SKINHELP: A DERMA TOOL

A PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

Shivam Sood (19103108)

Manav Dhiman (19103063)

Under the supervision of

Dr. Samayveer Singh
Assistant Professor

Department of Computer Science and Engineering
Dr. B. R. Ambedkar National Institute of Technology Jalandhar
-144008, Punjab (India)

May 2024

Dr. B. R. Ambedkar National Institute of Technology Jalandhar

CANDIDATES' DECLARATION

We hereby certify that the work presented in this project report entitled "**SkinHelp: A Drema Tool**" in partial fulfillment of the requirement for the award of a Bachelor of Technology degree in Computer Science and Engineering, submitted to the Dr. B R Ambedkar National Institute of Technology, Jalandhar is an authentic record of our own work carried out during the period from July 2023 to May 2024 under the supervision of Dr. Samayveer Singh, Assistant Professor, Department of Computer Science & Engineering, Dr. B R Ambedkar National Institute of Technology, Jalandhar.

We have not submitted the matter presented in this report to any other university or institute for the award of any degree or any other purpose.

Date: 9th May, 2024

Submitted by
ShivamSood (19103108)
ManavDhiman (19103063)

This is to certify that the statements submitted by the above candidates are accurate and correct to the best of our knowledge and are further recommended for external evaluation.

Dr. Samayveer Singh, Supervisor
Assistant Professor
Deptt. of CSE

Dr. Rajneesh Rani
Head and Associate Professor
Deptt. of CSE

ACKNOWLEDGEMENT

It is true that hundreds of people work behind the scenes for the success of a play. The end result of the SkinHelp project required a lot of guidance and help from many people and our group was very lucky to receive this during the course of the project. Whatever we are today is only due to such supervision and assistance and we thank them from the bottom of our hearts.

We would like to express our deepest gratitude to our project mentor Dr. Samayveer Singh Assistant Professor, who believed in our ideas and suggested new ideas when needed. He fully supported us in solving our problems.

We would like to express our deepest gratitude to Dr. Rajneesh Rani, Head of the Department of Computer Science and Engineering, for her direct and indirect support.

We are grateful to the Dr Aruna Malik, Coordinator Major Project for providing us mentors and all other support.

We are extremely thankful to have constant encouragement and guidance from all the Faculties of the Department of Computer Science & Engineering. We would also like to express our sincere thanks to all laboratory staff for their timely support.

Thank You.

[Shivam, Manav]

ABSTRACT

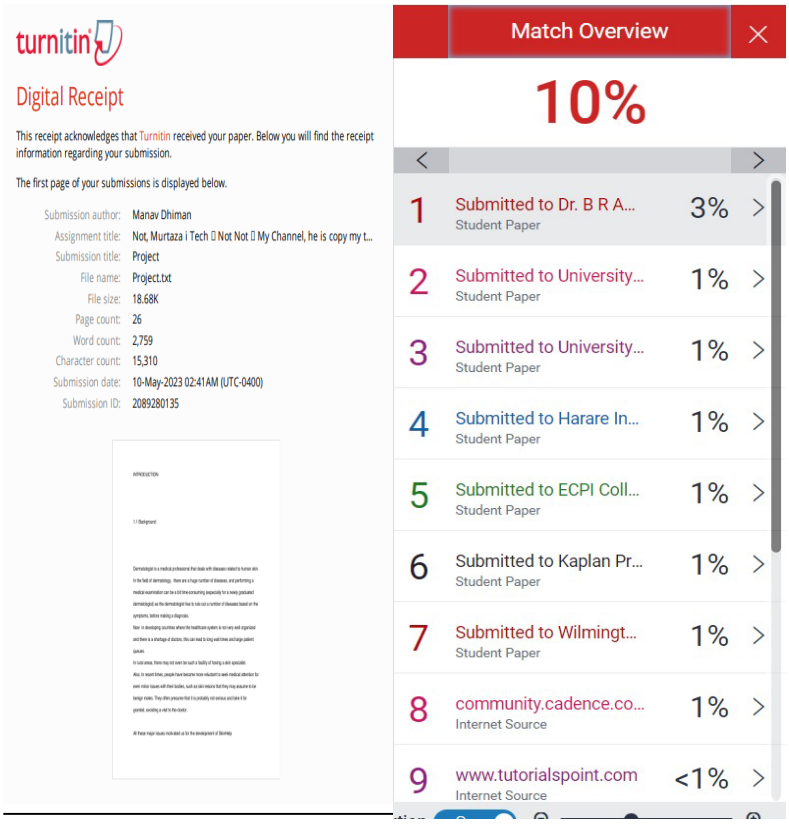
SkinHelp is an innovative web-based application that revolutionizes the preliminary diagnosis of skin diseases through the application of advanced deep learning technologies. This project provides a practical solution to the challenges faced by individuals in regions with limited access to specialized dermatological care. The technical foundation of SkinHelp is built on a robust artificial intelligence framework that employs a Convolutional Neural Network (CNN), enhanced by the principles of transfer learning to ensure high accuracy and efficiency in disease classification. The project begins with the collection of a comprehensive dataset of skin lesion images, specifically utilizing the HAM10000 dataset, which is a widely recognized resource in dermatological AI research. This dataset includes images of various skin diseases, which are preprocessed to normalize and augment the data, thus ensuring the model trains on high-quality inputs. The preprocessing steps involve resizing images, enhancing image quality, and applying data augmentation techniques to increase the diversity of the dataset, simulating different lighting conditions, rotations, and scales to make the model robust against real-world variations.

Following data preparation, the project utilizes transfer learning to fine-tune a pre-trained DenseNet169 architecture. This approach allows the model to leverage learned features from a vast array of images not limited to dermatological conditions, thereby enhancing its ability to generalize from dermatological imagery. The model training phase is meticulously managed with a focus on optimizing performance metrics such as accuracy and the F1 score, ensuring that the model not only recognizes common conditions but also rarer diseases with high reliability. SkinHelp's application layer is developed using Flask, a lightweight and powerful web framework that facilitates quick deployment of Python applications. The choice of Flask allows for the seamless integration of backend AI processes with a front-end user interface, enabling users to interact with the tool through a simple, intuitive web interface. Users can upload images of skin conditions and receive immediate predictions with associated confidence scores. This interaction is supported by a backend infrastructure that handles image data processing, model inference, and result presentation in real time.

