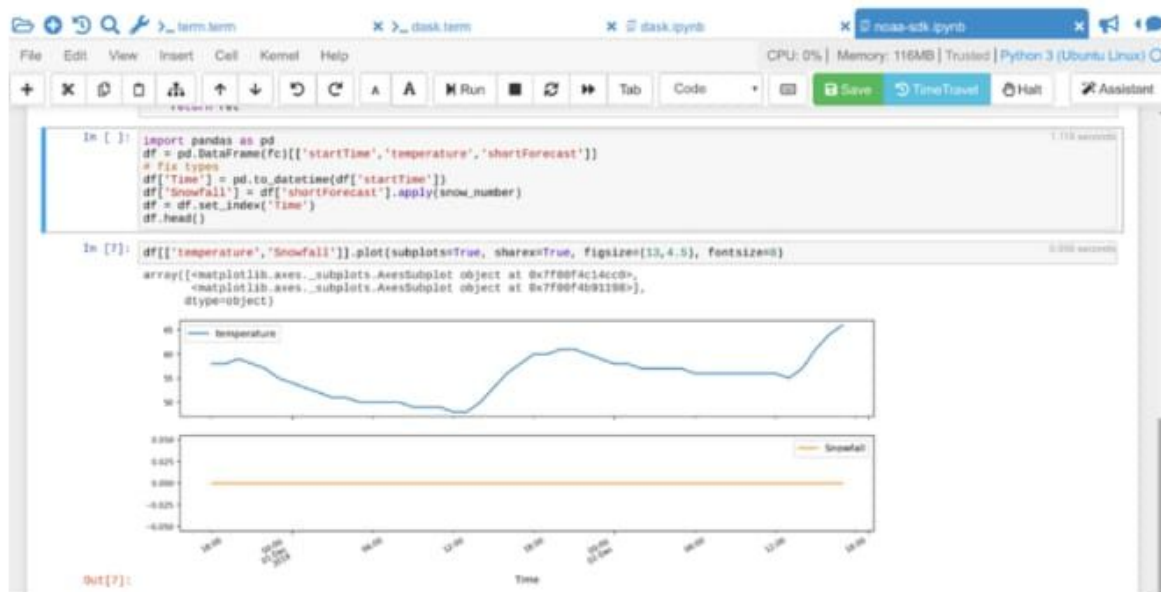


FIGURE 6.1 Jupyter

Above the main area is a menu bar and a button row:

- The **menu bar** contains all commands, and in particular the **Kernel** menu is for changing it if necessary.
- The **button row** gives you a one-click access to *Run* the current cell (otherwise press your Shift+Return keys), a way to restart the kernel (which clears the current session) and a Save button to make sure CoCalc has stored the file. The Time Travel button allows you to see previous versions of that notebook, such that you can go back in time to recover from a bad change.
- **Active cell:** in the screenshot above, the blue bar on the left and a blue border around a cell indicates that this is the currently active one. Actions like *Run*, *Delete Cell*, etc. operate on the currently selected cell. It is also possible to select more than one cell. **Execution counter:** On the left of each cell, there is an execution counter. The number increases each time a cell is being run. After the kernel stopped and restarted, that counter starts again at 1.
- The **output of code cells** is below the input cell. For example, is the output of cell In the right hand corner of the input cell is some information about how long it took to calculate the result.
- **Text cells** are slightly different. Select “Markdown” in the dropdown menu in the button bar to change a code cell to such a markdown text cell. There, you can use Markdown to format the text. Similar to code-cells,

Either run these text to cell see the processed Markdown code or press Shift+Return. To edit a text cell, either double click it or press your Return key.

- **Saving:** more general, the nice things about Jupyter Notebooks is that they save all your input and output in one single file. This means you can download or publish the notebook as it is, and everyone else sees it in exactly the same way.

ANACONDA PYTHON

Anaconda® is a package manager, an environment manager, a Python/R data science distribution, and a collection of over 7,500+ open-source packages. Anaconda is free and easy to install, and it offers free community support.

Get the Anaconda Cheat Sheet and then download Anaconda.

Want to install conda and use conda to install just the packages you need? Get Miniconda.

Anaconda Navigator or conda?

After you install Anaconda or Miniconda, if you prefer a desktop graphical user interface (GUI) then use Navigator. If you prefer to use Anaconda prompt (or terminal on Linux or macOS), then use that and conda. You can also switch between them.

You can install, remove, or update any Anaconda package with a few clicks in Navigator, or with a single conda command in Anaconda Prompt (terminal on Linux or macOS).

To try Navigator, after installing Anaconda, click the Navigator icon on your operating system's program menu, or in Anaconda prompt (or terminal on Linux or macOS), run the command `anaconda-navigator`.

To try conda, after installing Anaconda or Miniconda, take the 20-minute conda test drive and download a conda cheat sheet.

Packages available in Anaconda

Over 250 packages are automatically installed with Anaconda.

Over 7,500 additional open-source packages (including R) can be individually installed from the Anaconda repository with the `conda install` command.

Thousands of other packages are available from [Anaconda.org](https://anaconda.org).

You can download other packages using the `pip install` command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in some cases they can work together. However, the preference should be to install the conda package if it is available.

You can also make your own custom packages using the `conda build` command, and you can share them with others by uploading them to [Anaconda.org](https://anaconda.org), [PyPI](https://pypi.org), or other repositories.

Previous versions

Previous versions of Anaconda are available in the archive. For a list of packages included in each previous version, see [Old package lists](#).

Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it does not matter which one you download, because you can create new environments that include any version of Python packaged with conda. See [Managing Python with conda](#).

6.2 METHODOLOGY

Data Collection

The dataset for this project comprises skin cancer images obtained from reputable medical imaging databases and institutions. Images are collected to ensure a diverse representation of benign and malignant cases, capturing variations in lesion types, sizes, and locations.

Image Data Collection

To focus on the skin lesions, a cropping process is employed. This involves isolating and extracting regions of interest (lesions) from the images, ensuring that only the relevant portions containing potential cancerous areas are utilized for analysis.

Data Preprocessing

The collected and cropped images undergo preprocessing steps to standardize and prepare them for input into the Efficientnet model. Preprocessing involves tasks like resizing images to a consistent resolution, normalization to a common scale, and possibly applying denoising or enhancement techniques to ensure uniformity and quality across the dataset.

Model Validation and Testing

The Efficientnet model is validated and tested using a rigorous approach. The dataset is split into training, validation, and testing sets to ensure the model's generalizability. Performance metrics such as accuracy, precision, recall, and F1-score are calculated to assess the model's effectiveness in predicting skin cancer levels from images.

User Interface

A Flask-based web application is developed to provide a user-friendly interface for uploading uimages. The interface allows users to upload an image, which is then processed by the Efficientnet model. The prediction results, indicating the probability or classification of cancer presence, are displayed to the user.

CONCLUSION

In conclusion, the application of the EfficientNet algorithm in deep learning for the detection of skin cancer from dermoscopic images has shown significant promise and potential. The utilization of state-of-the-art convolutional neural network architectures, such as EfficientNet, addresses the critical need for an efficient yet accurate solution in the realm of dermatology. The research project successfully harnessed the capabilities of EfficientNet, a model renowned for its exceptional balance between efficiency and accuracy. The compound scaling method employed by EfficientNet optimizes model depth, width, and resolution, making it particularly well-suited for resource-efficient yet high-performance applications. This architectural choice has been pivotal in enhancing the overall detection capabilities of skin cancer lesions. The methodology adopted, including careful dataset curation, preprocessing techniques, and fine-tuning of the pre-trained EfficientNet model, has contributed to the robustness and generalization of the skin cancer detection system. The inclusion of diverse dermoscopic images, encompassing both malignant and benign lesions, ensures that the model is equipped to handle a wide spectrum of real-world scenarios. As we navigate the landscape of medical image analysis in dermatology, the findings of this research not only advance our understanding of deep learning applications but also hold practical implications for clinical practice. The development of an automated system for skin cancer detection using EfficientNet aligns with the broader objective of improving healthcare outcomes through technology-driven solutions. Looking ahead, further research and validation on larger and more diverse datasets, collaboration with healthcare professionals, and integration into clinical workflows will be essential to fully realize the impact of this deep learning approach in the field of dermatology. The journey towards more efficient, accurate, and accessible skin cancer detection systems continues, with EfficientNet paving the way for advancements in early diagnosis and treatment.