The deployment of the model is facilitated through Google Colab and AWS EC2, which provide the necessary computational power and scalability for training and hosting the AI models. These platforms were chosen for their robustness and cost-effectiveness, ensuring that the application remains accessible and functional under varying loads of user requests. In conclusion, SkinHelp stands out as a significant technological advancement in the field of dermatological diagnostics. It demonstrates the successful application of machine learning and web development technologies to create a tool that not only enhances the accessibility of healthcare but also empowers individuals to take preliminary steps in skin health management. The project underscores the potential of AI in transforming healthcare delivery and sets a precedent for future innovations in the domain of medical diagnostics.

PLAGIARISM REPORT

We have checked plagiarism for our Project Report for our project a **Turnitin.** We are thankful to our mentor Dr. Samayveer Singh for guiding us at this. Below is the digital receipt. The Plagiarism is approximately 10%.



turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Manav Dhiman
Assignment title:	Not, Murtaza i Tech Not Not My Channel, he is copy my t...
Submission title:	Project
File name:	Project.txt
File size:	18.68K
Page count:	26
Word count:	2,759
Character count:	15,310
Submission date:	10-May-2023 02:41AM (UTC-0400)
Submission ID:	2089280135

Match Overview

10%

1	Submitted to Dr. B R A... Student Paper	3%
2	Submitted to University... Student Paper	1%
3	Submitted to University... Student Paper	1%
4	Submitted to Harare In... Student Paper	1%
5	Submitted to ECPI Coll... Student Paper	1%
6	Submitted to Kaplan Pr... Student Paper	1%
7	Submitted to Wilmingt... Student Paper	1%
8	community.cadence.co... Internet Source	1%
9	www.tutorialspoint.com Internet Source	<1%

LIST OF FIGURES

Figure number	Description	Page number
Figure 1	User Dashboard	10

LIST OF TABLES

Table number	Description	Page number
Table 1	User Dashboard	10

LIST OF ABBREVIATIONS

CNNs: Convolutional Neural Networks

TABLE OF CONTENTS

	CANDIDATES' DECLARATION	i
	ACKNOWLEDGEMENT	ii
	ABSTRACT	iii
	PLAGIARISM REPORT	iv
	LIST OF FIGURES	v
	LIST OF TABLES	vi
	LIST OF ABBREVIATIONS	vii
1.	INTRODUCTION	
	1.1. Background of the Problem	1
	1.2. Literature Survey	2
	1.3. Problem Statement	3
	1.4. Motivation	3
	1.5. Feasibility	4
2.	PROPOSED SOLUTION	5-14
3.	TECHNOLOGY ANALYSIS	
	UML Diagram	15
	Tech Stack Analysis	16-20
4.	ECONOMIC ANALYSIS	21
5.	RESULT AND DISCUSSION	
	APP Usage Instructions	22-25
	Risk Analysis	26
6.	CONCLUSION	27
7.	REFERENCES	28

CHAPTER 1

INTRODUCTION

1.1 Background

Any problem related to human skin is cured by a doctor known as Dermatologist. Dermatology is the branch of medicine that focuses on the diagnosis and treatment of skin diseases and disorders. A dermatologist is a medical doctor who specializes in this field and is trained to handle a variety of skin conditions, ranging from common skin problems like acne, eczema, and psoriasis, to more serious conditions like skin cancer.

The process of diagnosing a skin condition can be quite complex and time-consuming. Dermatologists need to examine the patient's skin closely, taking into account the patient's medical history and any symptoms they may be experiencing. This often involves ruling out a number of possible conditions based on the patient's symptoms and the appearance of their skin.

In countries with poorly organized healthcare systems or a shortage of doctors, patients with skin conditions may have to wait for long periods to see a dermatologist. In rural areas, there may not be any dermatologists available at all. This can lead to delays in treatment and a reduced quality of care for patients with skin conditions. Additionally, some people may be reluctant to seek medical attention for minor skin issues, assuming that they are not serious. This can lead to delays in diagnosis and treatment, and in some cases, may result in more serious health problems.

The development of SkinHelp aims to address these issues by providing an online platform where patients can get a preliminary diagnosis for their skin condition without having to wait for a doctor's appointment. SkinHelp uses artificial intelligence to analyze images of the patient's skin and provide an initial diagnosis based on the symptoms and appearance of the skin. This can help patients to identify potential skin problems early on and seek appropriate medical attention if necessary.

Overall, the development of SkinHelp is a significant step towards improving access to healthcare for people with skin conditions, particularly in areas where there may be a shortage of dermatologists or limited access to medical facilities.

1.2. Literature Survey

After going through numerous research papers and GitHub repos to gather crucial information needed to execute this project, we came to the conclusion to use a CNN model [1] [2] [3] [6] trained using transfer learning [4][6][5] for skin disease classification.

Deep Convolutional Neural Networks (CNNs) have transformed computer vision and image processing tasks (which is the case for skin disease classification) in the latest years. A CNN is a kind of neural network that is particularly designed for image processing, where the input image is made to pass through multiple layers of convolutional filters, pooling, and activation functions to extract features.

Deep CNNs typically consist of multiple convolutional layers, followed by fully connected layers that perform the classification or regression task. The number of layers and filters in each layer can be adjusted depending on the complexity of the task and the size of the dataset.

Transfer learning has proven to be highly effective in computer vision applications (which is the case for skin disease classification), especially in cases where the size of the available dataset is relatively small (which is also the case for data around skin lesions).

One of the reasons for this is that many computer vision tasks share common features and patterns. For example, a model trained on ImageNet, which is a large dataset of images with 1,000 object categories, can learn a lot of useful features that can be applied to other computer vision tasks, such as object detection, segmentation, and classification.

By using a pre-trained model as a starting point and fine-tuning it on a specific task, transfer learning can help improve the accuracy and speed of a model while reducing the amount of training data required.

1.3. Problem Statement and its Necessity

The major issues that spurred the arrangement are as following:

1. **Unavailability of Healthcare facilities**

   In remote locations of any developing country, where there are no super specialists, SkinHelp can help people get a preliminary look at their skin problems.

2. **Early Diagnosis**

People these days are reluctant to get themselves checked by a doctor on mild skin conditions, which when detected later turn out to be quite harmful. So the general public can also use SkinHelp to get an idea of the problem they are facing and thus motivate them to get diagnosed by a doctor.

3. **Getting Skin Tests: -**
   Usually the procedure for a skin examination is such that first a doctor looks at the skin and other symptoms and then may or may not recommend some tests in order to make a diagnosis. If the doctor does recommend a test, then the patient has to revisit the doctor with the test results. With SkinHelp, the nurse can first upload the lesion and have a preliminary look at the issue and based on the confidence of predictions, the nurse herself/himself can recommend further skin tests and save doctor's time, which he/she can use to cure more patients.

4. **Assisting local Practitioners and newly Graduated Dermatologists:**
   In tier 3 or 4 cities or rural areas in India, there are no specialist doctors but RMP's (Rural Medical Practitioners), who are just MBBS doctors. They can use SkinHelp to get an estimate of the disease and further refer the patient to some other Hospital with skin specialists, if the situation is serious. Newly graduated Dermatologists,who lack experience, can also use SkinHelp to get a preliminary examination of the issue while making the diagnosis.

1.4. Motivation

- Access to effective solutions for skin diseases can be limited and expensive, requiring specialized expertise for successful treatment.
- SkinHelp was created to make skin disease diagnosis more accessible to people worldwide, including those in rural areas.
- Individuals can diagnose their skin conditions quickly and easily by taking a photo of the affected area and uploading it to the application.
- Deep learning algorithms analyze the image to predict the probability of a skin disease.

- This innovative approach provides an affordable and user-friendly way for people to gain insight into their skin health, even without specialized medical training or resources.

1.5. Feasibility : Non-Technical and Technical

As with any successful project, it is very crucial to have a plan of whether a project is feasible from different standpoints. The various possibilities/standpoints can be summarized as follows.

TECHNICAL:

With the availability of powerful high-level programming languages such as Python, along with comprehensive support for Machine Learning algorithms and Graphical Processing Unit assets provided on the internet (cloud) by Google Colaboratory and AWS EC2 instance, even low-end personal computers can now train sophisticated models. Furthermore, Open source web frameworks like flask makes it easy to develop web applications. Also, Open source UML editor plantText helps in coming up with UML diagrams.

SOCIAL:

There is currently no widely adopted or mainstream application that addresses this specific problem domain.

ECONOMICAL FEASIBILITY:

The development expenses for this project are not expected to be significant since we will be utilizing open source libraries and publicly available datasets to train our model.

SCOPE:

SkinHelp aims to help medical practitioners and normal people in getting a preliminary review of their skin problems

1.6 Research Objectives

- SkinHelp aims to revolutionize preliminary skin disease diagnosis by leveraging advanced deep learning technologies, specifically focusing on enhancing accessibility for individuals in regions with limited dermatological care.
- Employ transfer learning on a pre-trained DenseNet169 architecture, SkinHelp ensures high accuracy and reliability in diagnosing a wide range of skin conditions, thereby transforming healthcare delivery with AI-powered diagnostics.

CHAPTER 2
PROPOSED SOLUTION

SkinHelp is a web application that utilizes deep learning classification algorithms to detect and classify various skin lesions. With the help of this application, users can get a preliminary diagnosis for their skin conditions in a quick and efficient manner. The application can identify and classify seven major skin diseases with a high level of accuracy, providing users with a confidence score for each classification.

The development of SkinHelp involves several steps, including data collection, preprocessing, training, and deployment. The first step involves collecting a large dataset of images of skin lesions from various sources. The dataset is then preprocessed to remove any irrelevant or duplicate images and to standardize the image size and format.

Next, the deep learning algorithm is trained on the preprocessed dataset. This involves using a neural network architecture to learn the features and characteristics of the skin lesions and to develop an accurate classification model. The model is trained using a large amount of data and undergoes several iterations of testing and refinement to improve its accuracy.

Once the training is complete, the model is deployed on the SkinHelp web application, where users can upload images of their skin lesions for analysis. The application uses the trained model to classify the lesion and provides the user with a diagnosis and confidence score for each classification. This can help users to identify potential skin problems early on and seek appropriate medical attention if necessary.

Overall, SkinHelp is a valuable tool for improving access to healthcare for people with skin conditions. The application provides a fast, accurate, and accessible way for users to obtain a preliminary diagnosis for their skin lesions, which can lead to early detection and treatment of potential health problems.

We will first provide an overview of the project and then proceed to explain the solution in a step-by-step manner.

- The dashboard of the application allows patients to upload images of their affected skin.

Figure 2.1: User Dashboard

- Once the image is uploaded, the website redirects to the analysis of the image and predicts the percentage chance of 7 main skin diseases.

Figure 2.2: Result Page

As specified earlier, transfer learning was used to train the model and DenseNet169 [7] is the pre-trained model that is custom trained

The following was the procedure followed to train the model:

1. **Data Gathering**

   Our model was trained on the HAM10000 dataset, which consists of 10,000 images of skin lesions and is freely available from Harvard.

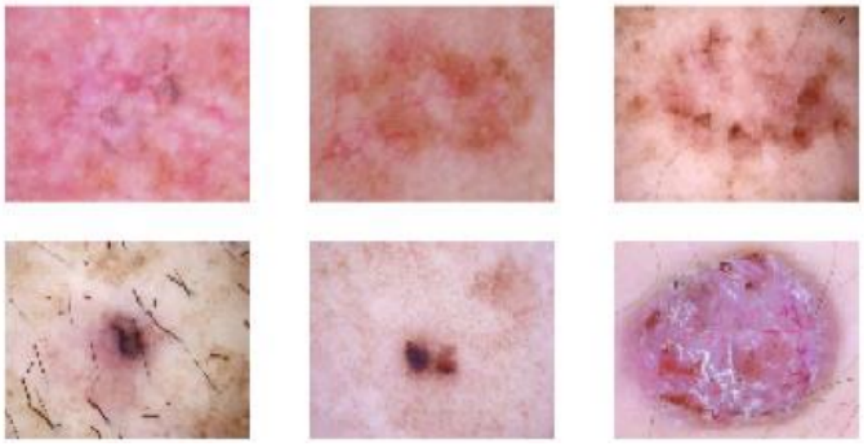


Figure 2.3: Sample Images from Dataset

2. **Data Labeling**

```
# Categories of the diferent diseases
lesion_type_dict = {
    'nv': 'Melanocytic nevi',
    'mel': 'Melanoma',
    'bkl': 'Benign keratosis ',
    'bcc': 'Basal cell carcinoma',
    'akiec': 'Actinic keratoses',
    'vasc': 'Vascular lesions',
    'df': 'Dermatofibroma'
}

df['lesion'] = df.dx.map(lesion_type_dict)
df.head()
```

Figure 2.4: Labeling of data

In this Python dictionary, the abbreviations for various types of skin lesions are mapped to their corresponding full names. The motive of this dictionary is to show the labels of distinct classes in the model assessment and reporting functions.