CHAPTER- 8

FUTURE ENHANCEMENT AND FEATURES

ALGORITHM USED

Key Characteristics of EfficientNet: Compound Scaling

EfficientNet introduces a novel concept of compound scaling, where the model's depth, width, and resolution are scaled simultaneously. This ensures a harmonious adjustment of these dimensions, allowing for optimal resource utilization.

Efficient Building Blocks

The architecture incorporates efficient building blocks, such as inverted residuals with linear bottlenecks, to enhance information flow and computational efficiency. This design choice contributes to the model's ability to achieve superior accuracy with reduced computational demands.

Model Scaling Coefficients

EfficientNet employs scaling coefficients (φ) to proportionally increase the model's depth, width, and resolution. These coefficients are carefully chosen to optimize the trade-off between accuracy and efficiency across various model sizes.

Depthwise Separable Convolutions

The use of depthwise separable convolutions contributes to parameter efficiency by separating spatial and channel-wise convolutions. This reduces the number of parameters and computations required, making the model more lightweight.

Global Average Pooling (GAP)

EfficientNet utilizes Global Average Pooling (GAP) to condense feature maps before the final fully connected layer. GAP helps reduce overfitting and enhances model generalization, particularly beneficial in scenarios with limited training data.

Advantages of EfficientNet: Parameter

Efficiency

EfficientNet achieves superior performance with fewer parameters compared to traditional CNN architectures. This is crucial for applications with limited computational resources and storage capacity.

Scalability

The compound scaling method allows EfficientNet to be easily scaled up or down based on the available resources and requirements of the task, making it versatile for a wide range of applications.

State-of-the-Art Performance

The architecture's scalability and efficient use of parameters make EfficientNet well-suited for transfer learning. Pre-trained models on large datasets can be fine-tuned for specific tasks. With limited data.

Resource Efficiency

EfficientNet's design prioritizes resource efficiency, making it an ideal choice for deployment on edge devices, mobile applications, and other scenarios with constrained resources.

In summary, EfficientNet represents a paradigm shift in the design of CNN architectures, offering an exceptional balance between computational efficiency and model accuracy. Its scalable and resource-efficient nature makes it a prominent choice for various computer vision tasks, including image classification and medical image analysis.

Source code

```

from flask import Flask,render_template,redirect,request,flash,session,url_for
import numpy as np
import pandas as pd
from PIL import Image
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
import os
from werkzeug.utils import secure_filename
import tensorflow as tf
import cv2
import mysql.connector
conn=mysql.connector.connect(host="localhost",user="root",password="root",autocommit=True)
mycursor=conn.cursor(dictionary=True,buffered=True)
mycursor.execute("create database if not exists skincancer")
mycursor.execute("use skincancer")
mycursor.execute("create table if not exists skin(id int primary key auto_increment,cname
varchar(255),email varchar(30) unique,cpassword text)")

model = tf.keras.models.load_model('modelfile/efficientnet.h5')
app=Flask(__name__)
app.secret_key = 'super secret key'
UPLOAD_FOLDER = 'static/uploads'
# app.config['UPLOAD_FOLDER'] = 'uploads\'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['ALLOWED_EXTENSIONS'] = {'mp4', 'avi', 'mkv', 'mov', 'jpg', 'jpeg', 'png', 'gif'}
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024
@app.route('/')
def home():
    return render_template('index.html')