3. Dataset Analysis

In this analysis, we explore the distribution of images among the seven categories of skin lesions. It is evident that the dataset is unbalanced, as the 'Melanocytic nevi' class contains 6705 samples, while only 115 samples are available for the 'Dermatofibroma' category. If we train using the entire dataset, our model will be suffering from overfitting.

```
print(df.lesion.value_counts())
```

Figure 2.5: Code to print value counts

```
Melanocytic nevi          6705
Melanoma                  1113
Benign keratosis          1099
Basal cell carcinoma       514
Actinic keratoses          327
Vascular lesions           142
Dermatofibroma             115
Name: lesion, dtype: int64
```

Figure 2.6: Displaying count of each Categories

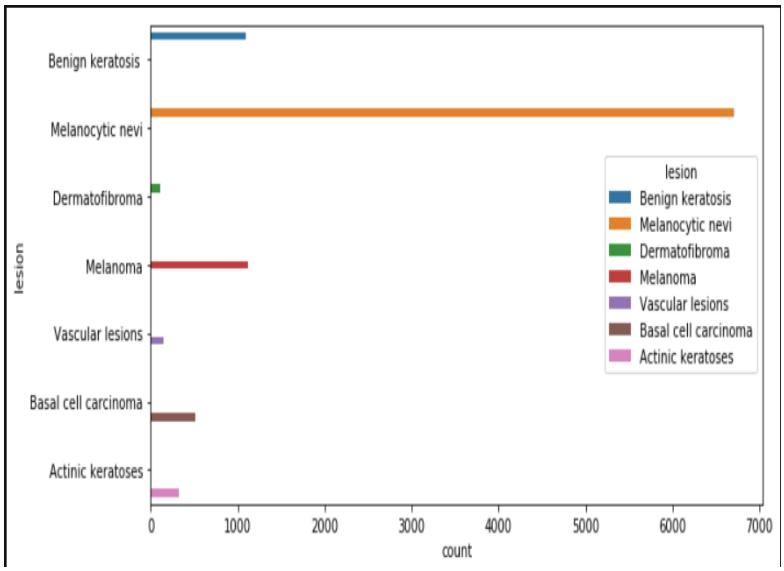


Figure 2.7: Graphical representation of counts

4. Under sampling

To address the issue of class imbalance, we utilize a technique called random under sampling, specifically targeting the classes that have over 200 samples.

```
num_sample = 200

df_df = df.loc[df['dx'] == "df"][0:115]
df_vasc = df.loc[df['dx'] == "vasc"][0:142]
df_akiec = (df.loc[df['dx'] == "akiec"]).sample(num_sample)
df_bcc = df.loc[df['dx'] == "bcc"][0:num_sample].sample(num_sample)
df_bkl = df.loc[df['dx'] == "bkl"][0:num_sample].sample(num_sample)
df_mel = df.loc[df['dx'] == "mel"][0:num_sample].sample(num_sample)
df_nv = df.loc[df['dx'] == "nv"][0:num_sample].sample(num_sample)

df = pd.concat([df_akiec, df_bcc, df_bkl, df_df, df_mel, df_nv, df_vasc])
df = shuffle(df)
```

Figure 2.8: Code for under sampling

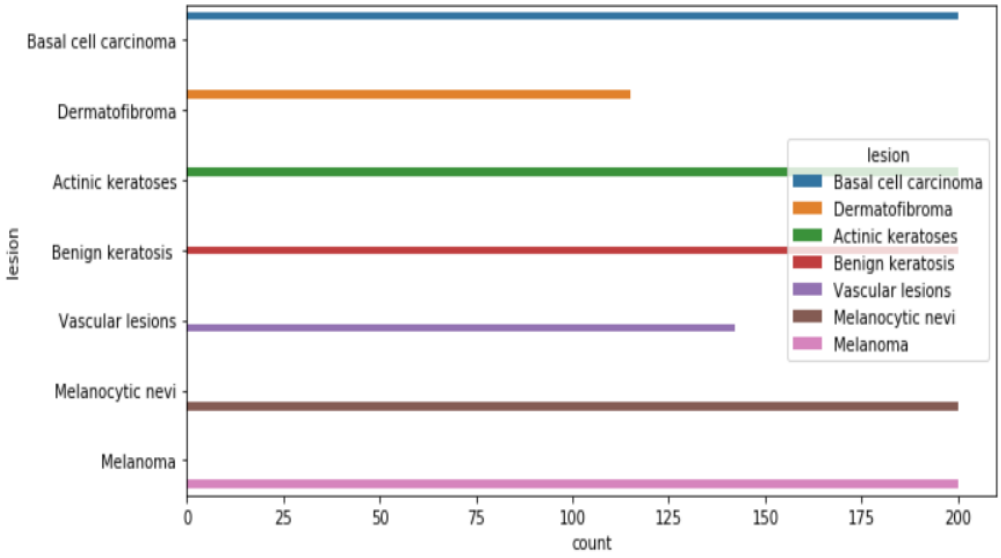


Figure 2.9: Results after Undersampling

5. Loading the data

Subsequently, we generate an ImageDataBunch object based on the undersampled dataset and apply typical data augmentations to it. Furthermore, we normalize this dataset by utilizing the ImageNet statistics.

```
tfms = get_transforms(flip_vert=True)
data = ImageDataBunch.from_df("data/", df, fn_col=1, suffix='.jpg', label_col=7, ds_tfms=tfms, size=224, bs=16)
data.normalize(imagenet_stats)
```

Figure 2.10: Code for ImagDataBunch

```
ImageDataBunch;

Train: LabelList (1006 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224)
y: CategoryList
Basal cell carcinoma,Dermatofibroma,Dermatofibroma,Benign keratosis ,Actinic keratoses
Path: data;

Valid: LabelList (251 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224)
y: CategoryList
Benign keratosis ,Basal cell carcinoma,Dermatofibroma,Actinic keratoses,Actinic keratoses
Path: data;

Test: None
```