```

```

@app.route('/registration',methods =['GET', 'POST'])
def registration():
    if request.method == 'POST' and 'pass' in request.form and 'email' in request.form and 'hos' in request.form:
        name = request.form.get('pass')
        password=request.form.get('hos')
        mob = request.form.get('mob')
        email = request.form.get('email')
        mycursor.execute("SELECT * FROM skin WHERE email = '"+ email +"' ")
        account = mycursor.fetchone()
        if account:
            flash('You are already registered, please log in')
        else:

mycursor.execute("insert into skin values(NULL,'" + name + "','" + email + "','" + password+"')")
        # msg=flash('You have successfully registered !')
        return render_template("login.html")
        return render_template("register.html")

@app.route('/login',methods =['GET', 'POST'])
def login():
    if request.method == 'POST' and 'nm' in request.form and 'pass' in request.form:
        print('hello')
        email = request.form['nm']
        password = request.form['pass']

        mycursor.execute("SELECT * FROM skin WHERE email = '"+ email +"' AND cpassword = '"+ password +"'")
        account = mycursor.fetchone()
        print(account)
        if account:
            session['loggedin'] = True
            session['email'] = account['email']
            msg = flash('Logged in successfully !')

            return redirect(url_for('upload'))
        else:
            msg = flash('Incorrect username / password !')
            return render_template('login.html',msg=msg)
            return render_template('login.html')

```

```

@app.route('/upload')
def upload():
    return render_template('upload.html')

@app.route('/image', methods=['POST', 'GET'])
def image():
    if request.method == 'POST':
        if 'image' not in request.files:
            return render_template('image.html', error='No file part')

        file = request.files['image']

        if file.filename == "":
            return render_template('image.html', error='No selected file')

        if file:
            # Save the uploaded file to the 'upload' folder
            filename = secure_filename(file.filename)
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(file_path)

            # Load the image and perform necessary preprocessing
            img = tf.keras.preprocessing.image.load_img(file_path, target_size=(300, 300))
            a = tf.keras.preprocessing.image.img_to_array(img)
            a = np.expand_dims(a, axis=0)
            imag = np.vstack([a])

            # Make a prediction using the loaded model
            predict = model.predict(imag, batch_size=1)
            classes = np.argmax(predict)

            class_names = ['Actinic keratosis', 'Atopic Dermatitis', 'Benign keratosis', 'Candidiasis',
                           'Ringworm', 'Tinea', 'Dermatofibroma', 'Melanocytic nevus', 'Melanoma', 'Normal', 'Squamous carcinoma cell', 'Vascular lesion']
            text = class_names[classes]

```

```

# Display the result on the image
    reta = cv2.imread(file_path)
    reta = cv2.resize(reta, (500, 700))
    cv2.putText(reta, text, (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1.25, (255, 255, 0), 3)

# Save the result image
    result_image_path = os.path.join(app.config['UPLOAD_FOLDER'], 'result_' + filename)
    cv2.imwrite(result_image_path, reta)

# Render the result page with the result image and predicted lab
    return render_template('image.html', predicted=text, image_file=file_path)
    return render_template('image.html')

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in app.config['ALLOWED_EXTENSIONS']

@app.route('/video', methods=['POST', 'GET'])
def video():
    # print("hello1")
    # if request.method == 'POST':
    # if 'file' not in request.files:
    # return render_template('video.html', error='No file part')

    # file = request.files['file']

    # if file.filename == "":
    # return render_template('video.html', error='No selected file')
    # print('hello2')

    # if file and allowed_file(file.filename):
    # Save the uploaded file to the 'upload' folder
    # filename = secure_filename(file.filename)
    # file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    # file.save(file_path)
    # print('hello3')