Figure 2.11: Results from above code

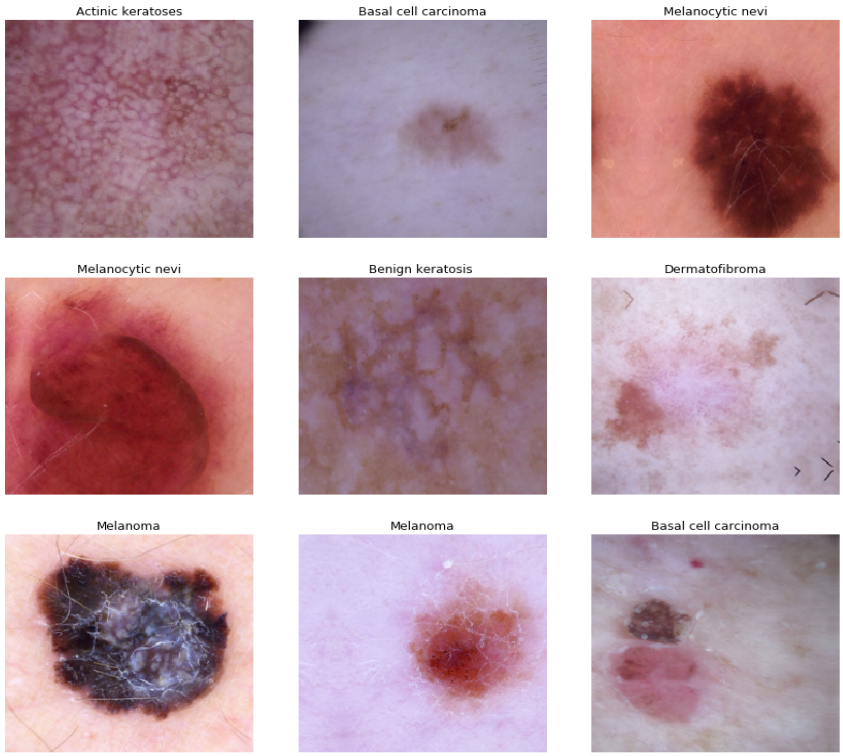


Figure 2.12: Labeling of each disease

6. Model Training and Evaluation

At this point, we generate a learner object by using a frozen, pre-trained densenet169 architecture.

```
learner = cnn_learner(data, models.densenet169, metrics=[accuracy, FBeta(average='macro')], model_dir="../models/")
learner.loss_func = nn.CrossEntropyLoss()
```

Figure 2.13: Training of model

To begin the training process, we first need to determine the ideal learning rate. We accomplish this by utilizing a learning rate finder. Based on the plot generated by this finder, the most significant drop in the loss value occurs when the learning rate is approximately at the 1e-03 value.

```
learner.lr_find()
learner.recorder.plot()
```

Figure 2.14: Code for ploting Learning rate curve

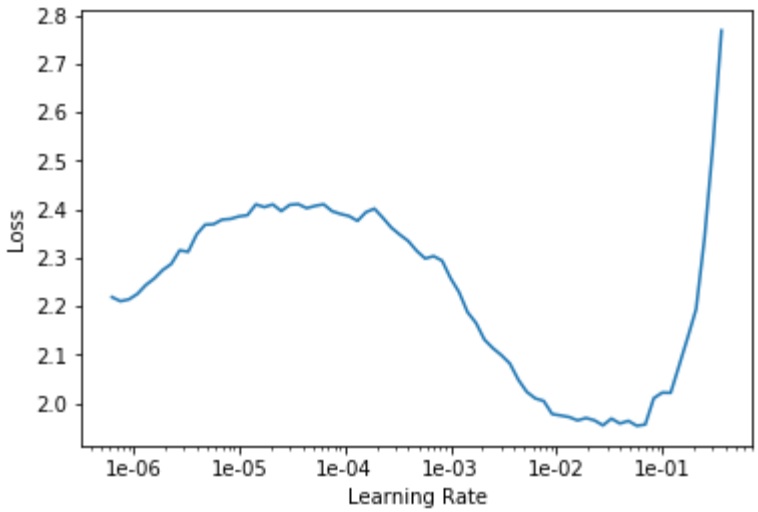


Figure 2.15: Learning Curve

We now proceed to train the model using the optimal learning rate. We also preserve the best model based on the accuracy achieved in each epoch. This saved file will be utilized to deploy our classifier later.

```
learner.fit_one_cycle(30, 1e-3, callbacks=[SaveModelCallback(learner, every='improvement', monitor='accuracy', name='model_best')])
```

Figure 2.16: Code for finding accuracy at each epoch

```
Better model found at epoch 0 with accuracy value: 0.350597620010376.
Better model found at epoch 1 with accuracy value: 0.6095617413520813.
Better model found at epoch 2 with accuracy value: 0.7091633677482605.
Better model found at epoch 3 with accuracy value: 0.7490040063858032.
Better model found at epoch 4 with accuracy value: 0.788844645023346.
Better model found at epoch 7 with accuracy value: 0.8167330622673035.
Better model found at epoch 8 with accuracy value: 0.824701189994812.
Better model found at epoch 9 with accuracy value: 0.8446215391159058.
Better model found at epoch 11 with accuracy value: 0.8645418286323547.
Better model found at epoch 14 with accuracy value: 0.8844621777534485.
Better model found at epoch 15 with accuracy value: 0.892430305480957.
Better model found at epoch 19 with accuracy value: 0.912350594997406.
```

Figure 2.17: Accuracy at each epoch

The most promising model was attained at epoch 19, with an accuracy of approximately 91.2% and an f-measure of 91.7%.

7. Model Interpretation

```
learner = learner.load("model_best")

interp = ClassificationInterpretation.from_learner(learner)
interp.plot_confusion_matrix(figsize=(10,8))
```