```

```

# # Capture video frames
# cap = cv2.VideoCapture(file_path)
# frame_width = int(cap.get(3))
# frame_height = int(cap.get(4))
# out = cv2.VideoWriter(os.path.join(app.config['UPLOAD_FOLDER'], 'result_' + filename),
# cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 10, (frame_width, frame_height))

# while True:
#     ret, frame = cap.read()
#     if not ret:
#         break

# # Perform necessary preprocessing on each frame
# frame = cv2.resize(frame, (300, 300))
# img = tf.keras.preprocessing.image.img_to_array(frame)
# img = np.expand_dims(img, axis=0)
# imag = np.vstack([img])
# print('hello4')
# # Make a prediction using the loaded model
# predict = model.predict(imag, batch_size=1)
# classes = np.argmax(predict)
# class_names = ['Actinic keratosis', 'Atopic Dermatitis', 'Benign keratosis', 'Candidiasis
Ringworm Tinea',
# 'Dermatofibroma', 'Melanocytic nevus', 'Melanoma', 'Normal', 'Squamous carcinoma cell',
'Vascular lesion']

# text = class_names[classes]
# print(text)
# # Display the result on the frame
# cv2.putText(frame, text, (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1.25, (255, 255, 0), 3)

# # Save the result frame
# out.write(frame)
# # Release the video capture and writer objects
# cap.release()
# out.release()
# print('5')
# # Render the result page with the result video and predicted label
# return render_template('video_output.html', predicted=text, video_file=file_path)

# return render_template('video.html')

```

```

app.config['ALLOWED_EXTENSIONS'] = {'mp4', 'avi', 'mkv', 'mov', 'jpg', 'jpeg', 'png', 'gif'}
@app.route('/video', methods=['POST', 'GET'])
def video():
    if request.method == 'POST':
        if 'file' not in request.files:
            return render_template('video.html', error='No file part')

        file = request.files['file']

        if file.filename == "":
            return render_template('video.html', error='No selected file')

        if file and allowed_file(file.filename):
            # Save the uploaded file to the 'upload' folder
            filename = secure_filename(file.filename)
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(file_path)

            # Capture video frames
            cap = cv2.VideoCapture(file_path)
            frame_width = int(cap.get(3))
            frame_height = int(cap.get(4))
            out = cv2.VideoWriter(os.path.join(app.config['UPLOAD_FOLDER'], 'result_' + filename),
                                cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'), 10, (frame_width, frame_height))

            while True:
                ret, frame = cap.read()
                if not ret:
                    break

                # Perform necessary preprocessing on each frame
                frame = cv2.resize(frame, (300, 300))
                img = tf.keras.preprocessing.image.img_to_array(frame)
                img = np.expand_dims(img, axis=0)
                imag = np.vstack([img])