Figure 2.18: Plotting Confusion matrix

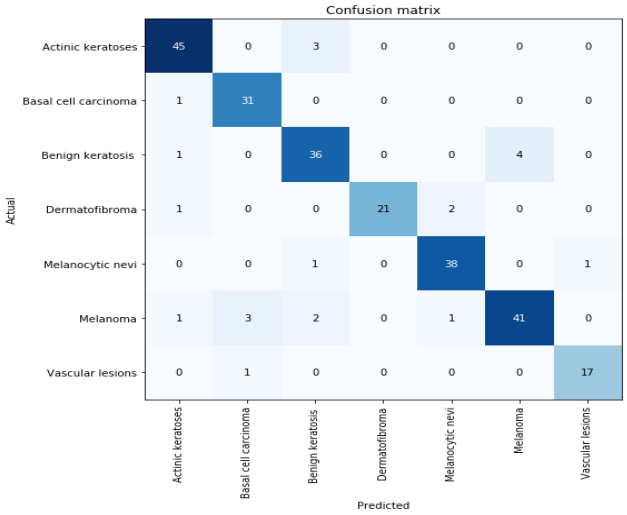


Figure 2.19: Confusion matrix

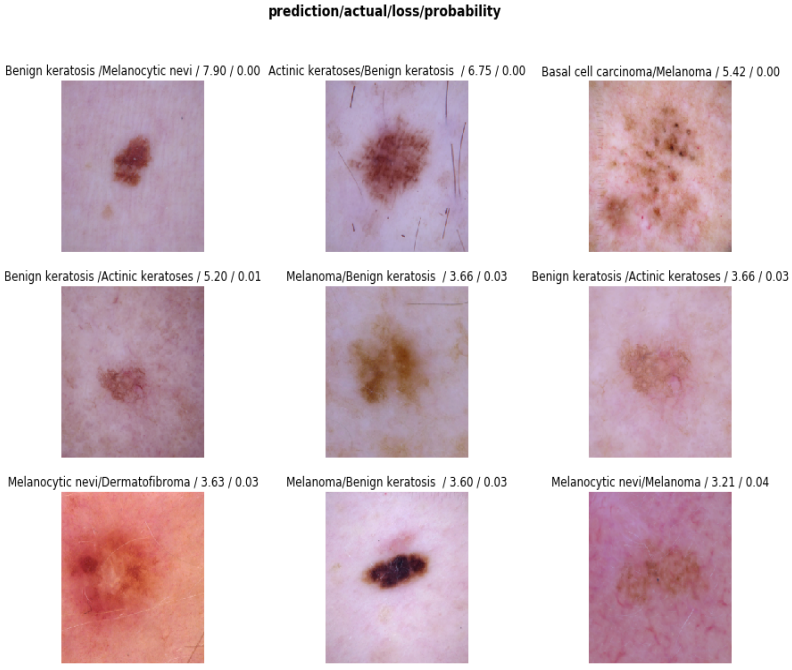


Figure 2.21: Model results

CHAPTER 3
TECHNOLOGY ANALYSIS

3.1. UML Diagram

The UML diagram you mentioned describes the flow of data and functions within the SkinHelp application. Here is a breakdown of the steps involved:

- The user uploads an image of their skin condition through the front-end interface.
- The front-end sends the uploaded image data to the back-end server.
- The back-end server receives the image and sends it to the machine learning (ML) trained model for analysis.
- The ML model processes the image and returns its prediction of the skin condition to the back-end server.

- The back-end server receives the prediction and sends it back to the front-end interface.
- The front-end interface displays the prediction results to the user.

This UML diagram shows the overall architecture and workflow of SkinHelp in a simplified manner.

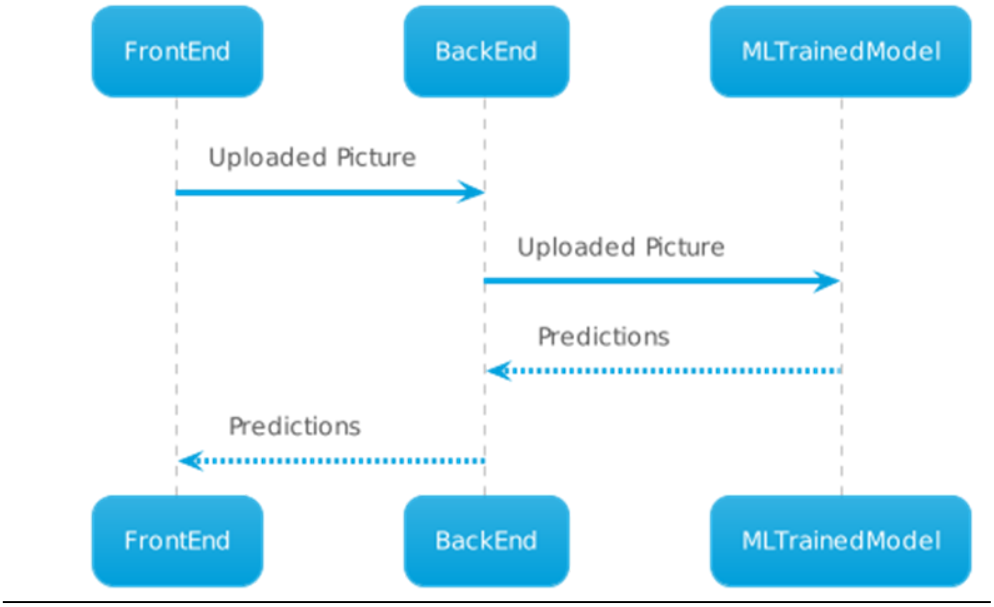


Figure 3.1: Uml Diagram

3.2. Tech Stack Analysis

The technologies used in the project are:-

3.2.1. Flask

Flask is an open-source web framework that enables developers to create web applications in Python. It was developed by Armin Ronacher and was first released in 2010. Flask stands out for its simplicity and flexibility, as it does not include pre-built functionalities like form validation, authentication, or database abstraction. Instead, Flask provides a micro-framework that offers essential tools to create web applications, giving developers the freedom to choose the tools they require for specific functionalities. Flask relies on the Werkzeug WSGI toolkit and the Jinja2 template engine to manage requests and responses, as well as to render HTML templates. It also supports extensions, which are third-party packages that extend the functionality of the framework. Flask is a popular choice for building small and medium-sized web applications, RESTful APIs, and prototypes due to its small size, flexibility, and customization options.

Figure 3.2: Flask web development, one drop at a time.

3.2.2. Google Colab

Google Colab is a web-based coding framework that enables users to write and run Python code without any setup or configuration. It offers free access to GPUs and provides easy sharing options. With Colab, users can write and execute code, build machine learning models, and collaborate with other developers on projects. It is based on Jupyter Notebooks and allows for the use of markdown cells to create formatted text alongside code. Additionally, it is entirely self-contained, meaning users do not need to install any software or manage any infrastructure to use it.

Figure 3.3: Google Colaboratory

3.2.3. Python

Python is a widely used high-level programming language that is interpreted and interactive. It supports object-oriented programming and is well-suited for a variety of applications, including machine learning. Python has a strong library ecosystem, with extensive support for machine learning algorithms through popular libraries such as sci-kit-learn, Keras, and TensorFlow. These libraries simplify the development and management

of machine learning algorithms, making it easier for developers to implement complex models and analyze large datasets.

Figure 3.4: Python

3.2.4. TensorFlow

Keras is a user-friendly API for TensorFlow that simplifies the development of machine learning models, with a focus on modern deep learning. It provides pre-built components, such as layers and optimizers, for easy model creation, and supports experimentation and iteration. Keras is widely used for research and production applications due to its ease of use and flexibility.

Figure 3.5: Python

3.2.5. HTML

The hypertext markup language, known as HTML, is used to create web pages and web applications. It provides a structure for web content by using tags and attributes to define different elements of a web page, such as headings, paragraphs, links, images, forms, and tables. HTML documents consist of tags and text content. Tags are enclosed in angle brackets and define the structure and formatting of the content, while attributes provide

additional information or settings for elements. HTML is not a programming language, but rather a markup language that is interpreted by web browsers to display web page content. It can be used with other languages like CSS and JavaScript to create more dynamic web applications.

Figure 3.6: HTML5

3.2.6. CSS

CSS, or Cascading Style Sheets, is a language used to define the presentation and style of web pages. By separating design from content, it enables developers to create consistent and attractive layouts for web pages and to control the appearance of individual elements such as fonts, colors, margins, and borders. CSS works by using rules to target HTML elements and define their style properties, allowing developers to group selectors and apply styles to multiple elements at once. CSS is applied to HTML documents in various ways and is essential for creating modern, visually appealing, and responsive web pages that can be viewed on different devices and screen sizes.

Figure 3.7: HTML5

3.2.7. AWS EC2

Amazon Web Services (AWS) offers a web service called Amazon Elastic Compute Cloud (EC2), which enables users to lease virtual computers for the purpose of running their own applications. EC2 provides scalable computing capacity in the cloud, making it easy to quickly launch and manage virtual machines (VMs) and deploy applications without having to invest in costly hardware or infrastructure. Users can choose from a variety of

preconfigured machine images or create their own custom images and can configure their VMs to meet their specific needs in terms of computing, memory, storage, and networking resources. EC2 also offers a range of security features and tools to help users secure their applications and data.

Figure 3.8: Amazon EC2

CHAPTER 4
ECONOMIC ANALYSIS

We have built our applications and platforms using free and secure technologies, including APIs, datasets, and dependencies. As these are also free software, they only require support and a willingness to adapt to any changes that may arise. Therefore, we guarantee that there will be no costs associated with using our application, and all requirements will be completely free.

- Our goal is to offer solutions that are affordable, user-friendly, and equipped with the necessary features to tackle common challenges.
- As for the various development stacks used, they are freely available and require an internet connection.
- Google Colabis free to use.

- AWS subscription is free upto a certain time and the basic EC2 instance resources are enough to train the model for free.

CHAPTER 5

RESULT AND DISCUSSION

5.1. App Usage Instructions

Hardware and software requirements: Requires a laptop with any computability to access the web application. When a user opens the SkinHelp application, they will see a navigation bar with the SkinHelp logo. Additionally, they will be prompted to upload an image of their skin lesion from their device to begin the skin cancer detection process. The user interface is designed to be simple and intuitive, allowing users to quickly and easily perform skin cancer detection without needing any medical expertise. This feature is an essential part of SkinHelp's mission to provide accessible and affordable skin disease diagnosis to people around the world.

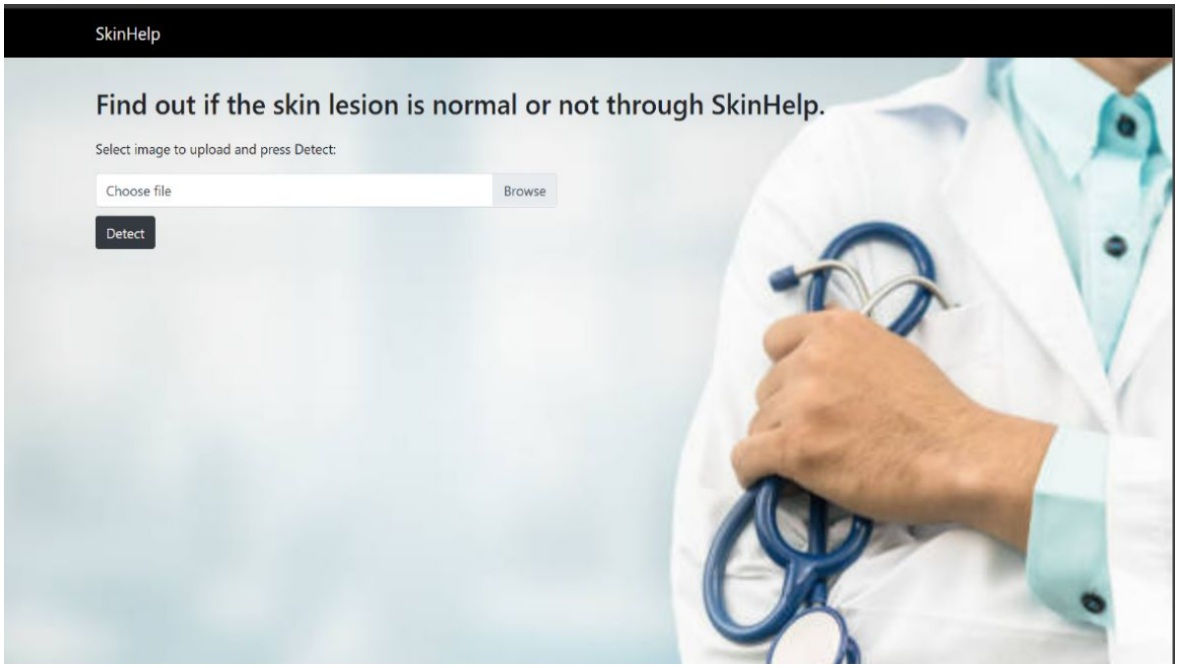


Figure 5.1: User Dashboard

- When the user clicks on the "browse" button in SkinHelp, a pop-up window will appear, enabling the user to browse through their device's directories to select the image they want to upload. This user-friendly feature allows users to quickly locate and select the image of their skin lesion, making the process of skin cancer detection more straightforward and accessible. By simplifying the process of uploading images, SkinHelp is helping to make skin disease diagnosis more accessible to people around the world, regardless of their technical abilities or medical knowledge.