```

```

# Make a prediction using the loaded model
predict = model.predict(imag, batch_size=1)
classes = np.argmax(predict)
class_names = ['Actinic keratosis', 'Atopic Dermatitis', 'Benign keratosis', 'Candidiasis Ringworm
Tine', 'Dermatofibroma', 'Melanocytic nevus', 'Melanoma', 'Normal', 'Squamous carcinoma cell',
'Vascular lesion']

text = class_names[classes]
print(text)

# Display the result on the frame
cv2.putText(frame, text, (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1.25, (255, 255, 0), 3)

# Save the result frame
out.write(frame)

# Release the video capture and writer objects
cap.release()
out.release()

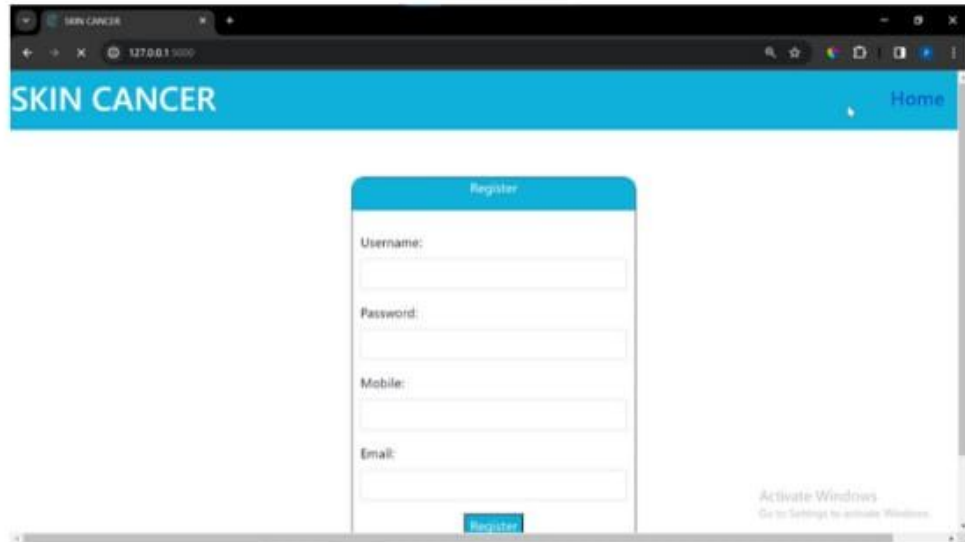
# Render the result page with the result video and predicted label
return render_template('video_output.html', predicted=text)

return render_template('video.html')

if __name__=="__main__":
    app.run(debug=True)

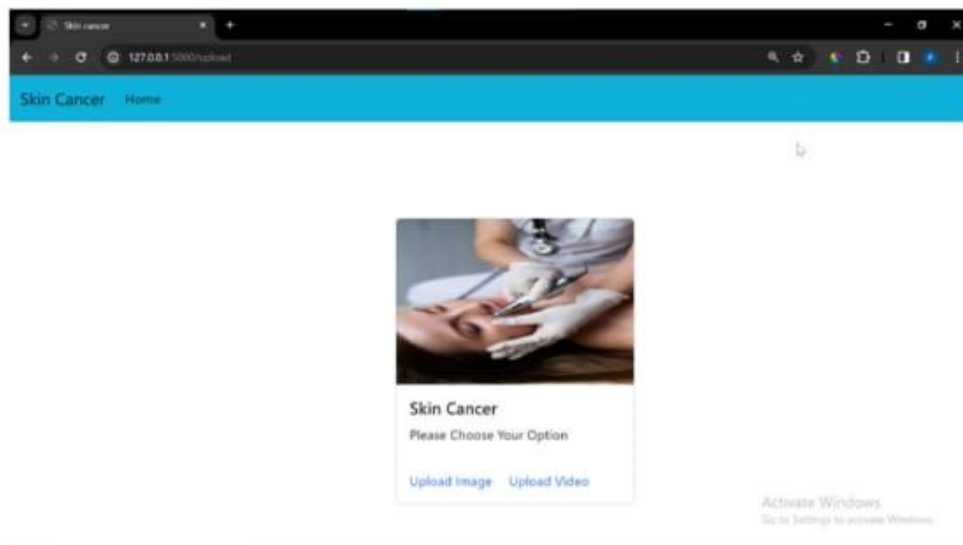
```

9.1 SAMPLE OUTPUT



A screenshot of a web browser displaying the 'SKIN CANCER' application. The browser's address bar shows '127.0.0.1:3000'. The page has a blue header with the text 'SKIN CANCER' on the left and a 'Home' link on the right. In the center, there is a white 'Register' form with a blue title bar. The form contains four input fields labeled 'Username:', 'Password:', 'Mobile:', and 'Email:'. Below these fields is a blue 'Register' button. In the bottom right corner of the page, there is a small 'Activate Windows' watermark.

FIGURE 9.2 LOGIN



A screenshot of the 'SKIN CANCER' application showing the upload interface. The browser address bar shows '127.0.0.1:3000/upload'. The blue header contains 'Skin Cancer' and a 'Home' link. The main content area features a central image of a doctor examining a patient's face. Below the image is a white box with the text 'Skin Cancer' and 'Please Choose Your Option'. At the bottom of this box are two links: 'Upload Image' and 'Upload Video'. The 'Activate Windows' watermark is visible in the bottom right corner.