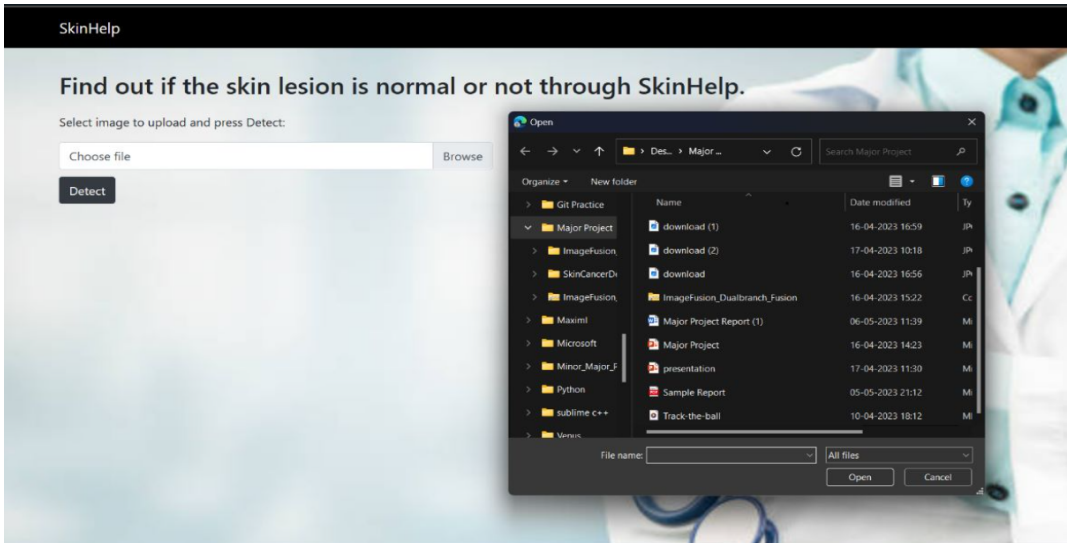


Figure 5.2: Finding Image from Directory

- After selecting the image of the skin lesion, the user should then click on the "detect" button within the SkinHelp application. This will initiate the skin cancer detection process, and the deep learning algorithms within SkinHelp will analyze the image to predict the probability of a skin disease. By clicking the "detect" button, users can receive a quick and reliable diagnosis for their skin lesion, making it easier to determine the best course of action for treatment or further medical consultation.

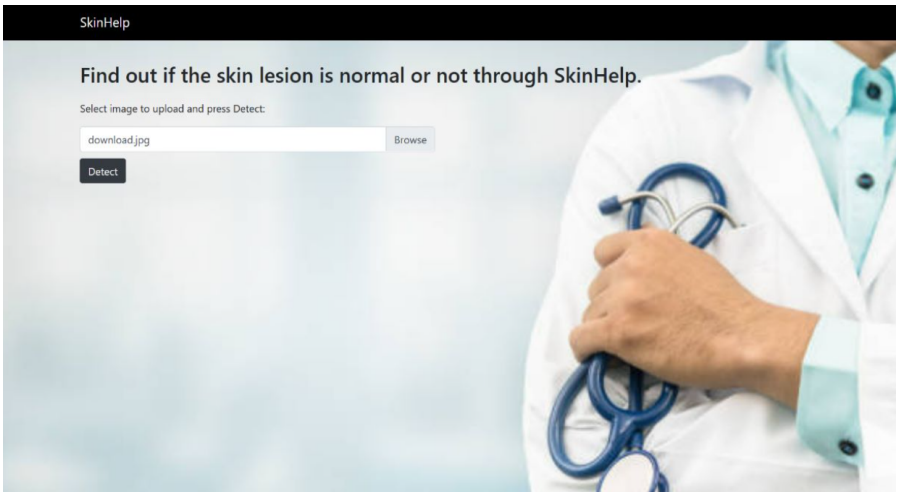


Figure 5.3: Image Uploading for diagnoses

- Once the skin cancer detection process is complete, SkinHelp will display the predicted skin diseases that may be associated with the uploaded image. The application will also indicate the confidence level of the predicted lesion type,

providing users with an understanding of the probability that the prediction is correct. This information can help users to make informed decisions about their skin health, such as seeking further medical consultation or scheduling an appointment with a dermatologist for a more thorough examination.

Figure 5.4: Results of Image

5.2 Risk Analysis

Designing or developing anything can't be risk-free. Risks can occur, but with a proper risk mitigation, monitoring, and management plan, we can make our project resistant to any adverse consequences, should any risk occur. Hereby, various risks are highlighted that are involved in our project, SkinHelp.

1. Stakeholders don't accept the system due to various reasons.
2. Risk of application crashes due to high load.

Point 2 can be easily tackled by ensuring proper scalability measures, which can be put into place should the user load exceed too much (the code can be deployed on the cloud, which is equipped to handle scalability issues).

The main risk boils down to Point 1, as a lack of acceptability is a major issue for any developer. We have reduced the risk level by having various discussions with both technical and non-technical stakeholders at various stages of the development of the

project to ensure the stakeholders and developers were on the same page as far as the project scope and requirements were concerned.

CHAPTER 6
CONCLUSION

Skin diseases are the fourth most common human illness. SkinHelp aims to provide robust skin disease detection that can help normal people, assist medical practitioners, and serve various other purposes. We should point out that SkinHelp does not aim to replace doctors but only be used for assisting doctors or providing them with a preliminary diagnosis and a doctor's opinion/diagnosis should always be preferred over SkinHelp's detections as no machine can yet fully replace human input, especially in the field of healthcare.

REFERENCES

[1]. P. Nagaraj, V. Muneeswaran, K. J. Krishna, K. Y. Reddy, J. R. Morries and G. P. Kumar, "Identification of Skin Diseases using a Novel Deep CNN," *3rd International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India*, pp. 992-997, 2022

[2]. R. Karthik, T. S. Vaichole, S. K. Kulkarni, O. Yadav, and F. Khan, "Eff2Net: An efficient channel attention-based convolutional neural network for skin disease classification," *Biomedical Signal Processing and Control*, vol. 73, p. 103406, Mar. 2022,

[3] V. Anand, S. Gupta, S. R. Nayak, D. Koundal, D. Prakash, and K. D. Verma, "An automated deep learning models for classification of skin disease using Dermoscopy images: a comprehensive study," Multimed Tools Appl, vol. 81, no. 26, pp. 37379–37401, Nov. 2022

[4] S. Kornblith, J. Shlens, and Q. V. Le, "Do Better ImageNet Models Transfer Better?," presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2661–2671.

[5] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1717–1724.

[6] M. Liu, S. Shan, R. Wang, and X. Chen, "Learning Expressionlets on Spatio-temporal Manifold for Dynamic Facial Expression Recognition," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2014, pp. 1749–1756.

[7] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 806–813.

[8] C. Ionescu, J. Carreira, and C. Sminchisescu, "Iterated Second-Order Label Sensitive Pooling for 3D Human Pose Estimation," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2014, pp. 1661–1668.

[9] Skin-Cancer-MNIST-HAM10000. https://github.com/PROxZIMA/Skin-Cancer-MNIST-HAM10000

[10] https://flask.palletsprojects.com/en/2.3.x/

[11] https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/