FIGURE 9.3 UPLODE PROCESS

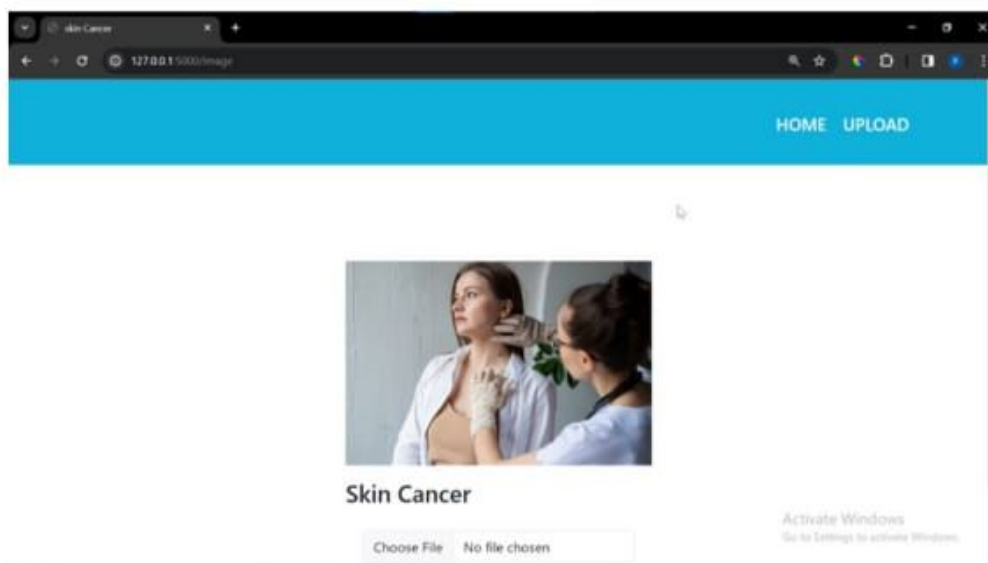


FIGURE 9.4 ANALYSE

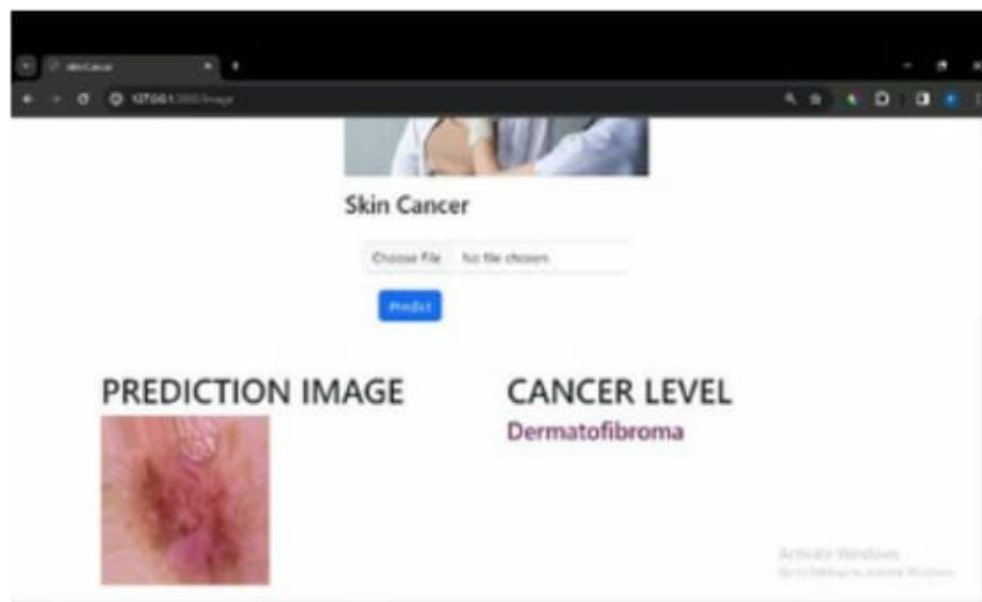


FIGURE 9.5 PREPROCESSING

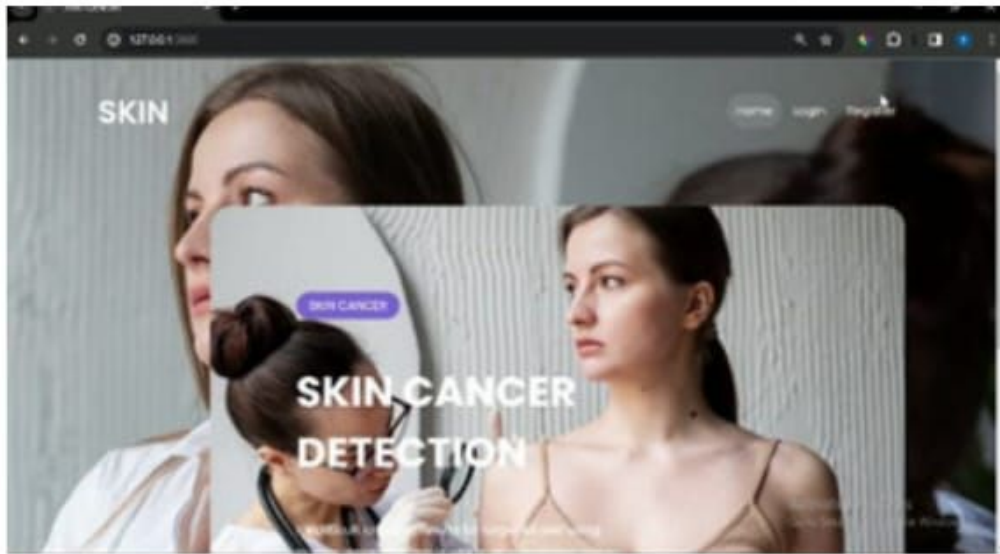


FIGURE 9.6 PREDICTION



FIGURE 9.7 DATA SET

CHAPTER - 10

REFERENCES

- [1] S. A. AlSalman, T. M. Alkaff, T. Alzaid, and Y. Binamer, "Nonmelanoma skin cancer in Saudi Arabia: Single center experience," *Ann. Saudi Med.*, vol. 38, no. 1, pp. 42–45, Jan. 2018.
- [2] K. S. Nehal and C. K. Bichakjian, "Update on keratinocyte carcinomas," *New England J. Med.*, vol. 379, no. 4, pp. 363–374, Jul. 2018.
- [3] Key Statistics for Melanoma Skin Cancer, American Cancer Society, Atlanta, GA, USA, 2022.
- [4] M. A. Albahar, "Skin lesion classification using convolutional neural network with novel regularizer," *IEEE Access*, vol. 7, pp. 38306–38313, 2019.
- [5] M. R. Hasan, M. I. Fatemi, M. M. Khan, M. Kaur, and A. Zaguia, "Comparative analysis of skin cancer (benign vs. malignant) detection using convolutional neural networks," *J. Healthc Eng.*, vol. 2021, Dec. 2021, Art. no. 5895156.
- [6] Damilola A. Okuboyejo, Oludayo O. Olugbara, and Solomon A. Odunaike, "Automating Skin Disease Diagnosis Using Image Classification," *Proceedings of the World Congress on Engineering and Computer Science 2013 Vol II, WCECS 2013*, 23-25 October, 2013, San Francisco, USA.
- [7] R. Yasir, M. A. Rahman and N. Ahmed, "Dermatological disease detection using image processing and artificial neural network," *8th International Conference on Electrical and Computer Engineering*, Dhaka, 2014, pp. 687-690, doi: 10.1109/ICECE.2014.7026918.